Computing and Informatics, Vol. 37, 2018, 533–552, doi: 10.4149/cai_2018_3_533

ACCELERATING STENCIL COMPUTATION ON GPGPU BY NOVEL MAPPING METHOD BETWEEN THE GLOBAL MEMORY AND THE SHARED MEMORY

Tieqiang Mo, Renfa Li

College of Information Science and Engineering
Hunan University
Changsha, Hunan, 410082, China
e-mail: `852020926@qq.com, renfali@vip.sina.com`

Abstract. Acceleration of stencil computation can be effectively improved by utilizing the memory resource. In this paper, in order to reduce the branch divergence of traditional mapping method between the global memory and the shared memory, we devise a new mapping mechanism in which the conditional statements loading the boundary stencil computation points in every XY-tile are removed by aligning ghost zone to reduce the synchronization overhead. In addition, we make full use of single XY-tile loaded into registers in every stencil computation point, common sub-expression elimination and software prefetching to reduce overhead. At last detailed performance evaluation demonstrates our optimized policies are close to optimal in terms of memory bandwidth utilization and achieve higher performance of stencil computation.

Keywords: Memory mapping, GPGPU, stencil computation, ghost zones

Mathematics Subject Classification 2010: 65Y05

1 INTRODUCTION

Stencil computations mean repeated updating of values associated with points on a multi-dimensional grid, using only values in a set of neighboring points. GPUs can effectively accelerate this type of application such as computational electrodynamics [1], the solution of partial differential equations (PDEs) which applies the finite difference [2], and image processing for CT or MRI imaging [3]. But for a highly tuned implementation how to manage memory resources remains a critical problem. The challenge is to design and implement optimized algorithms that make full use of memory bandwidth and/or arithmetic units and to reduce inefficiencies due to excessive memory traffic and unnecessary computations.

GPUs just like CPUs are subject to memory bandwidth bottleneck. For example, on NVIDIA C2050, a peak double precision performance of 515 GFLOP/s and a peak memory bandwidth of 144 GB/s can be achieved, that is to say, byte-per-flop ratio indicating the balance of the memory bandwidth versus float-point operations per second (FLOP/s) is 0.28. Similar performance can nowadays be achieved by the CPU-based shared memory system, e.g., a 4-way system with a total of 482.2 GHz Opteron cores can have a peak performance of 450 GFLOP/s and a peaked bandwidth of 170 GB/s (byte-per-flop is 0.38). For all these systems the CPU delivers more bandwidth relative to floating point computation than the GPU. What is more important is the fact that the byte-per-flop is relatively low in both systems. This implies that optimizing memory access and decreasing redundant memory operations is a must even for compute bound application.

In this paper, stencil computation algorithms running on NVIDIA GPUs are optimized. Firstly, we design a novel memory mapping mechanism between global memory and shared memory of XY-tiles which extends the classical XY-tile and ghost zone to include the aligned data points of 32 bytes (8 words) of neighboring XY-tile shown in Figure 2. In this way, control flow divergence due to the conditional statement in Listing 2 can be eliminated. Secondly, in every iteration only one XY-tile's data are loaded into shared memory and further are copied into registers in the next iteration. All stencil computations about this tile are made available for these registers. Data and the partial sub-sum are stored in the temporary registers. By this way, the saved shared memory can be used to launch more other threads. Moreover, FLOPs have been further reduced by developing *common sub-expression elimination* [4] policy for symmetric 27 points stencil computations. Finally, in our implementation the software pre-fetching is used to overlap arithmetic and memory instructions to the benefit of hiding memory latency.

Comparing with inter-tile communication mechanism, policy of local computation reduces the communicating overhead of stencil computation in the neighboring tiles. Nevertheless, overlapped memory access named ghost zone overhead in the neighboring tiles may be produced in this policy. This is because of the fact that the data in the boundary points of XY-tile must be reloaded twice for the adjacent XY-tiles. We analyze the efficiency of our stencil algorithms by utilizing a model of memory traffic which takes into account the size of ghost zone due to minimum data reloading from global memory to shared memory. From this model we conclude that overhead due to ghost zone may be largely alleviated by correct reuse of data onto ghost zone in global memory or texture caches. Our implementation of the 7-point stencil is bound by memory bandwidth for both single and double precision and close to optimal on Tesla C2050, i.e., it runs only 13 % slower than a memory loading routine without considering the ghost zone overheads.

The rest of the paper is organized as follows. Section 2 deals with the CUDA programming model and algorithms behind stencil computation. In Section 3 we detail algorithm framework of stencil computation based on registers and shared memory. Section 4 deals with new memory mapping mechanism in which ghost zone overheads are adequately considered to reduce memory access traffic on the GPU. In Section 5 software prefetching mechanism is analyzed. In Section 6 we evaluate the performance of our new memory mapping mechanism. In the last section all new ideas are concluded in this paper.

2 STENCIL COMPUTATIONS ON GPUS

A stencil computation is characterized by updating each point in a structured grid by an expression depending on values on a fixed geometrical structure of neighboring grid points. The simplest example is the 7-point stencil, approximation of the 3D Laplacian operator, as shown in Figure 1 a). The update of a data point depends on the current position and its neighbors on the left, right, front, back, above and below. A more complex example is 27-point stencils shown in Figure 1 b), where an update of point (i, j, k) depends on the weighted sum of point (i, j, k) and its 26 neighbors.

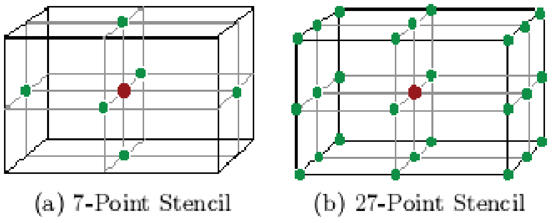


Figure 1. Stencil space structure

As an actual example, we may consider the 3D heat equation $\frac{\partial u}{\partial t} = k\nabla^2$ where ∇^2 is the Laplacian operator, and we assume a constant heat conduction coefficient and no heat sources. The following explicit finite difference scheme can solve the problem on a uniform mesh of points:

$$u^{n+1}_{i,j,k} = u^n_{i,j,k} + \frac{k\Delta t}{\Delta x^2}\left(u^n_{i,j,k-1} + u^n_{i,j-1,k} + u^n_{i-1,j,k} + u^n_{i+1,j,k} + u^n_{i,j+1,k} + u^n_{i,j,k+1} - 6u^n_{i,j,k}\right) \tag{1}$$

The superscript n denotes the discrete time step number (an iteration), the triple-subscript i, j, k denotes the spatial index. The quantity Δt is the temporal discretization (the time step) and the mesh spacing is equal in all directions. Note that in reality the formula in (1) is a 7-point computation stencil applicable only to inner grid points on a tile, and for simplicity we have omitted treatment of the boundary points.

For parallel computations on GPUs there exist thousands of threads to execute concurrently to hide memory and instruction latency: once active threads stall, warp scheduler chooses the ready threads in round robin mode. On the fine-grained level threads are arranged into completely synchronous groups of 32 threads named as one warp. Different warps are scheduled for execution independently of each other by the Streaming Multiprocessors (SM). Individual thread within the same warp is allowed to take a different execution path if conditional statements happen. However, since threads in a warp execute common instructions at a time, one execution path in one branch of that warp is disabled until different execution paths merge again. Consequently, control flow divergence significantly increases the number of the executing instructions to the sum of the instruction counts of all execution paths taken. In this situation, it is more beneficial to take branch-free implementation into consideration, especially when optimizing compute bound software.

Warps may be further grouped into 1D, 2D or 3D regular thread blocks. Warps belonging to the same thread block are executed on the same SM. The order in which different thread blocks are executed is random. Threads within a thread block can quickly synchronize and exchange data through a common shared memory. The position of a thread inside a block is described by a set of local threadIdx.x, threadIdx.y and threadIdx.z indices. Each thread block has a similar set of indices *(blockIdx.x, blockIdx.y, blockIdx.z)* locating the position of the block in the global data grid. In one policy of 3D stencil computations the grid is processed using 3D thread blocks [5]. Based on blockIdx and threadIdx coordinates threads compute global indices (i, j, k) of the processed data points.

A great many stencil optimization techniques have been previously suggested. Micikevicous [6] gave an implementation of 3D stencil (FDTD3d) in which the shared memory is used to load XY-tiles and registers are made available to save shared memory space. Phillips and Fatica [7] utilized the removing logic to reduce the number of conditional statements and made 4 texture caches available. So its memory access latency overheads increased. Recently Zhang et al. [8] statically allocated ghost zone data to threads during initialization of the thread blocks. In addition, the method avoids conditional statements, but its ghost zone in the Y-dimension are loaded in non-coalesced way. This might also lead to degraded performance.

In our memory mapping mechanism all updated stencil data in every computed point must be written into global memory. But in some applications the intermediate updated data require no writing into the global memory immediately in the defined stencil (e.g. Wave propagation solver, Jacobi solver). In this case, the temporal block optimization [5, 11, 13] could make time-dimension data available to speed up stencil computations for less memory access.

Many researchers have proposed automatic stencil code generation and performance tuning for modern multi-core heterogeneous architectures. The auto-tuning method searches through the space of parameters which may degrade performance, such as loop unrolling factor, types of memory and thread block size. This method demonstrates a portable high performance across different architectures and stencil types. Datta et al. [4] developed an auto-tuning framework for stencil computations, targeting multi-core systems, NVIDIA GPUs or Cell SPUs. Zhang et al. [8] and Christen [10] also developed the tuning framework to optimize performance on the GPU. Tang et al. [14] projected the Pochoir stencil compiler which uses a domain specific language embedded in C++ to produce high performance code for stencil computations using cache-oblivious policy for parallelism. Unat et al. [15] suggested a compiler framework named Mint using annotated C as the front-end and converting stencil computation into C code by utilizing pragmas with several levels of optimization.

In this paper a more efficient and widely used 2.5D blocking policy [6] is applied. Two-dimensional data points are used to tile the grid in the XY-plane and provide threads with the (i, j) indices of the grid points. A loop is then used to traverse the grid in the Z-dimension, providing additional k index. In this way there are no Z-dimension ghost zone in the 2.5D policy since data are processed by the thread block plane by plane. This guarantees data reuse in the neighboring XY-tile along Z-dimension and reduces plenty of memory bandwidth requirements. So this policy is superior to using 3D thread blocks because of data reuse and reduced shared memory requirements in 2.5D blocking policy. What is more, in 2.5D policy the initialization cost of the thread blocks (computing thread and grid indices, setting boundary conditions, etc.) is decreased over a larger number of grid points processed by every thread.

3 MORE EXPLOITATION OF REGISTERS IN STENCIL COMPUTATION

A novel pseudo-code implementation of stencil computation is presented in Listing 1. In registers *r1*, *r2*, ..., *r9*, values of stencil points of the same XY-plane are stored. Sub-routine load_block_ghost loads the whole XY-plane tile including ghost zone into shared memory space: sh_m from global memory; in sub-routine sh_m_regs, 9 values around stencil computation point (i, j, k) in the same XY-tile plane are copied to the registers from shared memory; in the function stencil_compute1, only the partial sub-sum is computed by adding the weighted values stored in current registers *r1*, *r2*, ..., *r9*, namely, the partial stencil result of the single XY-tile is produced by making all coefficients of the stencil kernel available to the data in one k-loop iteration.

The algorithm shown in Listing 1 runs a series of steps to finish all stencil computations. In line 9 and line 12 the input data of the first XY-tile and the second XY-tile are loaded into registers from shared memory, respectively. The updated value of every computed point is the sum of the weighted value of its neighboring points which belong to the three different planes being adjacent to each other, likewise, this value is equal to the sum of 3 partial sub-sums which can be computed by their corresponding stencil expressions in three different adjacent planes. So in line 11 and line 14 two partial sub-sums are computed for the first XY-tile plane and the second XY-tile plane, respectively. In line 15 the sum of these partial sub-sums for the present two planes is stored in the temporary variable *inter*1 (the corresponding PTX code is the register). In line 18 the input data of the third XY-tile of the current iteration are loaded into registers from shared memory. In line 19 the input data of the third XY-tile of the next iteration are loaded into shared memory from global memory or texture cache. In line 21 the loaded data of the third XY-tile in line 18 are used to compute the partial sub-sum of the third XY-tile and the first stencil computation result is obtained by further adding the value of inter1. Lines 22 and 23 initialize the subsequent iterations. In lines 25 and 26 the updated stencil value of the last XY-tile is computed. In the loop code beginning from line 17 only the data of the third XY-tile are loaded into the registers and can be used to compute the updated stencil value of the current iteration while the partial sub-sum of the other two XY-tiles comes from the temporary value inter1 of the last iteration. So the registers are fully utilized by our stencil computation and the saved shared memory can be allocated to more threads.

```
1   _global_ void stencil_compute(in, out, nx, ny, nz)
2{
     // shared memory for data point
3   extern _shared_ float sh_m[ ];
4   const uint tile_x=threadIdx.x;
5   const uint tile_y=threadIdx.y;
     // the temporary stored data in registers for XY-tile
6   float r1~r9;
     // registers for partial stencil sub-sum
7   float inter1,inter2;
     //load data into the shared memory for the first XY-plane
8   load_block_ghost(in, first ,tile_x ,tile_y ,sh_m);
9   sh_m_regs(sh_m,tile_x ,tile_y);
     // load data into the shared
10   load_block_ghost(in,second,tile_x ,tile_y ,sh_m);
     // memory for the second XY-plane
     //partial stencil sub-sum for the first XY-plane
11   inter1=stencil_compute1( r1~r9 );
12   sh_m_regs(sh_m,tile_x ,tile_y);
    //load data into shared memory for the third XY-plane
13   load_block_ghost(in,third,tile_x ,tile_y ,sh_m);
    //partial stencil sub-sum for second XY-plane
14   inter2=stencil_compute1(r1~r9);
     //partial stencil result obtained by the sum of
     // the variables  inter1 and inter2
15   inter1+=inter2;
16   out+=ix+iy*nx;
17   for(uint k=3;k<nz-1;k++){
     // load data into registers from the shared
     // memory for the third XY-plane
18       sh_m_regs(sh_m,tile_x ,tile_y);
```

```
    //initialize the next iteration
        load_block_ghost(in,k,tile_x,tile_y,sh_m);
        out+=nx*ny;
    // update current XY-tile
        out[0]=inter1+stencil_compute1(r1~r9);
    // initialize the next iteration
        inter1=inter2+stencil_compute1(r1~r9);
        inter2=stencil_compute1(r1~r9)
        }
    sh_m_regs(sh_m,tile_x,tile_y);
     // update the last XY-tile
    out[0]=inter1+stencil_compute1(r1~r9);
}
```

Listing 1. Algorithm framework of stencil computations

The popular implementation of stencil solution is made by a holistic stencil computation, that is to say, for every stencil computation tile k, all adjacent stencil data in $k - 1$, k and $k + 1$ tiles are loaded into shared memory or registers of the thread block and are computed by stencil expression. Reusing the k and $k - 1$ input tiles in the production of the next $k + 1$ tile relies on circular queue [7]. In contrast, our implementation of stencil solution is achieved by the accumulation of multiple partial stencil results. In every partial stencil calculation in a sub-routine *stencil_compute1* only one XY-tile data are loaded into registers. The acquired partial stencil results are accumulated in registers inter1 and inter2. Once stencil result accumulated from the adjacent three XY-tiles has finished this result is stored in the output array.

4 MEMORY MAPPING MECHANISM AND OPTIMIZATION

4.1 The Classical Memory Mapping Mechanism

The classical memory mapping algorithm from global memory to shared memory is illustrated with Listing 2, in which one XY-tile data are loaded into shared memory. The homogeneous implementation can be found in [6, 7]. In Listing 2 the adjacent indices of the computed stencil point are shown here only for comprehensibility. Actually, all indices are pre-computed before the k-loop at the beginning of the routine *stencil_compute*. Since the single thread computes all stencils points along the Z-axis, only the k-index changes and needs to be updated in the k-loop.

In line 6 data in a stencil point corresponding to one thread's ID is loaded into shared memory. The ghost zone is loaded into shared memory by boundary threads, which are shown from line 7 till line 11. From this figure we can find that there are many lines of conditional statements which can effectively aggregate control flow divergence of threads in one warp and increase the total number of executed instructions. By utilizing texture cache as shown in [7] the conditional statements have been removed and redundant loads of global memory have been prevented. But the texture cache is not designed to reduce latency, thus texture loads have similar cost of global loads regardless of whether or not there is a cache hit. In order to shun the conditional statements Zhang and Mueller [8] statically allocated ghost zone points to the individual threads at the start of the stencil routine. The threads first load the interior data points of XY-tile into shared memory then load ghost zone. Depending on the thread block size, some threads are loading the allocated ghost points while many other threads may be idle until all branches in one warp converge. On the other hand, the X-dimension ghost zone points are not loaded into the shared memory in a coalesced way.

```
load_block_ghost(in,k,tile_x,tile_y,sh_m)
{
 uint bx=blockDim.x+2;
 uint mx = blockIdx.x*blockDim.x + threadIdx.x;
 uint my = blockIdx.y*blockDim.y + threadIdx.y;
  sh_m[tile_x+tile_y*bx]=in[mx+my*nx+k*nx*ny];
//top and bottom ghost zones
if (tile_y==1) sh_m[tile_x+(tile_y-1)*bx]=in[mx+
    (my-1)*nx+k*nx*ny];
 if(tile_y==blockDim.y) sh_m[tile_x+(tile_y+1)*bx]
    =in[mx+(my+1)*nx+k*nx*ny];
// left and right ghost zones
if(tile_x==1) sh_m[(tile_x-1)+tile_y*bx]=
     in[mx-1+my*nx+k*nx*ny];
  if(tile_x==blockdim.x) sh_m[tile_x+1+tile_y*bx]=
    in[mx+1+my*nx+k*nx*ny];
 //corner ghost zones
  if(tile_x==1&&tile_y==1)
 {sh_m[tile_x-1+(tile_y-1)*bx] = in[mx-1 + (my-1)*nx
        + k*nx*ny];
 sh_m[tile_x-1+(tile_y+1)*bx] = in[mx-1 + (my+1)*nx
        + k*nx*ny];
 sh_m[tile_x+1+(tile_y-1)*bx] = in[mx+1 + (my-1)*nx
        + k*nx*ny];
 sh_m[tile_x+1+(tile_y+1)*bx] = in[mx+1 + (my+1)*nx
        + k*nx*ny];
   }
  _syncthreads ();
}
```

Listing 2. Ordinary memory mapping algorithm

4.2 Modeling Memory Mapping Mechanism

The purpose of modeling the memory mapping mechanism is to evaluate the efficiency of our stencil implementation in terms of memory bandwidth usage and memory overhead due to redundant data access to ghost zone. We will calculate the theoretical minimum and the algorithm constrained minimum amount of memory traffic in a single stencil computation, respectively.

Theoretically, for the global stencil memory access operations, the value of data points is loaded into shared memory once and computed, and then the result is written back to global memory once. If 4 bytes of data points are assumed for single precision floating point (SPFP) the number of bytes of memory traffic per data point is 8 bytes (4 bytes for loading and 4 bytes for writing).

For GPUs, different thread blocks executed on the stream multi-processors run independently and cannot access each other. Therefore, in stencil computation every thread block has to load the interior data of XY-tile and a layer of adjacent values targeting to calculate stencil value of peripheral points of XY-tile. The values of this peripheral layer of every XY-tile are often referred to as the ghost zone or halo which simultaneously are interior values of adjacent thread blocks. Thus, all in all, (blockDim.x + 2) $*$ (blockDim.y + 2) values of every XY-tile must be loaded into shared memory. Consequently, memory access is increased since some of the data are loaded more than once: first as interior data of XY-tile, again as the ghost zone of adjacent XY-tile. Of course, this repetitive access overhead can be lessened by enlarging the XY-tile perimeter, which produces a smaller ratio between the number of data points of ghost zone and that of interior data points in one XY-tile. Since the threads in different thread blocks cannot access each other, overhead of loading ghost zone cannot be explicitly eliminated and should be added to memory traffic of per data point.

Memory access must be aligned by 32, 64 or 128 bytes on GPUs. So loading data from global memory and writing data into global memory should be performed in a coalesced manner, in other words, threads in a warp should access the consecutive addresses within the aligned data segments. If the size of the thread block and the most frequently changing dimension of the tile is a multiple of the warp size full memory coalescence can be reached [9]. Generally, loading and writing of interior data of block can be fully coalesced so long as the tiles are properly partitioned. Since the values of the top and below Y-dimension ghost zone can also be correctly aligned in the memory accesses they are also loaded in a coalesced manner. But loading left or right X-dimension ghost zone becomes more challenging from the point of view of performance because of their non-consecutive memory access. If one tile has a thread block size of bx*by and ghost zone width of 1 the minimum number of the loading data values from the global memory is $bx*by+2bx+2(by+2)*$ 32/sizeof(float_point_number), where sizeof(float_point_number) is size of the data value in bytes: 4 and 8 bytes for single and double precision number, respectively. The number of points respectively in the interior tile and the top and bottom ghost zone is $bx*by$ and $2\,bx$ in a tile. Since the number of the minimum transaction bytes is the aligned 32 bytes, the minimum number of the loading data values for loading left and right ghost zone is $2(by + 2) * 32/\text{sizeof(float\_point\_number)}$. Further for single precision floating point numbers the loading number of one tile can be simplified as

$$bx * by + 2bx + 2(by + 2) * 8 = bx * by + 2bx + 16by + 32. \tag{2}$$

The expression (2) accounts for the minimum loading number of SPFP values including the ghost zone overhead produced by loading all data of XY-tile. One way to lessen the memory overhead is to enlarge the thread block. On the contrary, for some GPUs we can make use of the global memory caches to lessen the memory overhead, which can guarantee that a fewer number of data points in one XY-tile are directly loaded from global memory than the expression (2). In our tests on the Fermi-based Tesla C2050 with L1 and L2 global memory caches the global memory overhead brought by ghost zone can be partly or entirely cleared up if the data among different XY-tiles are reused appropriately. But for this maximized reuse the scheduling order of warps must be controlled and cannot be implemented in the current GPUs.

4.3 New Memory Mapping Mechanism

For expression (2) it must be divisible by 32 for more coalesced memory accesses in 32 threads of one warp, that is to say, the value of expression $(bx * by + 2 * bx + 16 * by + 32)$ mod 32 is zero. So bx and by must be equal to a multiple of 16 and 2, respectively. Further, in our test on Tesla C2050 with CUDA toolkit Visual Profiler if the size of XY-tile is 32×6, all warps show higher occupancy (defined as Active Warps/Maximum Active Warps). Then the picked XY-tile size is 32×6. On the other hand, for the ghost zone in every XY-tile Y-dimension width is 2 lines (top and bottom) and X-dimension aligned overhead is 16 words or 64 bytes (for SPFPs 8 words or 32 bytes on left and right side, respectively). So the size of loaded tile is 48×8, as shown in Figure 2. In order to eliminate threads divergence brought by conditional statements in Listing 2 all threads may be mapped onto two threads blocks of size 48×4. The detailed pseudo-code implementation is shown in Listing 3. In this way, the number of bytes of memory traffic per data point is 12 bytes (4 bytes in writing and 8 bytes in loading which consists of 4 bytes data and 4 bytes ghost zone overhead).

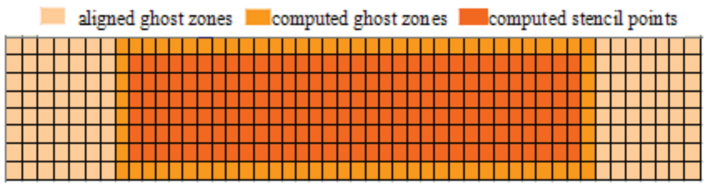


Figure 2. New-mapped XY-tile

```
1  load_block_ghost_new(in,k,sh_m)
2  { // change into one dimension index
3   uint  index= (threadIdx.y * blockDim.x) + threadIdx.x;
4   uint  tile_y1=index/48;
5   uint  tile_y2=tile_y1+4;
6   uint  tile_x=index%48;
7   int ix = blockIdx.x*blockDim.x + tile_x - 8;
8   int iy1 = blockIdx.y*blockDim.y + tile_y1 - 1;
9   int iy2 = blockIdx.y*blockDim.y + tile_y2 - 1;
10  // loading the data into the shared memory
11  sh_m[tile_x + tile_y1*48] = in[ix + iy1*nx + k*nx*ny];
12  sh_m[tile_x + tile_y2*48] = in[ix + iy2*nx + k*nx*ny];
13  _syncthreads();
}
```

Listing 3. New memory mapping algorithm

In global memory writing full coalescence can be achieved for the proper block size. If all interior data points of tile have an appropriate alignment in the memory, that is to say, for every thread block, every interior data value always starts a 128-byte aligned global memory address, writes of every warp can be coalesced into one 128-byte access on the GPU Compute Capability 2.0 architecture or into two 64-byte accesses on the GPU Compute Capability 1.x architecture. On the contrary, loading data in one warp may demand two shared memory accesses for some warps. The mapping to a thread block size of 48×4 produces misalignment, since 48 is not divisible by 32, namely, some warps have to access non-contiguous memory areas. Further the accessed data on the left ghost zone starts with a 32-byte offset from the 128-byte aligned interior data. In this way, one loading operation may be performed by the hardware as a combination of a separate 32-byte access and at least one other access. But it does not reveal significant penalty by our performance tests if the data is accessed by textures.

5 SOFTWARE PREFETCHING

Software prefetching is a well-known technique to overlap memory access latency with computation. In our prefeching mechanism the prefetched data in a k-iteration is used for stencil computation in $(k + 1)$-iteration. Concretely, this prefetching mechanism is implemented in the following sequence of Listing 1.

1. *load_block_ghost* – initialize the stencil computation by loading the first two XY-tiles from the global memory or texture cache to the shared memory (in line 10 and line 13).
2. *sh_m_regs* – copy the data loaded from the shared memory in the previous iteration to register for the current partial stencil computation sub-sum (in line 18).

3. *load_block_ghost* – load data for the next iteration from global memory or texture cache to the shared memory (in line 19).
4. *stencil_compute1* – compute the weighted sub-sum value for one XY-tile in the current iteration by utilizing data in registers (in line 21–23)
5. go to step 2 (iteration of the k-loop).

The data loaded from the global memory in step 3 is not promptly used in the current iteration. But the computations in step 4 are dependent on the data of being copied into registers in step 2, namely in the current iteration, which is loaded into the shared memory in the previous k-loop iteration. Therefore, operations of steps 3 and 4 can be overlapped by the hardware as they have no common operands. In this way, the global memory access latency can be hidden by performing the arithmetic instructions in step 4. While the warp scheduling mechanism can overlap arithmetic operations and memory access operations our prefetching method has more opportunity for the scheduled warps and leads to a higher occupancy.

6 PERFORMANCE EVALUATION

In our performance evaluation performance constraints of the available memory bandwidth are evaluated by memory bandwidth benchmarks and the memory mapping mechanism above. The computation constraints are evaluated by counting the number of instructions in the assembly PTX code produced by the compiler.

Architecture	Clock Cycle	Peak Memory	SPFP	DPFP
		Bandwidth	Performance	Performance
	GHz	GB/s	GFLOP/s	GFLOP/s
Tesla c2050	1.115	144	1 030	515

Table 1. Performance characteristics of Tesla C2050

First, we run a memory transit routine with zero-overhead shown in Listing 4 for 3D cube grids of single precision data whose memory access traffic per data point is 8 bytes (4 bytes for loading and writing respectively). Since there is no communication among different XY-tile threads and no ghost zone overhead for this transit routine, we can get the theoretical lower bound memory access throughput and an absolutely idealized constraint on stencil performance. Secondly, we run the routine transmitting all the data which has a thread block size of 32×6, Y-dimension bottom and top ghost zone width of 1, respectively, and aligned X-dimension left and right ghost zone size of 8 words, respectively. Then we can evaluate the efficiency and memory overhead of our new memory mapping mechanism by comparing the results of two benchmarks. Thirdly, we estimate how close the single precision float stencil computation by our novel memory mapping mechanism is to the theoretical lower value of 8 bytes per data point in memory access without the overhead of ghost zone. Finally, we compare the performance among different implementations of stencil computation.

We measure the performance on Tesla C2050 with error correction turned off whose main hardware performance parameters are enumerated in Table 1. During the tests we guarantee the grid size nx and ny is multiples of the thread block size. All the codes were compiled by utilizing NVIDIA CUDA 5.0 for the 2.0 GPU architecture.

```
_global_ void cube_transit(source, target, nx, ny, nz)
{
   const int ix=blockIdx.x*blockDim.x +threadIdx.x;
   const int iy=blockIdx.y*blockDim.y+threadIdx.y;
   For (int k=0;k<nz;k++)
      target[ix+iy*nx+k*nx*ny]=source[ix+iy*nx+k*nx*ny];
  }
```

Listing 4. Zero-overhead memory transit routine

6.1 Measure Memory Bandwidth of Traffic Without Ghost Zone Overhead

The global memory bandwidth is estimated by running the benchmark in Listing 4. The k-loop structure in *cube_transit* routine represents the loading and writing of 2.5D-blocking implementation. We measure the latency of transmitting a 3D data cube of SPFP values inside the global memory. The effective memory bandwidth can be calculated by the expression

$$\frac{nx \cdot ny \cdot nz \cdot (bytes\_per\_point)}{t \cdot 2^{30}} \text{ (GB/s)} \tag{3}$$

where t is the latency in seconds of one *cube_transit* call and can be obtained by n executions of *cube_transit* and the *bytes_per_point* is 8 bytes. In Table 2 we give the calculated bandwidth *bytes_per_point* (BPP), iteration time t and the throughput in data points per second gained by cube size of 256 × 256 × 256. Similarly, the throughput in the selected two other cube sizes of 192 × 192 × 192 and 512 × 512 × 512 can also be calculated easily. So in the last column of Table 2, the average throughput is produced by the three different cube sizes. The average memory bandwidth of *cube_transit* is approximately 75 % of the theoretical memory bandwidth on Tesla C2050 (comparing with Table 1).

Architecture	cube_transit			
	256 × 256 × 256			Avg. Throughput
	BPP [GB/s]	[GP/s]	t [s]	[GP/s]
Tesla C2050	112.7	14.1	1.20E−03	1.35E+01

Table 2. Memory bandwidth tests using benchmark cube_transit

6.2 Measure Memory Bandwidth of New Memory Mapping Mechanism

The efficiency of our new memory mapping mechanism can be analyzed by the routine *cube_ghost_transit* shown in Listing 5. What is more, by simply configuring data type and parameters of texture cache, this mapping policy shows increasing performance improvements in texture cache. Table 3 accounts for this result for cube $256 \times 256 \times 256$, where the memory bandwidth BPP is calculated by the expression (3) but the value of *bytes_per_point* is 12 bytes.

In comparison with the peak memory bandwidth (144 GB/s) given in Table 1 the throughput ($12.8 \times 12 = 153.6$ GB/s) of our texture access implementation given in Table 3 shows much advantage on Tesla C2050. It means there is much reuse of data in the ghost zone in the texture caches. In fact, we have proved it by using tools of Visual Profiler. On the other hand, It is significantly slower ($8.7 \times 12 = 104.4$ GB/s) to have access to the global memory for reduced data reuse. All these results demonstrate the texture caches can effectively eliminate the performance downside of the misalignment brought by ghost zone.

```
_global_ void cube_ghost_transit(in, out ,nx,ny,nz)
{
 const int ix=blockIdx.x*blockDim.x+threadIdx.x;
 const int iy=blockIdx.y*blockDim.y+threadIdx.y;
 const uint tile_x=threadIdx.x;
 const uint tile_y=threadIdx.y;
 // calculate the global and shared memory indices t1,
 //t1,t2, i1 and i2 acts like Listing~3
 out+=ix+iy*nx;
 for (int k=0;k<nz;k++) {
   sh_m[t1]=in[i1];
    sh_m[t2]=in[i2];
    i1+=nx*ny;
    i2+=nx*ny;
    _syncthreads();
    out[0]=sh_m[tile_x+8+(tile_y+1)*48];
    out += nx*ny;
    _syncthreads();
   }
  }
```

Listing 5. Bandwidth measure routine with ghost zones

The average transferred number of points of cube in our new global memory mapping routine *cube_ghost_transit* is computed as 12.8 GP/s in Table 3 in contrast to 13.5 GP/s in routine *cube_transit* shown in Table 2. Explicitly there is no ghost zone overhead in the running of the former. But ghost zone overhead (only reading the ghost zone and no writing the ghost zone) exists in the running of the latter.

So the average throughput of the latter is less than the former, that is to say, the latter needs more bytes of memory accesses traffic than the former in processing a data point of XY-tile. Further since the memory access traffic of the former is 8 bytes/point (4 bytes in reading and writing respectively) the effective memory access traffic of the latter can be computed as $13.5/12.8 \times 8 = 8.4$ bytes per data point in every XY-tile which is approximately 5 % more than the idealized memory access traffic lower bound value of 8 bytes per data point (without the ghost zone overhead). From Table 3 evidently the average performance is inferior to that achieved by $256 \times 256 \times 256$ cube, which is due to a lower cache reuse as can be revealed by the Visual Profiler of NVIDIA performance tools.

Architecture	Memory Type	cube_ghost_transit			
		$256 \times 256 \times 256$			Avg. Throughput
		BPP [GB/s]	[GP/s]	t [s]	[GP/s]
Tesla C2050	Global	126	10.5	1.61E−03	8.7
	texture	161	13.4	1.26E−03	12.8

Table 3. Memory bandwidth tests using benchmark cube_ghost_transit

6.3 Performance of Single Precision Floating Point Stencil Computation

There are 53 FLOPs (27 multiply and 26 add operations in all) in the PTX assembly code of the general 27-point stencil. By utilizing Common Sub-expression Elimination the number of FLOPs can be reduced to 18. On Tesla C2050 successive multiplication and addition operations are combined into a single hardware instruction: Fused Multiply Add (FMA). We summed up the assembly of the stencil routines compiled for Tesla C2050 GPU by using cuobjdump. There are 53 instructions in an iteration of the k-loop, including 27 floating point instructions (26 FMA and 1 FMUL) and other auxiliary instructions for loop branch, memory access, data copying and thread synchronization. If all instructions are executed with the peak floating point instruction number of 515 billion instructions per second, the computed maximized throughput for the general 27-point stencil is $515/53 = 9.7$ GP/s.

Listing 2 reveals the throughput of general 27-point stencil computation on Tesla C2050, along with memory bandwidth by the *cube_ghost_transit* routine and the computed constraint (9.7 GP/s) above. For the general 27-point stencil the constraint of the maximized instruction throughput is lower than the streaming memory bandwidth of *cube_ghost_transit* routine, therefore the 27-point stencil implementation is compute bound. In Figure 3, the 27-point stencil performance verges on the computed constraint above. This demonstrates that the time to access memory largely overlaps with the time to execute the instructions through GPU hardware. There are two reasons for it. Firstly, the implementation has a high occupancy, namely, there is a large number of resident warps per stream multi-processor. Consequently, when stencil data is transited there is a high likelihood that the stream multi-processors schedule instruction to execute for some warps. Secondly, the software prefetching leads to an explicit overlap between the memory transit and computations.

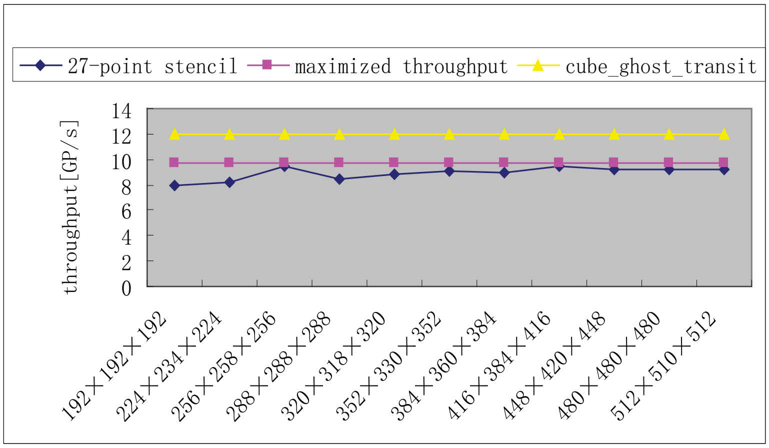


Figure 3. Bandwidth comparison among 27-point stencil, cube_ghost_transit routine and the computed constraint

In Table 4 the average performance of our stencil computation is given. The percent fraction of the peak Instruction per Cycle (IPC) is acquired by Visual Profiler. On average, the stencil routine is executed at 93 % of peak IPC and acquires approximately 46 % of the peak FLOP/s performance. In the last column, the percent fraction of the *cube_ghost_transit* routine throughput is listed as 71 % and achieves relatively lower fraction of the streaming memory bandwidth. This indicates that 27-point stencil is compute bound for Tesla C2050.

Architecture	27-Point Stencil			
	IPC [%]	[GP/s]	GFLOP/s	cube_ghost_transit [%]
Tesla C2050	93	9.1	472	71

Table 4. Average performance of single precision 27-point stencil

We ran the tests ourselves in the best thread block size of FDTD3d 7-point stencil [6] and listed the result in Table 5. From this table our new mapping implementation is almost twice faster than FDTD3d on Tesla C2050. It demonstrates that the loading policy of the shared memory has a great impact on performance. From the generated PTX code in the FDTD3d implementation 100 instructions are approximately generated in one iteration of the k-loop like Listing 1. In this NVIDIA's implementation, traditional loading policy is made available. However, the control flow divergence in traditional policy leads to non-synchronization of

32 threads in one warp, namely, two different groups' sequential execution. In this way, NVIDIA's implementation brings more instructions. The performance of the tuning framework devised by Zhang et al. [8] and Christen [10] listed in Table 5 is inferior to ours. It is because of the fact that the control flow divergence of warps brought by conditional statement is removed and higher occupancy holds due to our problem-specific hand-tuned optimization. But for another problem domain in which the intermediately processed data requires no writing into the global memory immediately in a stencil computation, a higher performance demonstrates with temporal blocking pipeline on the GTX 285 in Table 5 by Nguyen et al. [11], which is approximately 1.4 higher than ours. Recently in Naoya et al. [16] their stencil original data in ghost zone are approximately replaced by the data of nearby boundary points in the same computed XY-tile. So the communication overheads among different thread blocks require no consideration. Its optimized performance can reach 131.4 GFLOP/s shown in Table 5.

6.4 Performance of Double Precision Floating Point Stencil Computation

Table 5 also summarizes the performance of different kinds of stencil implementations. In the last two columns results of some double precision stencils are listed. Since processed data of double precision computation are twice as large as those of single precision memory bandwidth bound double precision implementation should perform roughly at 50 % of the single precision throughput. This case can be revealed by our implementation of 7-point stencil shown in Table 5. However, for the compute-bound 27-point stencil performance of double precision is only 32 % performance of single precision. Further by Nvidia's cuobjdump tool the total number of instructions of double precision implementation is about 91 more than that of single precision implementation. It is because of the fact that the coefficients of the stencil kernel are stored in the constant memory of Tesla C2050. For the single precision case the instructions get stencil coefficients directly from the constant memory, but for the double precision case the compiler remarkably produces additional instructions which load the coefficients from the constant memory to the registers. In this way, the occupancy decreases due to a relatively larger register usage. In reverse, the double precision implementation of 7-point stencil stores the coefficients in registers and achieves a relative 50 % more float point than its single precision version.

7 CONCLUSIONS

In this paper, we have devised a new memory mapping mechanism between shared memory and global memory to remove the conditional statement of the surrounding XY-tile stencil computation points by combining coalesced memory accesses of the GPU with aligned ghost zone overhead. In addition, in our stencil computation only one XY-tile is loaded into registers and the other two XY-tiles utilize the last

Policy	Type	Single Precision		Double Precision	
		GP/s	GFLOP/s	GP/s	GFLOP/s
Our implementation	7 points	12.3	97	6.5	51
	27 points	8.9	472	3.1	153
FDTD3d (Nvidia)	7 points	6.7	54		
Holewinski et al. [13]	7 points	5.9	48	3.2	26
Kamil et al. [12] GTX280	7 points			1.6	13
Nguyen et al. [11]	7 points	9.2	74	4.6	37
	7-p time steps	17	136		
Christen et al. [10]	7 points	3	24		
Zhang and Mueller [8]	7 points	10.9	87	5.7	46
Naoya et al. [16]	7 points	16.4	131		

Table 5. Performance of many kinds of different stencil computation implementation

results in the temporary registers. In this way a great deal of shared memory is saved for storing more values of XY-tiles and reduce shared memory clashes. We make full use of a thread block size of 32×6 to guarantee only two coalesced memory loading operations and no conditional statements. So it is significantly important for many-threaded GPUs to alleviate control flow divergence of a warp. On Tesla C2050 the acquired memory traffic for single precision data is 8.9 bytes per stencil computation point, only 11 % worse than the idealized value of 8 bytes about which only reading from and writing into global memory is considered and ghost zone overhead may be omitted. In comparison with previous implementation execution of our code has greater bandwidth than all other stencil executions in which the intermediate results must be written into global memory. At last our Common Sub-expression Elimination decreases the number of FLOPs in the 27-point stencil from 53 to 18.

As illustrated in the above sections, the previous optimization of stencil computation places stress on use of registers, more locality of texture cache, less control flow divergence and more effective coalesced accesses method. In our implementation, the new mapping mechanism guarantees more coalesced memory accesses and completely eliminates control flow divergence of loading boundary data points which constitutes ghost zone overhead in tiles. On the other hand, control flow divergence is weakness and bottleneck in GPU. Once it happens, two different thread groups of 32 threads in warps have to execute in sequence rather than in parallel mode. In this way, despite some other overhead, such as calculation in mapping addresses, the performance of stencil computation is improved.

Acknowledgements

The authors would like to thank to the anonymous reviewers for their valuable comments and suggestions. This work was partially supported by the National Natural Science Foundation of China (No. 61173036).

REFERENCES

[1] Taflove, A.—Hagness, S. C.: Computational Electrodynamics: The Finite-Difference Time-Domain Method. Artech House Publishers, Boston, 2005.

[2] Smith, G. D.: Numerical Solution of Partial Differential Equations: Finite Difference Methods. Oxford University Press, Philadelphia, 2004.

[3] Cong, J.—Huang, M.—Zou, Y.: Accelerating Fluid Registration Algorithm on Multi-FPGA Platforms. 2011 International Conference on Field Programmable Logic and Application, Chania, USA, September 2011, pp. 50–57, doi: 10.1109/FPL.2011.20.

[4] Datta, K.—Williams, S.—Volkov, V.—Carter, J.—Oliker, L.—Shalf, J.—Yelick, K.: Auto-Tuning the 27-Point Stencil for Multicore. 4th International Workshop on Automatic Performance Tuning (iWAPT 2009), 2009, pp. 75–84.

[5] Meng, J.—Skadron, K.: A Performance Study for Iterative Stencil Loops on GPUs with Ghost Zone Optimizations. International Journal of Parallel Programming, Vol. 39, 2011, No. 1, pp. 115–142, doi: 10.1007/s10766-010-0142-5.

[6] Micikevicius, P.: 3D Finite Difference Computation on GPUs Using CUDA. Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units (GPGPU-2), ACM, New York, NY, USA, 2009, pp. 79–84, doi: 10.1145/1513895.1513905.

[7] Everett, H. P.—Massimiliano, F.: Implementing the Himeno Benchmark with CUDA on GPU Clusters. IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2010), April 19–23, 2010, IEEE Computer Society Atlanta, GA USA, pp. 1–10.

[8] Zhang, Y.—Mueller, F.: Autogeneration and Autotuning of 3D Stencil Codes on Homogeneous and Heterogeneous GPU Clusters. IEEE Transactions on Parallel and Distributed Systems, Vol. 24, 2013, No. 3, pp. 417–427, doi: 10.1109/TPDS.2012.160.

[9] NVIDIA Corporation: CUDA C Programming Guide. Version 5.0. 2012.

[10] Christen, M.—Schenk, O.—Burkhart, H.: PATUS: A Code Generation and Autotuning Framework for Parallel Iterative Stencil Computations on Modern Microarchitectures. 25th IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2011), May 16–20, 2011, Anchorage, AK, pp. 676–687, doi: 10.1109/IPDPS.2011.70.

[11] Nguyen, A.—Satish, N.—Chhugani, J.—Kim, C.—Dubey, P.: 3.5-D Blocking Optimization for Stencil Computations on Modern CPUs and GPUs. Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis (SC '10), November 13–19, 2010, IEEE Computer Society New Orleans, LA USA, 2010, pp. 1–13.

[12] Kamil, S.—Chan, C.—Oliker, L.—Shalf, J.—Williams, S.: An Auto-Tuning Framework for Parallel Multicore Stencil Computations. 2010 IEEE International Symposium on Parallel and Distributed Processing (IPDPS 2010), April 19–23, 2010, IEEE Computer Society Atlanta, GA USA, pp. 1–12.

[13] Holewinski, J.—Pouchet, L.-N.—Sadayappan, P.: High-Performance Code Generation for Stencil Computations on GPU Architectures. Proceedings of the 26th ACM International Conference on Supercomputing (ICS '12), ACM, New York, NY, USA, 2012, pp. 311–320, doi: 10.1145/2304576.2304619.

[14] Tang, Y.—Chowdhury, R. A.—Kuszmaul, B. C.—Luk, C.-K.—Leiserson, C. E.: The Pochoir Stencil Compiler. Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA '11), 2011, pp. 117–128, doi: 10.1145/1989493.1989508.

[15] Unat, D.—Cai, X.—Baden, S. B.: Mint: Realizing CUDA Performance in 3D Stencil Methods with Annotated C. Proceedings of the 25th International Conference on Supercomputing, May 31–June 4, 2011, ACM, Tuscon, Arizona, USA, pp. 214–224, doi: 10.1145/1995896.1995932.

[16] Maruyama, N.—Aoki, T.: Optimizing Stencil Computations for NVIDIA Kepler GPUs. First International Workshop on High-Performance Stencil Computations (HiStencils '14), January 21, 2014, Vienna, Austria.

Tieqiang Mo is currently Assistant Professor in computer science and electronic engineering at Hunan Univerisity. He received his Ph.D. degree from Hunan University, China in 2016. Currently, he is engaged in the research of parallel and distributed algorithms, especially skilled in heterogeneous algorithms and automatic code generation.

Renfa Li received his Ph.D. degree in electronic engineering from Huazhong University of Science and Technology, Wuhan, China, in 2002. He is currently Professor of computer science and electronic engineering with Hunan University, Changsha, China. He is the Director with the Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha, China. He is also an expert committee member of the National Supercomputing Center in Changsha, China. His major interests include computer architectures, distributed computing systems and code optimization. He is a member of the Council of CCF and a Senior Member of ACM.

Computing and Informatics, Vol. 37, 2018, 553–580, doi: 10.4149/cai_2018_3_553

EFFORT ESTIMATION FOR SERVICE-ORIENTED COMPUTING ENVIRONMENTS

Siba Mishra, Chiranjeev Kumar

Department of Computer Science and Engineering
Indian Institute of Technology (Indian School of Mines)
Dhanbad, 826004
Jharkhand, India
e-mail: `sibamishracse@gmail.com, k_chiranjeev@yahoo.uk`

Abstract. The concept of service in Service-Oriented Architecture (SOA) makes possible to introduce other ideas like service composition, governance and virtualization. Each of these ideas, when exercised to an enterprise level, provides benefits in terms of cost and performance. These ideas bring many new opportunities for the project managers in making the estimates of effort required to produce SOA systems. This is because the SOA systems are different from traditional software projects and there is a lack of efficient metrics and models for providing a high level of confidence in effort estimation. Thus, in this paper, an efficient estimation methodology has been presented based on analyzing the development phases of past SOA based software systems. The objective of this paper is twofold: first, to study and analyze the development phases of some past SOA based systems; second, to propose estimation metrics based on these analyzed parameters. The proposed methodology is facilitated from the use of four regression(s) based estimation models. The validation of the proposed methodology is cross checked by comparing the predictive accuracy, using some commonly used performance measurement indicators and box-plots evaluation. The evaluation results of the study (using industrial data collected from 10 SOA based software systems) show that the effort estimates obtained using the multiple linear regression model are more accurate and indicate an improvement in performance than the other used regression models.

Keywords: Effort estimation, orchestration, SOA, regression, web services

Mathematics Subject Classification 2010: 68N30

1 INTRODUCTION

Prevalent business and industrial organizations around the globe adheres SOA style for building business, commercial and financial software applications. This is because SOA provides a promising way for addressing many problems related to the integration of heterogeneous applications in a distributed environment [1]. SOA is an architectural approach for developing enterprise level business systems using *loosely coupled* interoperable services. Services – the core component of SOA is defined as a *logical encapsulation* of self-contained business functionality. Technically, the term *self-contained functionality* suggests that any changes to the available services could be incorporated without affecting other services of the system. Moreover, the use of services in SOA increases the overall *flexibility* and adds improved flow of *functionality*. Due to this implicit advantage, in the last decades, SOA emerged up quite rapidly and has numerous applications in the field of biotechnology, health care systems, communication networks, irrigation, mass-customizations and e-health support services [38, 39, 40, 41, 42, 43, 44, 45, 46].

From these broad applications and advantages of SOA, it is clear that the design and development activities are different from that of traditional programming paradigms [2, 3]. Further, the development of SOA systems introduces many new concepts, technological factors and architectural issues for building complex business applications. These *new concepts* include services, messages, property of orchestration, loose coupling and many more [3, 4]. Also, developing SOA systems for business, financial and banking sectors are much more complex and expensive specifically in terms of resources and schedules. In the context of SOA project management, these new concepts and principles add many complex issues that are different from traditional software development paradigms [6]. In this way, the development of SOA systems is different from traditional software development. Moreover, from having an efficient effort estimate, a valid conclusion about the SOA system implementation phase are drawn for some measurement dimensions.

Estimation of effort[1] is an essential component of software project management. It is also a prerequisite feature of any software process, whether it is the design, testing, development, usability or the application as a whole. Generally, estimation depicts the way things will happen in the future based on the present conditions. In fact, it is an approximation for which some outcome is expected instead covering the set of possible outcomes. Having an efficient effort estimation technique is widely perceived by the business analysts and project managers. This is because an efficient estimation methodology helps in utilizing the project resources conveniently and thus helpful in avoiding project overestimation and late delivery [15]. As above mentioned, the development activities of SOA systems are different from traditional softwares. Thus, the existing software size and effort estimation techniques are not

[1] In the field of Software Engineering, "effort" estimation is also known as "cost" estimation. In this section and throughout the paper, both the terms have been used interchangeably.

adequate to capture specific development features for influencing the development effort parameters in building of the SOA based software applications.

For this objective and the aforementioned issues, we adopt a similar classification framework proposed by Lowe et al. [7] and Mendes et al. [8] for predicting the design and authoring effort of web hypermedia and software applications. However, our contribution includes the following additional research, i.e., the usage of metrics is designed and proposed considering various SOA related artifacts like orchestration, services, principle of loose-coupling and messages on different scales for estimating the development effort. In general, the service design phase covers the modeling of total number of processes – that is tasks and other constituent elements (like looping, parallel flow and synchronization) required for building SOA systems. This suggests by analyzing the service design phase, different measures could have been obtained for the SOA systems and that is considered as a suitable predictor of effort.

In our work, we measured some interesting theories relevant to service design phase and proposed some associated cost drivers necessary for predicting the SOA systems development effort. The proposed metrics highlights the design related issues of SOA systems mainly supported from the environment configuration and total size. Besides, following are the highlights of this work:

- The proposed approach geared up from an initial study with a motivation of identifying some *design measures* related to SOA systems.
- Introduction of *novel metrics* for facilitating the estimation methodology for the identified parameters of the initial study.
- For evaluating the accuracy (in terms of predictive power) for the obtained results, rigorous experiments were carried out using some statistical significance tests, performance measurement indicators and box-plots evaluation.

The rest of the paper is organized as follows: Section 2 discusses related works. Section 3 presents the principles of methodology relevant to our work. The proposed work has been introduced in Section 4. Section 5 reports and analyzes the empirical results and discussions. Section 6 concludes the paper.

2 RELATED WORK

So far in the literature, adequate attempts have been made to solve the problem of effort estimation for *traditional softwares* [15, 16, 17, 18, 19, 20, 21, 22]. These techniques[2] are classified mainly into probabilistic and statistical, expert judgement, analogy, algorithmic and machine-learning based estimation techniques [16, 48].

Generally, *probabilistic models* use the Baye's theory and probabilistic method for predicting the development effort. The *statistical models* use the method of regression for estimating the software development effort for some past data. The

[2] In this section and throughout the paper, the term techniques and models have been used interchangeably.

expert judgement models [18] involve consultation with one or more local experts, having knowledge about the core design and development environment or application domain in context to software project management. *Analogy models* estimate the development effort of a target project as a function of known efforts from a set of similar historical projects [19, 20]. In *algorithmic models*, the development costs are analyzed using some mathematical formula linking the costs with metrics to produce an estimated output. Next, the formula is applied to a formal model arising from the analysis of historical data. The *machine learning techniques* use both supervised and unsupervised learning techniques, for the training purpose and the development effort are calculated using a set of historical datasets. Each and every above mentioned estimation techniques are used based on some certain conditions and requirements. Research has still been in progress for investigating the best prediction technique.

In the last decades, SOA approaches are used for developing software applications sourced as virtual hardware resources, including on-demand and utility computing [49]. SOA uses both *services* and *messages* to support the development of low-cost distributed applications [50]. Moreover, recently the service-oriented technologies gained the mainstream attention quite remarkably, as SOA addresses a promising way for creating the basis of agility using which the software industries deliver more flexible business processes [49]. Despite the wide practice of using SOA, plethora amount of research has already been devoted to service-orientation research road-maps, challenges, fundamental perspectives, evolution, re-usability, governance and composition [50, 51, 53, 54, 55]. However, we believe that the research on effort estimation of SOA systems is definitely a novice option with many interesting challenges and new opportunities in terms of future research. Also, the research on effort estimation of SOA systems are very scarce in the literature. In the literature of traditional software development effort estimation, most of the work focused on *algorithmic techniques*, whereas in SOA system effort estimation, the *algorithmic* along with *probabilistic* techniques covers more than half of the reported work. To the best of our knowledge, no *analogy*, *statistical* and *machine learning* based estimation techniques has been reported in literature. Nevertheless, some approaches [10, 11, 12, 13, 14, 23, 47] are worth mentioning facilitating an efficient estimation, without any consideration of *predictive accuracy* for the set of past project data.

For example, Liu et al. [13] proposed a *probabilistic approach* using the *Bayesian net model.* This technique focuses only on different *service governance* processes. This method [13] highlights the Bayesian approach for predicting the development effort and *more improvement needs to be incorporated* for providing a systematic and accurate prediction, as suggested for their future work. However, we believe that this objective may be fulfilled using some detailed indicators and mathematical models in the analysis procedure. O'Brien [12] from NICTA, Australia also introduced a *probabilistic based* framework [SMAT-AUS] for capturing various aspects of SOA projects. Furthermore, the proposed framework used for determining the *scope and development effort* by considering the *technical*, *social-cultural*, *maturity models* and other *organizational* aspects. The SMAT-AUS framework is in its development stage and not yet fully developed. The complete framework may provide an efficient way for determining the scope and effort of SOA systems. The limitations of the above mentioned probabilistic approaches [12, 13], besides not being fully developed, are that it does not consider adequate cost drivers for balancing the trade-off between the estimation methodology and SOA systems.

The authors [11] introduced an *algorithmic* framework based on Divide and Conquer (D & C) technique for estimating the cost of building SOA softwares. The *novelty* of this approach is that the estimation mechanism is employed by focusing only the different types of services. However, this approach besides being incorporated as an efficient framework is also limited for not providing proper validation for the set of past project data. Additionally, Gomes [14] – A SOA architect of Unimix, introduced an *algorithmic* approach for estimating and counting SOA projects using service candidate descriptions, Web Services Description Language (WSDL) and XML Schema Definition (XSD). The proposed method is presumed as an *algorithmic approach* because of the use of *function points*. The proposed technique is suitable only for *small* sized projects and fails for large size and complex SOA systems. In addition, authors in [10] proposed a qualitative *expert judgement based approach* to *judge* the effort of different SOA styled project proposals before implementing the Web Services Compositions (WSCs). The authors borrowed D & C approach [11] to narrow down the problem of effort judgement of an entire SOA implementation rather individual Web Services. Moreover, the authors introduced a novel approach for determining the *effort factors* of WSCs, using *classification matrix* and *hypothesis*. This approach neither considers any case study evaluation nor provides any proper validation for the past project data.

Therefore, owing to the above reasons, some new metrics relevant to the design phase of SOA system are introduced in this paper. The estimation methodology comprises proposed metrics and statistic based regression techniques has been found as a suitable candidate for solving the problem of SOA system effort estimation. Furthermore, proper validation (predictive accuracy) has been incorporated into the calculated predicted values using some commonly used *performance measurement indicators* and *box-plots evaluation* for the data collected from multiple sources Indian software organization.

3 PRINCIPLES OF METHODOLOGY

In this section, we provide an overview about the principles of methodology and some essential background relevant to our work. We have used four statistic based regression techniques for calculating the predicted values. From the generated models, the development effort is computed based on some contextualized design related issues of SOA systems.

We have used regression techniques for predicting the development effort of SOA systems. "Regression" is one of the most popular statistical technique used

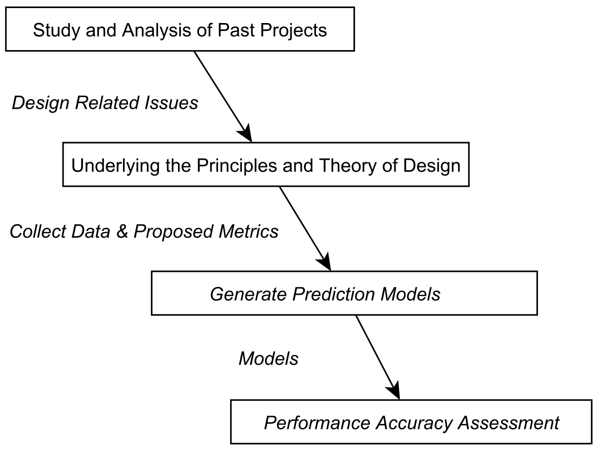

Study and Analysis of Past Projects
Design Related Issues
Underlying the Principles and Theory of Design
Collect Data & Proposed Metrics
Generate Prediction Models
Models
Performance Accuracy Assessment

Figure 1. Flow diagram of our proposed work

commonly in the field of estimation. Typically, regression models represent the relationship between independent and dependent variables of a used dataset. Moreover, the most important identified parameters from the design phase of SOA systems are *initial heads*, *configuration environment*, *definition* and *length*. From these identified parameters, a set of metrics for the SOA system is proposed with the notion that these metrics conceived to have some significant impact on the total size of the application[3].

Figure 1 shows the basic flow diagram of our proposed approach. The different stages of the flow diagram are described below.

- The *objective* of the *first stage* is to study and analyze the design and development related issues of SOA systems. The *output* of this phase classifies the identified contextualized *design related issues* relevant to SOA systems.
- The *second stage* focuses on grasping some basic theories and principles relevant to design phase of SOA systems. Here, the main *objective* is to design some *key metrics*, *cost drivers* and *other theories* relevant to service design phase. One more *aim* of this phase is to collect data for these identified issues. The *output* of this stage provides *data* that are to be used for generating regression models. This stage also facilitates some key design related issues based upon which metrics are designed and proposed.
- The *third stage* emphasizes the generation of different regression models for some past project data. The stage wraps up soon after the generation of prediction models. This stage gets final completion of two defined *objectives*:

[3] An application is a process or a task implemented as a web service or a scripting language like Java Script or an orchestrated task or a fully integrated application.

   - *Validation of data values for the used dataset*: The objective of this sub-stage is to identify the *missing and influential* data-points for the used dataset and normalize the collected data values.
   - *Selection of appropriate variables and regression model*: This sub-stage assists in selecting some appropriate *dependent and independent variables* from the used dataset and choosing an appropriate regression model.

- The *final stage* illustrates performance assessment (predictive accuracy) of the generated estimation models. This is achieved with the help of some commonly used performance measurement indicators and statistical significance tests. The *objective* of this stage is to ensure the *accuracy level* of the calculated values.

The data of the used dataset constitute 10 different SOA styled applications. The dataset aimed at the following objectives.

1. Design and development of processes.
2. Implementing tasks as loosely coupled web services for the processes.
3. Development of service-oriented orchestrations using X-Path queries.
4. Development of parallel loops (⟨while⟩, ⟨repeatUntil⟩ and ⟨forEach⟩), concurrency elements (⟨scope⟩) and synchronization mechanism associated with the processes.

All the analytical and empirical results presented in this paper have been carried out using the data collected from the design related issues of past SOA styled software applications. The data corresponding to the used dataset are provided by an *Indian software organization.* The projects of the used dataset were developed between the years 2009 to 2013. Moreover, the projects corresponding to the used dataset include integrated SOA applications for *universal banking*, *public retail* and *health care solution systems.* The used dataset constitutes 10 different SOA applications with 30 data points.

Variable Name	Variable Description	N	Missing	Mean	Median	Std. Dev.	Min	Max
Actual_Effort (in PH)	Total development effort (PH)	10	0	7 014.91	3 363.47	7 931.52	377.8	22 479.3
No. of Processes	Total number of processes for an application	10	0	2.9	2.5	1.96	1	7
No. of Tasks	Total number of tasks for processes	10	0	6.9	5.5	4.58	2	15
TCC	Total size based on the definition of processes and tasks of an application	10	0	975.7	539	1 027.62	74	2 890
No of partnerLinks	Total number of partnerLink Elements	10	0	11.9	10.5	7.56	3	23
Task Variables	Total number of used input and output variables	10	0	72.6	71.5	38.25	22	130
Event Variables	Total number of receive and reply start events	10	0	6.2	5.5	4.10	2	13
Elements	Total number of variables and message definitions	10	0	14.9	13.5	7.15	7	28
XScript	Total number of X-path queries	10	0	7.4	4.5	6.65	1	20

Table 1. Descriptive statistics of some numerical variables of the used dataset

The analysis of our proposed methodology is aimed at measuring the metrics used as arguments for the generation of regression models. More concise form of the proposed metrics is described in Section 4. We have presented the descriptive statistics of the used dataset in Table 1. The statistical summary is presented only considering *some numerical* variables of the dataset. In Table 1, the variable "TCC" denotes the total code size and "N" signifies total number of projects in the used dataset. The variable Actual_Effort (in PH) denotes the actual effort needed for developing the final application. The descriptive statistics are essential for carrying out the empirical study because it presents the data in more meaningful way and facilitates simpler interpretation of data.

4 PROPOSED WORK

This section illustrates the proposed methodology. After rigorous in-depth study and analysis, the metrics are proposed and presented in the first subsection. The process of generation of regression based estimation models using the proposed metrics is highlighted in the next subsection.

Items	Type	Description
Output	Integrated Application	Developed SOA styled integrated application
	Process and Element Definition	Process, abstract process and flow elements
	Tasks	Processes tasks implemented as web services
	SOA-Orchestration	X-path Queries
	Scripting Languages	Java Script or VB Script
	Message Start Events	Receive or reply events
Software	GUI tools	IBM Web Sphere or BPMN Modeler
People	Designers	Involved in design of processes
	Developers	Persons engaged in development of the application
Technique	Application Design	Exercise carried out in the design of application
	Integration	Exercise carried out for integrating the web application
	Task	Exercise carried out for developing the tasks of processes

Table 2. Initial items for the case study

4.1 Proposed Metrics

The proposed metrics are aimed to measure the different types of items listed in Table 2. For each category of items, there exists set of measuring metrics, that we classified into 4 different categories. They are: environment configuration and re-usability, length and size, effort, and perplexing factors. Each category of items defined in Table 2 plays a vital role in the estimation process. The last categorical variables consisting the perplexing factors also play an influential role in the overall estimation process.

Before moving to the metrics some essential concepts, parameters and cost drivers are recalled in this section. Let us consider the item type "Process and Element Definition" defined in Table 2. In the context of service design phase, for the associated processes, all the composite links to the services using which the process interact are known as the *partnerLink* elements. These elements serve as a reference to the actual implementation, using which the processes interacts with external services. Moreover, the tasks of processes are implemented as a loosely coupled web service which *defines* the participant of web services, and the *properties* of the participant are linked to the partnerLink elements. Furthermore, the *partnerLink elements* are defined "how two individual service partner interact with each other and what each of the partner has to offer". As the partnerLink element is defined and included in each and every service involved in the process design phase, it is considered as an important parameter (cost driver) in context of SOA system development. Similarly, the XML Path Expression (XPath Expression) is used to check the data constraint of the service offered by the client. Generally, XPath queries are available to access the Domain Value Maps (DVMs) which are responsible for SOA orchestration. The SOA *orchestration* allows the work-flow definition between two different services. This is the reason to use X-Path queries as a critical parameter, as it facilitates the SOA system orchestration using the mapping process. Therefore, the *partnerLink elements* and *XPath queries* are included in the SOA system effort estimation as an essential cost driver.

Items	Metrics	Description
Process and Elements Definition	Re-used Process Count	Total number of re-used processes
	Re-used Task Count	Total number of re-used tasks
	Re-used Participant Count	Total number of re-used participants of web services corresponding to the tasks
	Re-used Space allotment Count	Total space (in Kilo Bytes) of the re-used application
	Re-used partnerLink Element	Total number of re-used invoked and client partnerLink elements
Integrated Application	Re-used Code Count	Total lines of code for all the re-used processes

Table 3. Environment configuration and re-usability metrics

Items	Metrics	Description
Process and Elements Definition	Total Process	Number of processes
	Total Abstract Process	Number of abstract processes
	Total partnerLink Elements	Number of invoked and client partnerLink elements
	Total Process Mapping	Number of variables and message definitions
	Total Parallel Flow	Number of links for a process
	Total Code Length	Total lines of code of a process
	Total Participants	Number of participants subjected to type of configuration
	Total Interfaces	Number of interfaces
	Total Scripts	Number of lines of the scripting languages and number of fault handlers
Tasks	Total Tasks	Number of tasks implemented as web services
	Total Operations	Number of designed operations
	Total Variables	Number of used input and output variables
SOA-Orchestration	Orchestration Count	Total number of X-path queries
Message Start Events	Total Receive Events Count	Number of receive start events
	Total Reply Events Count	Number of reply start events
	Total Confirmed and Submit Count	Total number of submit and confirmed events
Integrated Application	Total Code Count	Total lines of code for an application
	Comment Count*	Number of comment lines
	Space Count	Size of application (in Kilo Bytes)

* In our empirical study, the Comment Count metric is not used. This is because the dataset used does not match these requirements.

Table 4. Length and size metrics

Items	Metrics	Description
Process and Elements Definition	Process Effort	Estimated time for designing all the processes, interfaces and abstract processes of an application
Tasks	Task Effort	Estimated time for developing all the implemented tasks (partnerLink elements and parallel activities) of processes
SOA-Orchestration	Orchestration Count	Estimated time for building SOA Orchestration
Message Start Events	Event Effort	Estimated time for designing all types of events of an application
Integrated Application	Total Effort	(Process + Task + Orchestration + Event) Effort

Table 5. Effort metrics

Items	Metrics	Description
People	Skill	Design experience of subject on a scale of 0 to 5
Tool	Type*	Types of tool (GUIs) used in the design process of the application

* In our empirical study, the Type metric is not used. This is because the dataset used does not match these requirements.

Table 6. Perplexing factors

Similarly, the perplexing factors presented in Table 6 are the parameters conceived to have an effect on the estimated (dependent) variable, but were considered in the experimental (independent) variables unlike the confounding factors used in statistics [5]. Additionally, Tables 3 to 6 exemplifies metrics for different categories of items based on the environment configuration, size and effort related constraints. Furthermore, Table 3 presents some re-usable aspect of the packaged solution of SOA systems. The primary focus is on the interaction and dependency among the service groups like composite services which enables the middle-ware and platform technologies [6]. The re-usable metrics focuses only on the parameters that are being re-used[4]. Table 4 imitates the analyzed size and length related metrics. Table 5

[4] The dataset used in this empirical study does not constitute any re-used artifacts. Thus, we have not used the Environment Configuration and Re-usability Metrics, while generating the regression models.

presents different types of effort related metrics. The *total effort* of an application is calculated by *adding* all the calculated effort for the classified items.

4.2 Estimation Methodology

The main aim of any regression model is to analyze the relationship between different variables. This analysis is carried out with the implication of some general purpose regression models through the estimation of the relationship. These regression models are constructed with the help of some appropriate variables. The selection of variables from the proposed metrics is a preliminary activity for carrying out the further process. The general form of the statistic based regression model is defined in Equation (1).

$$y = f(x_1, x_2, \ldots, x_k) \tag{1}$$

where y is the dependent variable and x_1, x_2, ..., x_k are the independent variables.

The empirical and simulation results calculated using the regression models serve the following two purposes:

1. How is the predictor or dependent variables (y) affected with some changes in each of the response variables of x (x_1, x_2, ..., x_k), and
2. to predict the value of y using the values of x.

The data collected from multiple sources of an Indian software organization are used to generate statistic based regression models on the set of proposed metrics. The various techniques included in our proposed work are: simple linear, multiple linear, stepwise and ordinary least square regression models. We generate the estimation models for each category of *items* defined in Table 2. The estimated variables namely (Total Effort) is *computed and summed* with respect to each classified item.

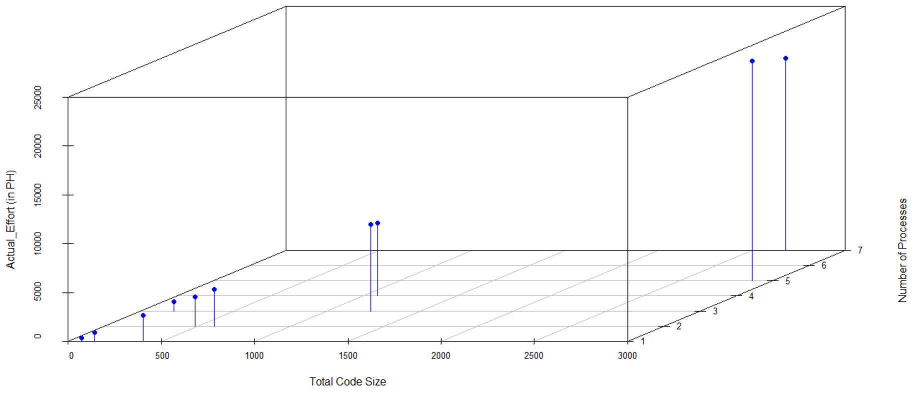


Figure 2. 3D scatter plot for the numerical variables of used dataset

Figure 2 shows the 3D scatter plot plotted for some numerical variables, that are used for generating the regression based estimation models. These numerical variables are: the actual development effort (Actual_Effort (in PH)), the total size (Total Code Size) and the total number of processes (Number of Processes) implemented as services for an individual application. A *Scatter plot* is a mathematical diagram used to represent the displayed data as a collection of points using the value and position of used variables. A scatter plot also depicts a different kind of *correlation* that exists between certain variables with a confidence interval of the dataset. For a scatter plot, if the pattern of dots slopes from lower left corner to the upper right corner, it suggests that a *positive correlation* exists between the set of pair variables. A 3D scatter plot allows better visualization of multivariate data for multiple scalar variables and displays them on the different axes in space. Figure 2 is also useful for discovering the relationship between three variables simultaneously. The plot of Figure 2 suggests that the three variables are positively *correlated* and *associated*, since the variable (Actual_Effort (in PH)) increases linearly.

For generating regression models, we need some dependent and independent variables. The response (dependent) and predictor (independent) variables used for the generation of regression models are listed in Table 7. The variable (Total Effort) corresponding to the item "Integrated Application" is responsible for generating the prediction models. Further, it helps in computing the predicted results through summing all the calculated effort values for different classified items. (See Section 4 for more details).

5 RESULTS AND DISCUSSIONS

The predicted values have been calculated from the generated (simple linear, multiple linear, stepwise and ordinary least square regression) models using R 3.0.2 for Windows. Furthermore, this section discusses the following.

- Calculation of the predicted values using the proposed metrics and from generating 4 regression based estimation models.
- The obtained predicted values are further examined for investigating some statistical properties. These properties included (the linearity, normality and symmetry) and were tested on some commonly used statistical significance tests such as Shapiro-Wilk test, Kolmogorov-Smirnov test, Box-Cox transformation, correlation coefficient (r) and skewness distribution values.
- A comparative analysis of the generated regression models in terms of predictive accuracy is discussed and presented using some commonly used performance measurement indicators and box-plots evaluation.
- Some research threats to validity relevant to our work are also identified and discussed in this section.

For each generated regression model, a set of different plots are presented for assessing the statistical properties of the data variables of the used dataset. These different plots are constructed using some essential statistical artifacts like *residuals*, *fitted values*, *standardized residuals*, *theoretical quantiles*, *leverages*, *scale-location*,

Type	Items	Variables
Response (Dependent) Variables	Process and Elements Definition	Process Effort
	Tasks	Task Effort
	SOA-Orchestration	Orchestration Count
	Message Start Events	Event Effort
	Integrated Application	Total Effort
Predictor (Independent) Variables	Process and Elements Definition	Total Process
		Total Abstract Process
		Total partnerLink Elements
		Total Process Mapping
		Total Parallel Flow
		Total Code length
		Total Participants
		Total Interfaces
		Total Scripts
		Re-used Process Count*
		Re-used Task Count*
		Re-used Participant Count*
		Re-used Space allotment Count*
		Re-used partnerLink Element*
	Tasks	Total Tasks
		Total Operations
		Total Variables
	SOA-Orchestration	Orchestration Count
	Message Start Events	Total Receive Events Count
		Total Reply Events Count
		Total confirmed and submit Count
	Integrated Application	Re-used Code Count*
		Total Code Count
		Space Count

* Note: These predictor variables are useful only for the re-used artifacts.

Table 7. Selection of the variables

standard deviance residuals and *correlation.* Figure 3 flourishes box-plot considering some important numerical variables of the used dataset. The variables used in the box-plot are: the total number of processes, tasks, code size and the actual development effort *versus* the total number of projects (transformed into logarithmic scale) for the used dataset. The variables *Code_Size* and *Actual_Effort* correspond to the total length and effort for all the items and are listed in Table 2. Typically, box-plot is a graphical tool used to check the existence of outliers. Figure 3 depicts that there exist *no outliers* for the used data points of the used dataset.

Thus, there is no need of creating any *new variables* for the set of data variables, as it satisfies the property of *normality* and *linearity.* Additionally, some statistical

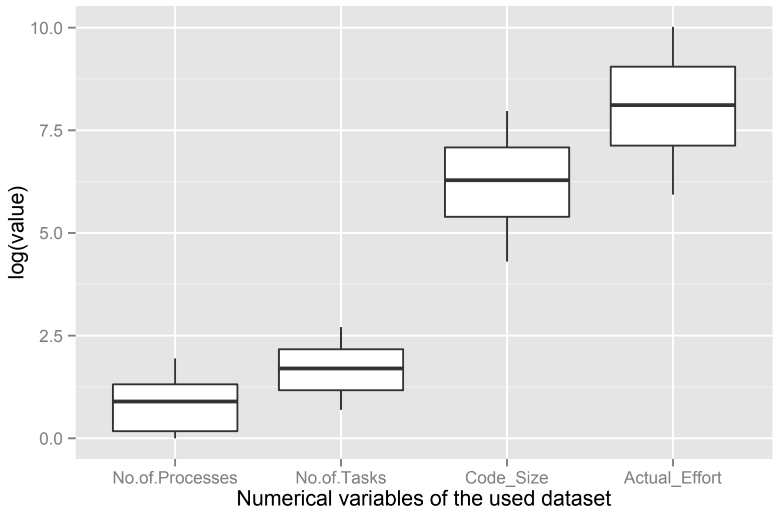


Figure 3. Box-plots for the numerical variables of the used dataset

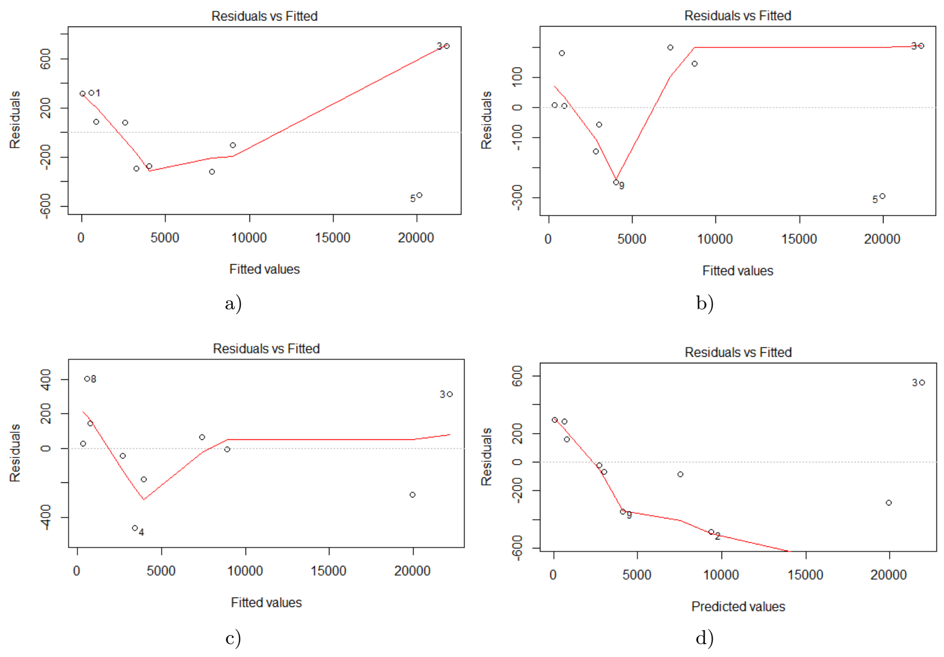


Figure 4. Residual Plot of different generated Regression Models. a) Simple Linear Regression. b) Multiple Linear Regression. c) Step-wise Regression. d) Ordinary Least Square Regression.

tests are performed for scrutinizing the linearity, normality and symmetry property of the used dataset. Figure 4 shows the graphical comparison between *residuals* (actual effort − estimated effort) in the Y-axis and *fitted values* (estimated effort) treated same as the predicted values in X-axis for the generated regression models. The different plots of Figures 4 a), b), c) and d) are known as Residual plot[5]. Figure 4 also indicates that the residuals and predicted values calculated from the generated regression models are not correlated and *equally spread.* Also, there exist no *non-linear* and *non-constant* variances for the used data points.

Additionally, Figure 5 reinforces the normal Quantile-Quantile (Q-Q plot)[6] comparing the randomly generated independent standard normal quantiles data (sample standardized residuals) on the vertical axis and the standard normal population (theoretical quantiles) on the horizontal axis for the generated regression models. Almost, all the points of Q-Q plot lie approximately on a straight line, but not necessarily on the line $y = x$. This is also marked as the condition of *linearity,* in-spite of some points do not lie on the line $y = x$. Moreover, the different plots of Figures 5 a), b), c) and d) are commonly used for scrutinizing the property of *skewness* and *normality.* The simple and multiple linear regression models generated using the proposed metrics for the used dataset yields an adjusted R^2 value as 0.997 and 0.999 respectively. Thus, it indicates 99 % of variation to the used dependent variables (proposed effort metrics). Moreover, none of the projects corresponding to the dataset denotes distance greater than the cook's distance $[3 * (4/10)]$.

After the implication of simple linear and multiple linear regression models, the stepwise regression model has been generated. We have generated the stepwise regression model using both forward and backward procedures as the mode of variable selection. The evaluation criterion for this model is characterized by Akaike Information Criterion (AIC)[7]. Allegedly, AIC provides an efficient mean of model selection because AIC deals with the association between *goodness of fit* and the *complexity* of model [33]. The output values induced by this criterion offer a relative estimate of data loss, when a regression model is used to represent the dependent variables of the used dataset.

Let us consider, a set of regression based candidate models having AIC values as: AIC_1, AIC_2, AIC_3, $\ldots$, AIC_n, respectively. The AIC value for the generated regression model is always chosen from the candidate models having the *minimum* AIC value. In this way, the AIC value provides the *relative estimate* of the data loss. Let AIC_{min} denote minimum AIC values and AIC_i depict other values for the set of candidate models. The expression is interpreted as the relative probability and

[5] The *Residual plot* is a graphical plot commonly used in statistics for showing the relationship between the fitted (estimated) values and residuals.

[6] Q-Q plot is basically a probability plot used for comparing two different probability distributions by plotting the quantiles with each other.

[7] The AIC measures the relative quality of the generated regression model for a given set of data values corresponding to the dataset.

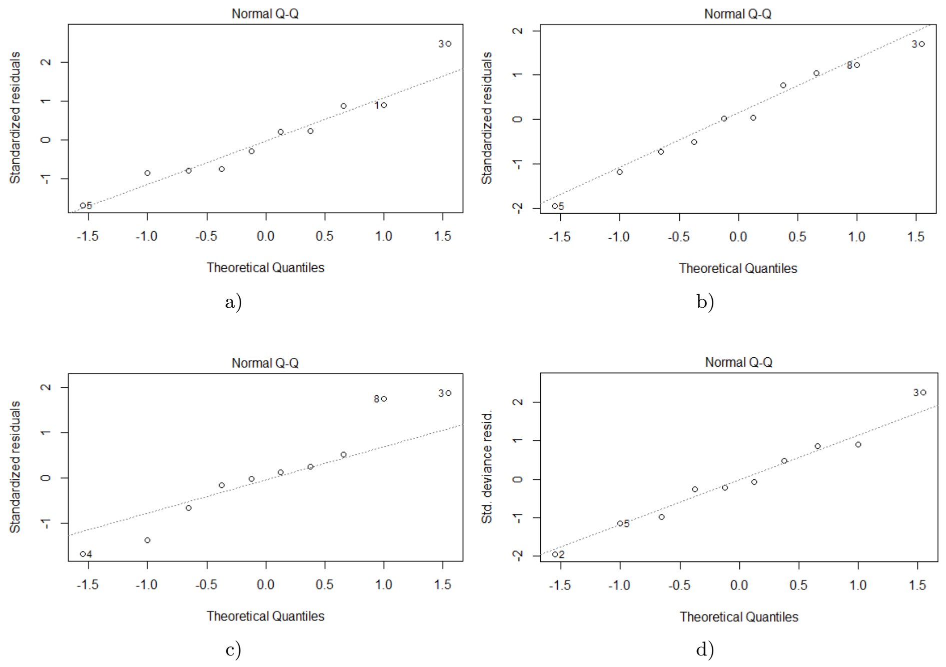


Figure 5. Normal Q-Q plot of different generated Regression Models. a) Simple Linear Regression. b) Multiple Linear Regression. c) Step-wise Regression. d) Ordinary Least Square Regression.

the i^{th} model minimizes the estimated data loss. Equation (2) specifies the relative likelihood of the i^{th} model.

$$e^{(AIC_{min}-AIC_i)/2}. \tag{2}$$

While generating the stepwise regression model, a set of four candidate models are generated having AIC values as $180.52, 121.39, 118.75$ and 118.27, respectively. The chosen AIC value for the stepwise regression model is 118.27. The candidate model having AIC value 118.27 omits all other generated candidate models for *minimizing* the overall data loss. The generated stepwise regression model using the proposed metrics for the used dataset induces adjusted R^2 value as 0.998. It indicates 99 % of variation to the used dependent variables. Again, none of the projects corresponding to the dataset denotes distances greater than the cook's distance for both forward and backward variable selection procedure modes.

Lastly, the Ordinary Least Square (OLS) regression technique have been generated using the proposed metrics. It is a linear approach to the multiple regression technique which results in *eliminating* the error terms. The OLS regression model is a linear approach but many times it works efficiently for the non-linear data. In our work, this regression model have been generated considering the *family type* as "Gaussian". The density function of the Gaussian family is defined in Equation (3).

$$p(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{(y-\mu)^2}{2\sigma^2}\right] \tag{3}$$

where μ is the mean, σ^2 is the variance and y is the response variable. The deduced AIC value for the OLS regression model is computed as 151.

Table 8 shows the values of Multiple R^2, Adjusted R^2 and AIC values, which are calculated using four regression models for all the items defined in Table 2. Typically, R^2 is a statistical value used for determining the goodness of fit of a regression model. R^2 is mostly used for determining the coefficient of the regression model. The AIC value helps to choose the minimum value from a set of candidate models generated for the stepwise and OLS regression techniques. Table 8 illustrates the coefficient values based upon the best criteria[8] for the used regression models. In the OLS regression model, the *dispersion parameter* of Gaussian family, using the density function is computed as 135 740.4.

Regression Models	Multiple R^2	Adjusted R^2	AIC Value
Simple Linear Regression	0.9978	0.9975	–
Multiple Linear Regression	0.9995	0.9999	–
Stepwise Regression	0.9989	0.9984	118.27
Ordinary Least Square (OLS) Regression	–	–	151

Table 8. Coefficient of determination for different generated regression models

5.1 Examining the Statistical Properties

In our work, an effort was made for examining the property of *normality*, *linearity* and *symmetry*. Although the property of *normality* and *linearity* could have been reviewed from the stability, normal Q-Q and residual plot. But for the sake of completeness, a few tests are required to investigate the statistical properties of the computed prediction results. Moreover, we have generated the regression based estimation models following the existing practices and rules [31].

The Shapiro-Wilk test has been introduced to the obtained absolute residual values using the simple and multiple linear regression models, for investigating the property of *normality* and *linearity*. Similarly, box-cox transformation have been employed in the stepwise regression model. In general, the most popular and widely used test for scrutinizing the property of *normality* is Shapiro-Wilk test. Some researchers also used Kolmogorov-Smirnov test as an alternative test for investigating the property of normality [35]. The Kolmogorov-Smirnov test is used to compare

[8] The criterion (AIC) is applicable only for those regression models, which can enable to generate multiple candidate models.

an observed cumulative distribution function (cdf) to an estimated cumulative distribution function. Moreover, the Kolmogorov-Smirnov test is an effective method for comparing the shape of two different cumulative distribution function samples for a small size dataset. For large real-time dataset, the calculated values comprised biases because the sample mean and standard deviation are used to estimate the *population* mean and standard deviation. Thus, Shapiro-Wilk test is presumed to be a *better* approach for testing the property of *normality* over Kolmogorov-Smirnov test. For the generated regression models, the probability-value (p-value) of the absolute residuals are calculated as follows: 0.16, 0.50, 0.33 and 0.92, respectively. Generally, lower the p-value, the *lesser* is the chance of *normality.* Furthermore, many statisticians used p-value 0.05 as the cut-off, the p-value lower than 0.05 depicts that the sample *deviates* from *normality.* For the generated regression models, the absolute residual values are *normally distributed* and satisfy the property of *normality*, as p-value is *greater* than the defined cutoff (> 0.05) value. The *absolute residuals* represent the difference between the actual effort and predicted effort values, and the variable actual effort is used as the dependent variable for generating the regression models. So, based upon this criterion (choosing absolute residual values), it is double checked that the used data and the predicted values obtained using the regression models are normally distributed [34].

We have also investigated the property of *symmetry* for the absolute residual values calculated for the generated regression models. The property of *symmetry* is validated from calculating the *skewness distribution* values. Thus, for the absolute residual values (x) corresponding to the generated regression models, the skewness distribution values are calculated as: 0.62, -0.22, 0.35 and 0.13, respectively. The computed skewness distribution values are skewed both towards right and left. This is because as a rule the *negative skewness* indicates that the mean of absolute residuals for different regression models is less than the median and data distribution is therefore *left-skewed.* Similarly, *positive skewness* indicates that the mean of absolute residuals is larger than the median and data distribution is *right-skewed.* So, for the multiple linear regression model, the data distribution values for the absolute residuals are left-skewed.

Some important guidelines for generating the regression models are "the residual values have not been correlated" and the "used independent variables should not be *linearly dependent* [31]". For validating the property of *linearity*, we have calculated the correlation coefficient (r) between the dependent variables (Actual_Effort (in PH)) and the independent variables (predicted effort values) for each of the generated regression models. If the value of "r" for the two variables is close to 1, then the variables are *linearly positively* related [34]. The calculated correlation coefficient (r) for the different generated regression models are computed as 0.998, 0.999, 0.999, 0.999, respectively. Since the calculated "r" value for all the generated regression models are close to 1. It is concluded that both actual and predicted effort values are *linearly positively* related.

Similarly, for the stepwise regression model the Box-Cox transformation is used for examining the statistical properties. The Box-Cox transformation is best suited

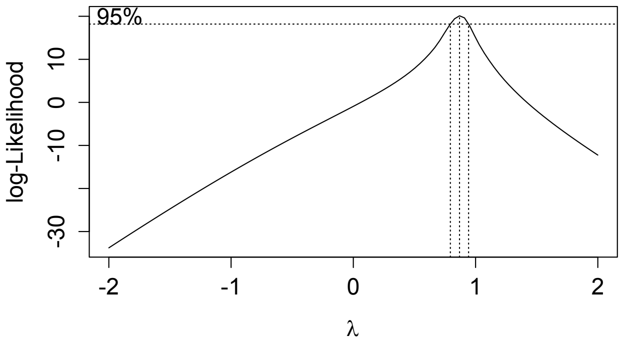


Figure 6. 2D Box-Cox transformation plot for the step-wise regression model

for examining the statistical properties of step-wise regression models and it computes and optimally plots a 2-Dimensional (2D) curve comprising log-likelihood value versus *lambda* (λ). Typically, lambda is the vector values for the chosen parameters. By default, the range of lambda varies within -2 to 2. Figure 6 shows 2D Box-Cox transformation plot for the generated step-wise regression model. In Figure 6, X-axis denotes the vector series (lambda) and the Y-axis represents the log-likelihood values of a particular variable or the parameter for the step-wise regression model. The Box-Cox linearity plot provides an efficient way for finding the suitable transformation mechanism without engaging in a lot of hits and trial fitted models [9]. After generating the step-wise regression model, the Box-Cox transformation have been employed for scrutinizing the statistical properties of the model. The value of lambda (λ) for the generated step-wise regression model is calculated as 0.86. This is an essential data transformation technique used to stabilize the variance and make the data normally distributed, for improving the validity of the associated measures. Table 9 presents the summary of the statistical significant results for the four generated regression based estimation models.

Property	Techniques	Simple Linear Regression	Multiple Linear Regression	Step-Wise Regression	Ordinary Least Square Regression
Normality	Shapiro-Wilk test (p-Value)	0.16	0.50	0.33	0.92
Linearity	Correlation Coefficient (r)	0.998	0.999	0.999	0.999
Symmetry	Skewness Value (s)	0.62	-0.22	0.35	0.13

Table 9. Results of statistical significance tests

5.2 Measuring the Results in Terms of Predictive Accuracy

Each and every used regression model has been iterated for 10 iteration and the average results are computed and presented for calculating the generalization error. Firstly, the used dataset is partitioned into two sets, i.e. the training and testing set. We have used this validation method because the training set of the used dataset have been partitioned randomly (a variant of k-fold cross validation). The training set is defined by $k-1$ samples, and the testing set is defined by k^{th} subset. The process is performed k times and for each iteration and it uses a different project of the used dataset as the testing set[9].

Moreover, we have evaluated the predictive power of the generated regression models using some commonly used performance measurement indicators like Mean Magnitude of Relative Error (MMRE) and Root Mean Square Error (RMSE) [24, 25, 26]. In our work, we have calculated the performance measurement indicator values in terms of *percentage*. This is because for any regression based empirical measurement, there is always a need for combining both the response and predictor variables, for measuring the accuracy. In general, the values of measurement indicators are not *exact*, thus calculating the percentage value allows comparing the predicted (estimated) values to an exact (actual) values. The *percentage value* for any measurement indicator like (MMRE, RMSE) gives the difference between the estimated and exact values in terms of the percentage of exact values. It is helpful in concluding how close the estimated values are with the actual values. The lower values of measurement indicators assure better prediction model. The used performance measurement indicators are described below.

1. **MMRE:** In the era of software effort estimation, MMRE is the most commonly used performance measurement indicators and is used in all types of estimation techniques [25, 26]. The basic metric of MMRE is the Magnitude of Relative Error (MRE) and is defined inside the braces of Equation (4). After calculating MRE, MMRE is obtained from the mean value of MRE.

$$MMRE = \frac{\sum_{i=1}^{n} \left(\frac{|Eact_i - Eest_i|}{Eact_i} \right)}{n} \times 100 \tag{4}$$

   where:

   - $Eest_i$: is the total estimated effort for i number of projects of a dataset.
   - $Eact_i$: is the actual effort for i number of projects of a dataset.
   - n: is the total number of applications or projects of a dataset.

2. **RMSE:** RMSE measures the difference between the estimated value ($Eest_i$) and the actual value ($Eact_i$), for i number of projects of the dataset. In RMSE, the

[9] For conducting the experiment, the value of k is 10, same as the size of the used dataset.

basic metric for computing the error is Mean Square Error (MSE). Taking the square root of MSE yields root mean square error having the same units as the quantity estimated for an unbiased estimator [32]. The RMSE metric is defined in Equation (5).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Eact_i - Eest_i)^2} \times 100. \tag{5}$$

Generated Models	MMRE (in %)	RMSE (in %)
Simple Linear Regression	15.07	3.54
Multiple Linear Regression	**10.80**	**1.75**
Stepwise Regression	14.10	2.47
Ordinary Least Square (OLS) Regression	19.30	3.08

Table 10. Comparison of the generated regression models in terms of predictive accuracy

Table 10 illustrates the comparative results of the four generated regression models in terms of predictive accuracy. The major observation from Table 10 is that the technique of *multiple linear regression* is treated as the *best* estimation model, as it induces lower MMRE and RMSE percentage values, compared to other generated models.

5.3 Comparison of Results Using Boxplots

For any statistical techniques, it is important to investigate the values of *absolute residuals*. An accurate measure is relatively dependent on how much the values of residuals are. Lower values of absolute residuals denote the predicted and actual effort to be similar. Figure 7 shows the box-plot evaluation of the four generated regression models for the values of absolute residual. The box-plot evaluation verifies the results obtained from the performance measurement indicators.

The computed absolute residuals for the generated regression models are shown in the vertical side of each box-plot in Figure 7. Moreover, Figure 7 suggests that the generated *multiple linear regression* model is having the *minimum* values, of absolute residuals when compared against other generated regression models. Intuitively, the box-plots signifies the spread distribution much wider when the absolute residuals are compared with each other for the different generated regression models.

Thus, from the box-plot evaluation and from the implication of performance measurement indicators, it is double-checked that the predicted values obtained using the *multiple linear regression* model furnishes best results in terms of inducing lower residual values for the used dataset. Furthermore, the results (effort values) computed using the multiple linear regression model outperforms all other employed regression models in terms of predictive accuracy.

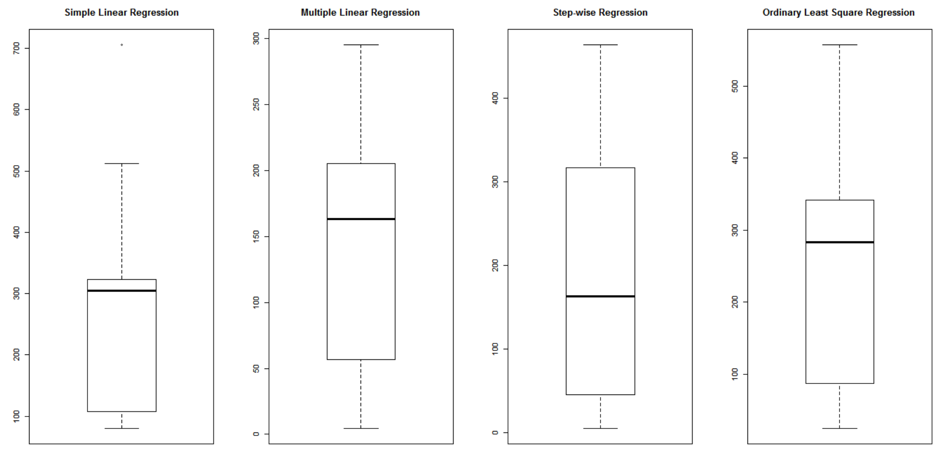


Figure 7. Box-plots of the absolute residuals for different generated regression models

5.4 Threats to the Validity

When conducting an experiment or empirical study, there are always threats to the validity of results. This subsection discusses the validity threats associated with our empirical results on the basis of list of threats by Cook and Campbell [56]. This is conceived as an effective step for concluding the results procured using the statistical methods.

The major factor that may act as an *internal threat* to our simulated results is the ability to draw conclusions about the connections between the chosen independent and dependent variables in the model generation process [56]. This part might also subject to errors and bias. To reduce this threat, manual cross verification of the obtained results was undertaken between two researchers.

Threats to *external validity* associated with the calculated results may be the *size and structure* of the used dataset. However, we conceive that this does not affect the validity of the results, since statistical significant results have been calculated and obtained. Moreover, in the literature of traditional software and web development effort estimation, the prediction results have been calculated using smaller size dataset of 12 and 15 projects respectively [27, 36, 37]. Furthermore, in regard to the external validity, the used dataset overcomes this threat, as the dataset used in our experimental study has been collected from *multiple sources* Indian software organization. On the other hand, we believe that for enhancing more accurate validation, it is essential to collect data from the multiple industrial organization.

Threats to conclusion validity refers to the degree of which the conclusions reached and their relationships using our data are reasonable [56]. To address this threat, the obtained results are examined using some commonly used statistical significance tests. Moreover, the violated assumptions of statistical tests were reduced from the advent of some important test measures for the data variables.

6 CONCLUDING REMARKS

The main aim of this paper was to accord an efficient technique for estimating the SOA systems development effort along with a proper validation. For this, we presented a methodology based on analyzing some initial items associated with the service development life cycle. The measuring metrics are proposed for these identified items. To the best of our knowledge, this was the first time that someone tried to create mapping rules between the service design phase and regression models for generating effort estimation models for SOA systems with their support. More explicitly, none of the previous works used statistics based approach to solve the aforesaid problem along with proper validation for some past project data.

We believe that this approach would definitely add an ease for the readers, analysts and project managers practicing SOA system effort estimation. Our approach of estimating the development effort, builds from the generation of four regression models using the proposed metrics listed in Tables 3, 4, 5, 6. These metrics are proposed based on the different classified parameters like environment configuration, length, size, reused services, effort and perplexing factors for a set of initial items defined in Table 3. Considering the *interest of practitioners,* our proposed technique serves as helpful in dealing with many new complex challenges that project managers encounter with the large size business process SOA systems. In this regard, our proposed metrics goes well beyond the typical capabilities offered by the traditional software estimation techniques.

The use of SOA in developing business process solutions provides better customer services through increased transparency and better consolidation of data and functionality. Some important *statistical properties* have been scrutinized for enhancing the accuracy of the calculated predicted results. The predictive accuracy of the generated regression models has also been demonstrated for some past industrial data using some commonly used performance measurement indicators and box-plots evaluation. The predicted values computed using the multiple linear regression model outperforms every other generated (simple linear, stepwise and ordinary least squares) regression models.

In addition, there is a persuasive need of an efficient effort estimation technique for SOA systems, as the implication of some new features increases the overall complexity of the system. Thus, having an efficient effort estimation technique could contribute in reduction of cost and time implied for developing future SOA systems. As a future work, there is some interesting challenge to perform a replicated study by judging the use of micro services in SOA systems. It will also intrigue to analyze the use of analogy and machine learning approaches in SOA system effort estimation.

REFERENCES

[1] MUKHI, N. K.—KONURU, R.—CURBERA, F.: Cooperative Middleware Specialization for Service-Oriented Architectures. 13th International World Wide Web Conference on Alternate Track Papers and Posters, ACM, New York, USA, 2004, pp. 206–215, doi: 10.1145/1013367.1013401.

[2] ERL, T.: SOA: Principles of Service Design. Prentice Hall, Upper Saddle River, NJ, USA, 2008.

[3] JOSUTTIS, N. M.: SOA in Practice. O'Reilly Media, Inc., Sebastopol, CA, USA, 2007.

[4] VASILIEV, Y.: SOA and WS-BPEL. Packt Publishing Ltd., Birmingham, 2007.

[5] MONTGOMERY, D. C.: Design and Analysis of Experiments. Wiley, New York, 1984.

[6] BELL, M.: Service-Oriented Modeling : Service Analysis, Design, and Architecture, John Wiley and Sons, Inc., Hoboken, New Jersey, 2008.

[7] LOWE, D.—HALL, W.: Hypertext and the Web – An Engineering Approach. John Wiley and Sons, New York, 1998.

[8] MENDES, E.—MOSLEY, N.—COUNSELL, S.: Web Metrics – Estimating Design and Authoring Effort. IEEE MultiMedia, Vol. 8, 2001, No. 1, pp. 50–57, doi: 10.1109/93.923953.

[9] KUTNER, M. H.—NACHTSHEIM, C.—NETER, J.—LI, W.: Applied Linear Statistical Models. McGraw-Hill/Irwin, Blacklick, Ohio, USA, 2005.

[10] LI, Z.—O'BRIEN, L.: A Qualitative Approach to Effort Judgment for Web Service Composition Based SOA Implementations. 25th International Conference on Advanced Information Networking and Applications (AINA), IEEE, Biopolis, Singapore, 2011, pp. 586–593.

[11] LI, Z.—KEUNG, J.: Software Cost Estimation Framework for Service-Oriented Architecture Systems Using Divide-and-Conquer Approach. Fifth IEEE International Symposium on Service Oriented System Engineering (SOSE), IEEE Press, Nanjing, China, 2010, pp. 47–54, doi: 10.1109/SOSE.2010.29.

[12] O'BRIEN, L.: A Framework for Scope, Cost and Effort Estimation for Service-Oriented Architecture (SOA) Projects. Australian Software Engineering Conference (ASWEC '09), IEEE, Gold Coast, Queensland, Australia, 2009, pp. 101–110, doi: 10.1109/ASWEC.2009.35.

[13] LIU, J.—QIAO, J.—LIN, S.—LI, Q.: A Bayesian Net Based Effort Estimation Model for Service Governance Processes. Second International Conference on Information and Computing Science, Manchester, England, UK, IEEE, 2009, pp. 83–86, doi: 10.1109/ICIC.2009.129.

[14] GOMES, Y. M. P.: Functional Size, Effort and Cost of SOA Projects with Function Points. Service Technology Magazine, Issue LXVIII, 2012.

[15] JORGENSEN, M.—SHEPPERD, M.: A Systematic Review of Software Development Cost Estimation Studies. IEEE Transactions on Software Engineering, Vol. 33, 2007, No. 1, pp. 33–53, doi: 10.1109/TSE.2007.256943.

[16] BOEHM, B. W.: Software Engineering Economics. IEEE Transactions on Software Engineering, Vol. SE-10, 1984, No. 1, pp. 4–21.

[17] Heemstra, F. J.: Software Cost Estimation. Information and Software Technology, Vol. 34, 1992, No. 10, pp. 627–639, doi: 10.1016/0950-5849(92)90068-Z.

[18] Hughes, R. T.: Expert Judgement as an Estimating Method. Information and Software Technology, Vol. 38, 1996, No. 2, pp. 67–75, doi: 10.1016/0950-5849(95)01045-9.

[19] Shepperd, M.—Schofield, C.—Kitchenham, B.: Effort Estimation Using Analogy. 18th International Conference on Software Engineering, IEEE, Berlin, Germany, 1996, pp. 170–178, doi: 10.1109/ICSE.1996.493413.

[20] Shepperd, M.—Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering, Vol. 23, 1997, No. 11, pp. 736–743, doi: 10.1109/32.637387.

[21] Li, J.—Ruhe, G.—Al-Emran, A.—Richter, M. M.: A Flexible Method for Software Effort Estimation by Analogy. Empirical Software Engineering, Vol. 12, 2007, No. 1, pp. 65–106.

[22] Nagpal, G.—Uddin, M.—Kaur, A.: Analyzing Software Effort Estimation Using K Means Clustered Regression Approach. ACM SIGSOFT Software Engineering Notes, Vol. 38, 2013, No. 1, pp. 1–9.

[23] Tansey, B.—Stroulia, E.: Valuating Software Service Development: Integrating COCOMO II and Real Options Theory. First International Workshop on the Economics of Software and Computation (ESC ’07), IEEE, Minneapolis, MN, 2007, pp. 8–8.

[24] Conte, S. D.—Dunsmore, H. E.—Shen, V. Y.: Software Engineering Metrics and Models. Benjamin-Cummings Publishing Co. Inc., Redwood City, CA, USA, 1986.

[25] Kitchenham, B. A.—Pickard, L. M.—MacDonell, S. G.—Shepperd, M. J.: What Accuracy Statistics Really Measure: Software Estimation. IEE Proceedings – Software, Vol. 148, 2001, No. 3, pp. 81–85.

[26] Lo, B.—Gao, X.: Assessing Software Cost Estimation Models: Criteria for Accuracy, Consistency and Regression. Australasian Journal of Information Systems, Vol. 5, 1997, No. 1, pp. 30–44.

[27] Di Martino, S.—Ferrucci, F.—Gravino, C.—Mendes, E.: Comparing Size Measures for Predicting Web Application Development Effort: A Case Study. First International Symposium on Empirical Software Engineering and Measurement (ESEM ’07), IEEE, Madrid, Spain, 2007, pp. 324–333, doi: 10.1109/ESEM.2007.20.

[28] Ferrucci, F.—Gravino, C.—Di Martino, S.: A Case Study Using Web Objects and COSMIC for Effort Estimation of Web Applications. 34th Euromicro Conference Software Engineering and Advanced Applications (SEAA ’08), IEEE, Parma, 2008, pp. 441–448, doi: 10.1109/SEAA.2008.60.

[29] Mendes, E.: Cost Estimation Techniques for Web Projects. IGI Global, Hershey, PA, USA, 2007.

[30] Mendes, E.—Mosley, N.—Counsell, S.: Investigating Web Size Metrics for Early Web Cost Estimation. Journal of Systems and Software, Vol. 77, 2005, No. 2, pp. 157–172, doi: 10.1016/j.jss.2004.08.034.

[31] Maxwell, K. D.: Applied Statistics for Software Managers. Prentice-Hall, Software Quality Institute Series, Harlow, United Kingdom, 2005.

[32] Wackerly, D. D.—Mendenhall III, W.—Scheaffer, R. L.: Mathematical Statistics with Applications. Brooks/Cole CENGAGE Learning, Seventh Edition, 2007.

[33] Burnham, K. P.—Anderson, D. R.: Multimodel Inference Understanding AIC and BIC in Model Selection. Sociological Methods and Research, Vol. 33, 2004, No. 2, pp. 261–304, doi: 10.1177/0049124104268644.

[34] Gentle, J. E.: Computational Statistics. Springer, New York, 2009, doi: 10.1007/978-0-387-98144-4.

[35] Abrahão, S.—Gómez, J.—Insfran, E.: Validating a Size Measure for Effort Estimation in Model-Driven Web Development. Information Sciences, Vol. 180, 2010, No. 20, pp. 3932–3954.

[36] Di Martino, S.—Ferrucci, F.—Gravino, C.—Sarro, F.: Using Web Objects for Development Effort Estimation of Web Applications: A Replicated Study. In: Caivano, D., Oivo, M., Baldassarre, M. T., Visaggio, G. (Eds.): Product-Focused Software Process Improvement (PROFES 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6759, 2011, pp. 186–201.

[37] Ruhe, M.—Jeffery, R.—Wieczorek, I.: Using Web Objects for Estimating Software Development Effort for Web Applications. Proceedings of Ninth International Software Metrics Symposium, IEEE, Sydney, NSW, Australia 2003, pp. 30–37, doi: 10.1109/METRIC.2003.1232453.

[38] Lund, K.—Eggen, A.—Hadzic, D.—Hafsoe, T.—Johnsen, F. T.: Using Web Services to Realize Service Oriented Architecture in Military Communication Networks. IEEE Communications Magazine, Vol. 45, 2007, No. 10, pp. 47–53.

[39] Omar, W. M.—Taleb-Bendiab, A.: E-Health Support Services based on Service-Oriented Architecture. IT Professional, Vol. 8, 2006, No. 2, pp. 35–41.

[40] Barker, A.—Weissman, J. B.—Van Hemert, J. I.: Reducing Data Transfer in Service-Oriented Architectures: The Circulate Approach. IEEE Transactions on Services Computing, Vol. 5, 2012, No. 3, pp. 437–449, doi: 10.1109/TSC.2011.23.

[41] Girbea, A.—Suciu, C.—Nechifor, S.—Sisak, F.: Design and Implementation of a Service-Oriented Architecture for the Optimization of Industrial Applications. IEEE Transactions on Industrial Informatics, Vol. 10, 2014, No. 1, pp. 185–196, doi: 10.1109/TII.2013.2253112.

[42] Kim, T.-W.—Kim, H.-C.: Service-Oriented Architecture Structure for Healthcare Systems Utilising Vital Signs. IET Communications, Vol. 6, 2012, No. 18, pp. 3238–3247.

[43] Dietrich, A. J.—Kirn, S.—Sugumaran, V.: A Service-Oriented Architecture for Mass Customization – A Shoe Industry Case Study. IEEE Transactions on Engineering Management, Vol. 54, 2007, No. 1, pp. 190–204, doi: 10.1109/TEM.2006.889076.

[44] Melament, A.—Peres, Y.—Vitkin, E.—Kostirev, I.—Shmueli, N.—Sangiorgi, L.—Mordenti, M.—D'Ascia, S.: BioMIMS – SOA Platform for Research of Rare Hereditary Diseases. Annual SRII Global Conference, IEEE, San Jose, CA, 2011, pp. 83–90, doi: 10.1109/SRII.2011.19.

[45] Xu, L.—Chen, L.—Chen, T.—Gao, Y.: SOA-Based Precision Irrigation Decision Support System. Mathematical and Computer Modelling, Vol. 54, 2011, No. 3-4, pp. 944–949.

[46] Cucinotta, T.—Mancina, A.—Anastasi, G. F.—Lipari, G.—Mangeruca, L.—Checcozzo, R.—Rusina, F.: A Real-Time Service-Oriented Architecture for Industrial Automation. IEEE Transactions on Industrial Informatics, Vol. 5, 2009, No. 3, pp. 267–277, doi: 10.1109/TII.2009.2027013.

[47] Verlaine, B.—Jureta, I. J.—Faulkner, S.: A Requirements-Based Model for Effort Estimation in Service-Oriented Systems. In: Lomuscio, A. R., Nepal, S., Patrizi, F., Benatallah, B., Brandić, I. (Eds.): Service-Oriented Computing – ICSOC 2013 Workshops. Springer International Publishing, Lecture Notes in Computer Science, Vol. 8377, 2014, pp. 82–94.

[48] Boehm, B. W.—Abts, C.—Chulani, S.: Software Development Cost Estimation Approaches – A Survey. Annals of Software Engineering, Vol. 10, 2000, No. 1-4, pp. 177–205.

[49] Demirkan, H.—Kauffman, R. J.—Vayghan, J. A.—Fill, H.-G.—Karagiannis, D.—Maglio, P. P.: Service-Oriented Technology and Management: Perspectives on Research and Practice for the Coming Decade. Electronic Commerce Research and Applications, Vol. 7, 2008, No. 4, pp. 356–376, doi: 10.1016/j.elerap.2008.07.002.

[50] Papazoglou, M. P.—Traverso, P.—Dustdar, S.—Leymann, F.: Service-Oriented Computing: State of the Art and Research Challenges. Computer, Vol. 40, 2007, No. 11, pp. 38–45, doi: 10.1109/MC.2007.400.

[51] Brzoza-Woch, R.—Czekierda, Ł.—Długopolski, J.—Nawrocki, P.—Psiuk, M.—Szydło, T.—Zaborowski, W.—Zieliński, K.—Żmuda, D.: Implementation, Deployment and Governance of SOA Adaptive Systems. In: Ambroszkiewicz, S., Brzeziński, J., Cellary, W., Grzech, A., Zieliński, K. (Eds.): Advanced SOA Tools and Applications. Springer, Berlin, Heidelberg, Studies in Computational Intelligence, Vol. 499, 2014, pp. 261–323.

[52] Papazoglou, M. P.—Traverso, P.—Dustdar, S.—Leymann, F.: Service-Oriented Computing: A Research Roadmap. International Journal of Cooperative Information Systems, Vol. 17, 2008, No. 2, pp. 223–255, doi: 10.1142/S0218843008001816.

[53] Joachim, N.—Beimborn, D.—Weitzel, T.: The Influence of SOA Governance Mechanisms on IT Flexibility and Service Reuse. The Journal of Strategic Information Systems, Vol. 22, 2013, No. 1, pp. 86–101, doi: 10.1016/j.jsis.2012.10.003.

[54] Fiadeiro, J.—Lopes, A.—Abreu, J.: A Formal Model for Service-Oriented Interactions. Science of Computer Programming, Vol. 77, 2012, No. 5, pp. 577–608, doi: 10.1016/j.scico.2011.12.003.

[55] Vassiliadis, B.—Stefani, A.—Tsaknakis, J.—Tsakalidis, A.: From Application Service Provision to Service-Oriented Computing: A Study of the IT Outsourcing Evolution. Telematics and Informatics, Vol. 23, 2006, No. 4, pp. 271–293, doi: 10.1016/j.tele.2005.09.001.

[56] Cook, T. D.—Campbell, D. T.: Quasi-Experimentation: Design and Analysis Issues for Field Settings. Houghton Mifflin, Boston, Massachusetts, United States, 1979.

Siba Mishra received his Bachelor of Technology (B. Tech.) degree in computer science and engineering from Biju Patnaik University of Technology, Rourkela, India in 2009 and the Master of Technology (M. Tech.) degree in computer science and engineering from Kiit University, Bhubaneswar, India in 2012. He is currently working towards his Ph.D. degree in computer science and engineering at the Indian Institute of Technology (Indian School of Mines), Dhanbad, Jharkhand, India. His main research interests include software effort estimation, service-oriented architecture, aspect-oriented programming and program slicing. He is a student member of the IEEE and ACM.

Chiranjeev Kumar is working as Full Professor at the Department of Computer Science and Engineering, Indian Institute of Technology (Indian School of Mines), Dhanbad, Jharkhand, India. He received his Ph.D. degree in computer science and engineering from Allahabad University, India in 2006. He was the gold medalist of his M. Eng. batch at the Department of Computer Science and Engineering, Motilal Nehru National Institute of Technology (MNNIT), Allahabad, Uttar Pradesh, India in 2001. In 1998, he was felicitated upon certified novell administrator (CNA) and certified novell engineer (CNE). In his about 18 years of teaching and research carrier, he has contributed for several research papers in leading refereed journals and conference proceedings of the national and international repute. His main research interests include mobility management in wireless networks, ad hoc networks and software engineering. He is an IEEE member since 2006, and a fellow of the Inventive Research Organization (IRO) since 2016.

Computing and Informatics, Vol. 37, 2018, 581–613, doi: 10.4149/cai_2018_3_581

PERSONALIZING A CONCEPT SIMILARITY MEASURE IN THE DESCRIPTION LOGIC $\mathcal{ELH}$ WITH PREFERENCE PROFILE

Teeradaj Racharak

School of Information, Computer, and Communication Technology
Sirindhorn International Institute of Technology, Thammasat University, Thailand
&
School of Information Science
Japan Advanced Institute of Science and Technology, Japan
e-mail: `r.teeradaj@gmail.com`

Boontawee Suntisrivaraporn

School of Information, Computer, and Communication Technology
Sirindhorn International Institute of Technology, Thammasat University, Thailand
&
Business Intelligence and Data Science, Transformation Group
Siam Commercial Bank Co., Ltd., Thailand
e-mail: `boontawee.suntisrivaraporn@scb.co.th`

Satoshi Tojo

School of Information Science
Japan Advanced Institute of Science and Technology, Japan
e-mail: `tojo@jaist.ac.jp`

Abstract. Concept similarity measure aims at identifying a degree of commonality of two given concepts and is often regarded as a generalization of the classical reasoning problem of equivalence. That is, any two concepts are equivalent if and only if their similarity degree is one. However, existing measures are often devised based on objective factors, e.g. structural-based measures and interpretation-based measures. When these measures are employed to characterize similar concepts in an ontology, they may lead to unintuitive results. In this work, we introduce a new notion called concept similarity measure under preference profile with a set of formally defined properties in Description Logics. This new notion may be interpreted as measuring the similarity of two concepts under subjective factors (e.g. the agent's preferences and domain-dependent knowledge). We also develop a measure of the proposed notion and show that our measure satisfies all desirable properties. Two algorithmic procedures are introduced for top-down and bottom-up implementation, respectively, and their computational complexities are intensively studied. Finally, the paper discusses the usefulness of the approach to potential use cases.

Keywords: Concept similarity measure, semantic web ontology, preference profile, description logics

Mathematics Subject Classification 2010: 68-T30

1 INTRODUCTION

Most Description Logics (DLs) [1] are decidable fragments of First Order Logic (FOL) with clearly defined computational properties. DLs are the logical underpinning of the DL flavor of the ontology languages OWL and OWL 2. The advantage of this close connection is that the extensive DLs literature and implementation experiences can be directly exploited by OWL tools. More specifically, DLs provide unambiguous semantics to the modeling constructs available in OWL DL and OWL 2 DL. These semantics make it possible to formalize and design algorithms for a number of reasoning services, which enable the development of ontology applications to become prominent. For instance, ontology classification (or ontology alignment) organizes concepts in an ontology into a subsumption hierarchy and assists in detecting potential errors of a modeling ontology. Though this subsumption hierarchy inevitably benefits ontology modeling, it merely gives two-valued responses, i.e., inferring a concept is subsumed by another concept or not. However, certain pairs of concepts may share commonality even though they are not subsumed. Thus, a considerable amount of research effort has been devoted on measuring similarity of two given concepts, i.e. *concept similarity measure.*

Basically, a concept similarity measure (abbreviated as CSM) is a function mapping from a concept pair to a unit interval (i.e. $0 \leq x \leq 1$ for any real number x). The higher the value is mapped to, the more likely similarity of them may hold. Intuitively, the value 0 can be interpreted as *total dissimilarity* whereas the value 1 can be interpreted as *total similarity* or *equivalence.* Hence, one may regard CSM as a generalization of the classical reasoning problem of equivalence. It plays a major role in the discovery of similar concepts in an ontology. For example, it is employed in bio-medical ontology-based applications to discover functional similarities of gene [2], it is often used by ontology alignment algorithms [3]. There is currently a significant number of measures in DLs. Prominent examples are [4, 5, 6, 7, 8, 9]. However, these measures are devised based on objective factors. For example, they use the structure (or the interpretation) of concept descriptions to measure. When these measures are employed to characterize similar concepts in an ontology, they may lead to unintuitive results. The following example illustrates that using objective-based measures may not suffice to answer the agent's request.

Example 1. An agent A wants to visit a place for doing some active activities. At that moment, he would like to enjoy walking. Suppose that a place ontology has been modeled as follows:

$$\begin{aligned}
\mathsf{ActivePlace} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk}.\mathsf{Trekking} \sqcap \exists \mathsf{canSail}.\mathsf{Kayaking} \\
\mathsf{Mangrove} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk}.\mathsf{Trekking} \\
\mathsf{Beach} &\sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canSail}.\mathsf{Kayaking} \\
\mathsf{canWalk} &\sqsubseteq \mathsf{canMoveWithLegs} \\
\mathsf{canSail} &\sqsubseteq \mathsf{canTravelWithSails}
\end{aligned}$$

Suppose that a measure used by that Agent A considers merely the objective aspects, it is reasonable to conclude that both Mangrove and Beach are equally similar to the concept ActivePlace. However, by taking into account also the agent's preferences, Mangrove appears more suitable to his perception of ActivePlace at that moment. In other words, he will not be happy if an intelligent system happens to recommend him to go for a Beach.

The example shows that preferences of the agent play a decisive role in the choice of alternatives. In essence, when the choices of an answer are not totally similar to a concept in question, a measure may need to *be tuned* by subjective factors, e.g. the agent's preferences. Another example is shown in [10] on the experiment of the measure sim against SNOMED CT[1], which is one of the largest and the most widely used medical ontologies currently available. It reports that roleGroup and the SNOMED CT top concept SCT-Top can unintentionally increase the degree of similarity. By augmenting that knowledge, the experiment could produce more accurate outputs. The main purpose of this paper is to investigate the use of concept similarity measure under the agent's preferences. As a result, the advantages of our approach are fourfold (cf. Section 4 and Section 5). Firstly, it formalizes the notion of concept similarity measure under the agent's preferences and identifies its desirable properties. Secondly, inspired by the skeptical and credulous measures in [5], when used under different agent's preferences, our theory corresponds to different types of a rational agent, i.e., it has ordering when used by different agents. Thirdly, it presents the similarity measure sim^π with mathematical proofs on the satisfaction

[1] `http://bioportal.bioontology.org/ontologies/SNOMEDCT`

of those properties. Lastly, it presents two algorithmic procedures for implementing the measure, viz. a top-down and a bottom-up versions of the proposed measure.

Our developed measure sim^π is driven by the structural subsumption characterization by means of tree homomorphism. It is worth to mention that Baader proposes this idea in [11, 12] for $\mathcal{ELH}$ w.r.t. an unfoldable TBox, i.e., the subsumption is characterized by means of an existence of a homomorphism in the reverse direction. The notion of homomorphism degree is originally introduced in [13] and employed at the heart of similarity measure for $\mathcal{EL}$. This idea is extended at the heart of *concept similarity measure under the agent's preferences* for $\mathcal{ELH}$. Preliminary studies of this applicability are reported in our proceeding papers [14, 15]. It should be noted that our measure we introduced, i.e. sim^π, may look similar to the measure proposed in [16] in a sense that both are recursive definitions for the same DL $\mathcal{ELH}$; however, they are radically different. These are caused by the distinction of their inspirations and we discuss those points in Section 7. Preliminary, empirical evaluation, and the conclusion are discussed in Section 2, Section 6, and Section 8, respectively.

2 PRELIMINARIES

In this section, we review the basics of the Description Logic $\mathcal{ELH}$ and the problem of concept similarity measure including the measure sim, which is extended to the development of sim^π (originally introduced in [15]).

2.1 Description Logic $\mathcal{ELH}$

We assume countably infinite sets CN of concept names and RN of role names that are fixed and disjoint. The set of concept descriptions, or simply concepts, for a specific DL $\mathcal{L}$ is denoted by $\mathsf{Con}(\mathcal{L})$. The set $\mathsf{Con}(\mathcal{ELH})$ of all $\mathcal{ELH}$ concepts can be inductively defined by the following grammar:

$$\mathsf{Con}(\mathcal{ELH}) ::= A \mid \top \mid C \sqcap D \mid \exists r.C$$

where $\top$ denotes the *top concept*, $A \in \mathsf{CN}$, $r \in \mathsf{RN}$, and $C, D \in \mathsf{Con}(\mathcal{ELH})$. Conventionally, concept names are denoted by A and B, concept descriptions are denoted by C and D, and role names are denoted by r and s, all possibly with subscripts.

A *terminology* or TBox $\mathcal{T}$ is a finite set of (possibly primitive) concept definitions and role hierarchy axioms, whose syntax is an expression of the form $(A \sqsubseteq D)$ $A \equiv D$ and $r \sqsubseteq s$, respectively. The set $\mathsf{CN}^{\mathsf{def}}$ of *defined concept names* are concept names which appear on the left-hand side of a concept definition. Other concepts are called *primitive concept names*, denoted by $\mathsf{CN}^{\mathsf{pri}}$. A TBox $\mathcal{T}$ is called *unfoldable* if all concept definitions are *unique* and *acyclic* definitions. A concept definition A is unique if $\mathcal{T}$ contains at most one concept definition for $A \in \mathsf{CN}^{\mathsf{def}}$ and is acyclic if A is not referred directly or indirectly (via other concept definitions) to itself. For every primitive concept definition $A \sqsubseteq D$ in $\mathcal{T}$, each can be transformed into an equivalent one by introducing a fresh concept name A' via the rule $A \sqsubseteq D \longrightarrow A \equiv A' \sqcap D$. When a TBox $\mathcal{T}$ is unfoldable, concept names can be expanded by exhaustively replacing all defined concept names by their definitions until only primitive concept names remain. Such concept names are called *fully expanded concept names*.[2]

Like primitive definitions, a role hierarchy axiom $r \sqsubseteq s$ can be transformed into a semantically equivalent role definition, by introducing a fresh role name r' via the similar rule $r \sqsubseteq s \longrightarrow r \equiv r' \sqcap s$. Role names occurring on the left-hand side of a role definition are called *defined role names*, collectively denoted by $\mathsf{RN}^{\mathsf{def}}$. All others are called *primitive role names*, collectively denoted by $\mathsf{RN}^{\mathsf{pri}}$. A set of all r's super roles, denoted by $\mathcal{R}_r$, is defined as $\mathcal{R}_r = \{s \in \mathsf{RN} | r \sqsubseteq^* s\}$ and, $r \sqsubseteq^* s$ if $r = s$ or $r_i \sqsubseteq r_{i+1} \in \mathcal{T}$ where $1 \leq i \leq n, r_1 = r, r_n = s$, and $*$ is a transitive closure.

An interpretation $\mathcal{I}$ is a pair $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ where $\Delta^{\mathcal{I}}$ is a non-empty set representing the domain of the interpretation and $\cdot^{\mathcal{I}}$ is an interpretation function which assigns to every concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role name r a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\cdot^{\mathcal{I}}$ is inductively extended to $\mathcal{ELH}$ concepts in the usual manner:

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}; \qquad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}};$$

$$(\exists r.C)^{\mathcal{I}} = \left\{ a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}} : (a,b) \in r^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \right\}.$$

An interpretation $\mathcal{I}$ is said to be a *model* of a TBox $\mathcal{T}$ (in symbols, $\mathcal{I} \models \mathcal{T}$) if it satisfies all axioms in $\mathcal{T}$. $\mathcal{I}$ satisfies axioms $A \sqsubseteq C$, $A \equiv C$, and $r \sqsubseteq s$, respectively, if $A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$, $A^{\mathcal{I}} = C^{\mathcal{I}}$, and $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$. The main inference problem for $\mathcal{ELH}$ is the subsumption problem. That is, given $C, D \in \mathsf{Con}(\mathcal{ELH})$ and a TBox $\mathcal{T}$, C is *subsumed* by D w.r.t. $\mathcal{T}$ (in symbols, $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. Furthermore, C and D are *equivalent* w.r.t. $\mathcal{T}$ (in symbols, $C \equiv_{\mathcal{T}} D$) if $C \sqsubseteq_{\mathcal{T}} D$ and $D \sqsubseteq_{\mathcal{T}} C$. When a TBox $\mathcal{T}$ is empty or is clear from the context, we omit to denote $\mathcal{T}$, i.e. $C \sqsubseteq D$ or $C \equiv D$.

Let $C \in \mathsf{Con}(\mathcal{ELH})$ be a fully expanded concept to the form: $P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.C_1 \sqcap \cdots \sqcap \exists r_n.C_n$, where $P_i \in \mathsf{CN}^{\mathsf{pri}}$, $r_j \in \mathsf{RN}$, $C_j \in \mathsf{Con}(\mathcal{ELH})$ in the same format, $1 \leq i \leq m$, and $1 \leq j \leq n$. The set $P_1, \ldots, P_m$ and the set $\exists r_1.C_1, \ldots, \exists r_n.C_n$ are denoted by $\mathcal{P}_C$ and $\mathcal{E}_C$, respectively, i.e. $\mathcal{P}_C = \{P_1, \ldots, P_m\}$ and $\mathcal{E}_C = \{\exists r_1.C_1, \ldots, \exists r_n.C_n\}$. An $\mathcal{ELH}$ concept description can be structurally transformed into the corresponding $\mathcal{ELH}$ description tree. The root v_0 of the $\mathcal{ELH}$ description tree $\mathcal{T}_C$ has $\{P_1, \ldots, P_m\}$ as its label and has n outgoing edges, each labeled with $\mathcal{R}_{r_j}$ to a vertex v_j for $1 \leq j \leq n$. Then, a subtree with the root v_j is defined recursively relative to the concept C_j. In [11, 12], a characterization of subsumption for the DL $\mathcal{ELH}$ w.r.t. an unfoldable TBox is proposed. Instead of considering concept descriptions, the so-called $\mathcal{ELH}$ description trees corresponding to those concept descriptions are considered. The subsumption is then characterized by an existence of a homomorphism in the reverse direction (cf. Theorem 1).

[2] In this work, we assume that concept names are fully expanded and the TBox can be omitted.

Definition 1 (Homomorphism [11, 12])**.** An $\mathcal{ELH}$ description tree $\mathcal{T}$ is a quintuple (V, E, rt, l, p) where V is a set of vertices, $E \subseteq V \times V$ is a set of edges, rt is the root, $l : V \to 2^{\mathsf{CN}^{\mathsf{pri}}}$ is a vertex labeling function, and $\rho : E \to 2^{\mathsf{RN}}$ is an edge labeling function. Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two $\mathcal{ELH}$ description trees, $v_1 \in V_1$ and $v_2 \in V_2$, there exists a homomorphism h from $\mathcal{T}_1$ to $\mathcal{T}_2$ (written as $h : \mathcal{T}_1 \to \mathcal{T}_2$) iff the following conditions are satisfied:

- $h(rt_1) = rt_2$ and $l_1(v_1) \subseteq l_2(h(v_1))$; and
- for each successor w_1 of v_1 in $\mathcal{T}_1$, $h(w_1)$ is a successor of $h(v_1)$ with $\rho_1(v_1, w_1) \subseteq \rho_2(h(v_1), h(w_1))$.

Example 2. (Continuation of Example 1) Each primitive definition can be transformed to a corresponding equivalent full definition as follows.

$$\mathsf{ActivePlace} \equiv X \sqcap \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking} \sqcap \exists \mathsf{canSail.Kayaking}$$

$$\mathsf{Mangrove} \equiv Y \sqcap \mathsf{Place} \sqcap \exists \mathsf{canWalk.Trekking}$$

$$\mathsf{Beach} \equiv Z \sqcap \mathsf{Place} \sqcap \exists \mathsf{canSail.Kayaking}$$

where X, Y, and Z are fresh primitive concept names. Similarly, $\mathsf{canWalk} \equiv t \sqcap \mathsf{canMoveWithLegs}$ and $\mathsf{canSail} \equiv u \sqcap \mathsf{canTravelWithSails}$, where t and u are fresh primitive role names. In other words, $\mathcal{R}_{\mathsf{canWalk}} = \{t, \mathsf{canMoveWithLegs}\}$ and $\mathcal{R}_{\mathsf{canSail}} = \{u, \mathsf{canTravelWithSails}\}$. Figure 1 depicts $\mathcal{T}_{\mathsf{ActivePlace}}$, as an illustration.

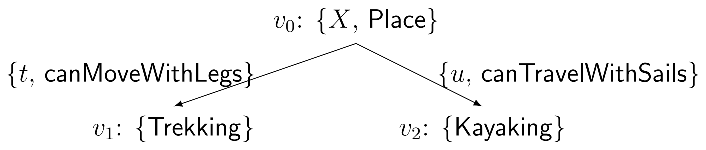


Figure 1. The description tree of concept ActivePlace

Theorem 1 ([11, 12])**.** Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $\mathcal{T}_C$ and $\mathcal{T}_D$ be the corresponding description trees. Then, $C \sqsubseteq D$ iff there exists a homomorphism $h : \mathcal{T}_D \to \mathcal{T}_C$ that maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C$.

From Example 2, it is also not difficult to find a failed attempt of identifying a homomorphism mapping the root of $\mathcal{T}_{\mathsf{ActivePlace}}$ to the root of $\mathcal{T}_{\mathsf{Mangrove}}$, i.e. $h : \mathcal{T}_{\mathsf{ActivePlace}} \not\to \mathcal{T}_{\mathsf{Mangrove}}$. Hence, this infers $\mathsf{Mangrove} \not\sqsubseteq \mathsf{ActivePlace}$ by Theorem 1.

2.2 Concept Similarity Measure in DLs

Concept similarity measure (abbreviated as CSM) is a function mapping from a concept pair to a unit interval ($0 \leq x \leq 1$ where x is a real number). The higher the value is mapped to, the more likely similarity of that concept pair may hold. In the following, we have formally defined the notion of CSM in DLs.

Definition 2. Given two concept descriptions $C, D \in \mathsf{Con}(\mathcal{L})$, a *concept similarity measure* w.r.t. a TBox $\mathcal{T}$ is a function $\sim_{\mathcal{T}} : \mathsf{Con}(\mathcal{L}) \times \mathsf{Con}(\mathcal{L}) \to [0, 1]$ such that $C \sim_{\mathcal{T}} D = 1$ iff $C \equiv_{\mathcal{T}} D$ (*total similarity*) and $C \sim_{\mathcal{T}} D = 0$ indicates *total dissimilarity* between C and D.

When a TBox $\mathcal{T}$ is clear from the context, we simply write $\sim$. Furthermore, to avoid confusion on the symbols, $\sim_{\mathcal{T}}$ is used when referring to arbitrary measures.

The measure sim [13, 10] extends Theorem 1 to the case where no such homomorphism exists but there is some commonality. Since an extension to sim is presented in Subsection 4.1 for taking into account the agent's preferences, the original definitions of homomorphism degree hd and sim are included here for self-containment.

Definition 3 (Homomorphism Degree [10])**.** Let $\mathbf{T}^{\mathcal{ELH}}$ be a set of all $\mathcal{ELH}$ description trees and $\mathcal{T}_C, \mathcal{T}_D \in \mathbf{T}^{\mathcal{ELH}}$ correspond to two $\mathcal{ELH}$ concept names C and D, respectively. The *homomorphism degree* function hd: $\mathbf{T}^{\mathcal{ELH}} \times \mathbf{T}^{\mathcal{ELH}} \to [0, 1]$ is inductively defined as follows:

$$\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = \mu \cdot \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu) \cdot \mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) \tag{1}$$

where $\mu = \frac{|\mathcal{P}_D|}{|\mathcal{P}_D \cup \mathcal{E}_D|}$ and $| \cdot |$ represents the set cardinality;

$$\mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C) = \begin{cases} 1, & \text{if } \mathcal{P}_D = \emptyset, \\ \frac{|\mathcal{P}_D \cap \mathcal{P}_C|}{|\mathcal{P}_D|}, & \text{otherwise,} \end{cases} \tag{2}$$

$$\mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C) = \begin{cases} 1, & \text{if } \mathcal{E}_D = \emptyset, \\ 0, & \text{if } \mathcal{E}_D \neq \emptyset \text{ and } \mathcal{E}_C = \emptyset, \\ \sum_{\epsilon_i \in \mathcal{E}_D} \frac{\max_{\epsilon_j \in \mathcal{E}_C} \{\mathsf{e\text{-}hd}(\epsilon_i, \epsilon_j)\}}{|\mathcal{E}_D|}, & \text{otherwise} \end{cases} \tag{3}$$

with ϵ_i, ϵ_j existential restrictions; and

$$\mathsf{e\text{-}hd}(\exists r.X, \exists s.Y) = \gamma(\nu + (1 - \nu) \cdot \mathsf{hd}(\mathcal{T}_X, \mathcal{T}_Y)) \tag{4}$$

where $\gamma = \frac{|\mathcal{R}_r \cap \mathcal{R}_s|}{|\mathcal{R}_r|}$ and $0 \leq \nu < 1$.

The value of ν in Equation (4) determines how important the roles are to be considered for similarity between two existential restriction information. For instance, $\exists\mathsf{canWalk.Trekking}$ and $\exists\mathsf{canWalk.Parading}$ for dissimilar nested concepts $\mathsf{Trekking}$ and $\mathsf{Parading}$ should not be regarded as entirely dissimilar themselves. If ν is assigned the values 0.3, 0.4, 0.5, then $\mathsf{e\text{-}hd}(\exists\mathsf{canWalk.Trekking}, \exists\mathsf{canWalk.Parading})$ is 0.3, 0.4, 0.5, respectively. This value might vary among applications. In this work, ν is set to 0.4 for exemplifying the calculation of hd.

Theorem 2 ([10])**.** Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $\mathcal{T}_C, \mathcal{T}_D$ be their corresponding description tree, respectively. Then, the following are equivalent:

1. $C \sqsubseteq D$; and
2. $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$.

Using a proof by induction, together with Theorem 1, it is not difficult to observe the correspondence between the homomorphism degree hd and subsumption. Intuitively, Theorem 2 describes a property of concept subsumption, i.e. C is a subconcept of D if the homomorphism degree of the corresponding description tree $\mathcal{T}_D$ to $\mathcal{T}_C$ is equal to 1, and vice versa.

Definition 4 ($\mathcal{ELH}$ Similarity Degree [10])**.** Let C and D be $\mathcal{ELH}$ concept names and $\mathcal{T}_C, \mathcal{T}_D$ be the corresponding description trees. Then, the $\mathcal{ELH}$ *similarity degree* between C and D (in symbols, $\mathsf{sim}(C, D)$) is defined as follows:

$$\mathsf{sim}(C, D) = \frac{\mathsf{hd}(\mathcal{T}_C, \mathcal{T}_D) + \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)}{2}. \tag{5}$$

Example 3. (Continuation of Example 2)
For brevity, let $\mathsf{ActivePlace}$, $\mathsf{Mangrove}$, Beach, Place, $\mathsf{Trekking}$, $\mathsf{Kayaking}$, $\mathsf{canWalk}$, and $\mathsf{canSail}$ be abbreviated as AP, M, B, P, T, K, cW, and cS, respectively. Using Definition 3, the homomorphism degree from $\mathcal{T}_{\mathsf{AP}}$ to $\mathcal{T}_{\mathsf{M}}$, or

$$\begin{aligned}\mathsf{hd}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{M}}) = &\left(\frac{2}{4}\right)\left(\frac{1}{2}\right)\\ &+ \left(\frac{2}{4}\right)\left(\frac{\max\{\mathsf{e\text{-}hd}(\exists \mathsf{cW.T}, \exists \mathsf{cW.T})\}}{2} + \frac{\max\{\mathsf{e\text{-}hd}(\exists \mathsf{cS.K}, \exists \mathsf{cW.T})\}}{2}\right)\\ = &\left(\frac{2}{4}\right)\left(\frac{1}{2}\right) + \left(\frac{2}{4}\right)\left(\frac{1+0}{2}\right) = 0.5.\end{aligned}$$

Similarly, $\mathsf{hd}(\mathcal{T}_{\mathsf{M}}, \mathcal{T}_{\mathsf{AP}}) = 0.67$, $\mathsf{hd}(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{B}}) = 0.5$, and $\mathsf{hd}(\mathcal{T}_{\mathsf{B}}, \mathcal{T}_{\mathsf{AP}}) = 0.67$. Thus, $\mathsf{sim}(\mathsf{M}, \mathsf{AP}) = 0.59$ and $\mathsf{sim}(\mathsf{B}, \mathsf{AP}) = 0.59$

3 PREFERENCE PROFILE

We first introduced *preference profile* (denoted by π) in [14] as a collection of preferential elements in which the development of CSM should be considered. Its first intuition is to model different forms of preferences (of an agent) based on concept names and role names. Measures adopted this notion are flexible to be tuned by an agent and can determine the similarity conformable to that agent's perception.

The syntax and semantics of each form are given in term of partial functions because agents may not have preferences over all concept names and role names. We recommend to devise similarity measures with considerations on preference profile if we aim at developing concept similarity measure for general purposes – a measure based on both subjective and objective factors. Mathematical definitions for each form of preferences are formally defined as follows.

Definition 5 (Primitive Concept Importance)**.** Let $\mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive concept names occurring in $\mathcal{T}$. Then, a *primitive concept importance* is a partial function $\mathfrak{i}^{\mathfrak{c}} : \mathsf{CN} \to [0, 2]^3$, where $\mathsf{CN} \subseteq \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$.

For any $A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{c}}(A) = 1$ captures an expression of normal importance for A, $\mathfrak{i}^{\mathfrak{c}}(A) > 1$ (and $\mathfrak{i}^{\mathfrak{c}}(A) < 1$) indicates that A has higher (and lower, respectively) importance, and $\mathfrak{i}^{\mathfrak{c}}(A) = 0$ indicates that A is of no importance to the agent.

Example 4. (Continuation of Example 2) Suppose that an agent A is using a similarity measure for querying some names similar to $\mathsf{ActivePlace}$. He concerns that those names will be similar to $\mathsf{ActivePlace}$ if they are *places*. Thus, the agent can express this preference as $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 2$, i.e., values should be higher than 1.

On the other hand, suppose he does not care if those are *places* or not, he may express this preference as $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 0$, i.e., values must be equal to 0.

Definition 6 (Role Importance)**.** Let $\mathsf{RN}(\mathcal{T})$ be a set of role names occurring in $\mathcal{T}$. Then, a *role importance* is a partial function $\mathfrak{i}^{\mathfrak{r}} : \mathsf{RN} \to [0, 2]$, where $\mathsf{RN} \subseteq \mathsf{RN}(\mathcal{T})$.

For any $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{i}^{\mathfrak{r}}(r) = 1$ captures an expression of normal importance for r, $\mathfrak{i}^{\mathfrak{r}}(r) > 1$ (and $\mathfrak{i}^{\mathfrak{r}}(r) < 1$) indicates that r has higher (and lower, respectively) importance, and $\mathfrak{i}^{\mathfrak{r}}(r) = 0$ indicates that r is of no importance to the agent.

Example 5 (Continuation of Example 2)**.** Suppose that the agent A wants to enjoy *walking*. He may express this preference as $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 2$, i.e., values should be higher than 1.

Definition 7 (Primitive Concepts Similarity)**.** Let $\mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive concept names occurring in $\mathcal{T}$. For $A, B \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, a *primitive concepts similarity* is a partial function $\mathfrak{s}^{\mathfrak{c}} : \mathsf{CN} \times \mathsf{CN} \to [0, 1]$, where $\mathsf{CN} \subseteq \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, such that $\mathfrak{s}^{\mathfrak{c}}(A, B) = \mathfrak{s}^{\mathfrak{c}}(B, A)$ and $\mathfrak{s}^{\mathfrak{c}}(A, A) = 1$.

For $A, B \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T})$, $\mathfrak{s}^{\mathfrak{c}}(A, B) = 1$ captures an expression of total similarity between A and B and $\mathfrak{s}^{\mathfrak{c}}(A, B) = 0$ captures an expression of their total dissimilarity.

Example 6 (Continuation of Example 2)**.** Suppose that the agent A believes that *trekking* and *kayaking* invoke similar feeling. Thus, he can express $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Trekking}, \mathsf{Kayaking}) = 0.1$, i.e., values should be higher than 0.

[3] In the original definition of preference profile, elements in the domains of both $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{i}^{\mathfrak{r}}$ are mapped to $\mathbb{R}_{\geq 0}$, which is a minor error.

Another example is the similarity of concepts Pet_1 and Pet_2, in which both are defined as follows: $\mathsf{Pet}_1 \sqsubseteq \mathsf{Dog} \sqcap \exists \mathsf{hasOwned.Human}$; $\mathsf{Pet}_2 \sqsubseteq \mathsf{Cat} \sqcap \exists \mathsf{hasOwned.Human}$. Here, Dog and Cat are both primitive concept names. Intuitively, Dog and Cat are similar, then we may attach this knowledge in form of $\mathfrak{s}^{\mathfrak{c}}$ in order to yield more accuracy on the measure.

Definition 8 (Primitive Roles Similarity)**.** Let $\mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$ be a set of primitive role names occurring in $\mathcal{T}$. For $r, s \in \mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$, a *primitive roles similarity* is a partial function $\mathfrak{s}^{\mathfrak{r}} : \mathsf{RN} \times \mathsf{RN} \to [0,1]$, where $\mathsf{RN} \subseteq \mathsf{RN}^{\mathsf{pri}}(\mathcal{T})$, such that $\mathfrak{s}^{\mathfrak{r}}(r,s) = \mathfrak{s}^{\mathfrak{r}}(s,r)$ and $\mathfrak{s}^{\mathfrak{r}}(r,r) = 1$.

For $r, s \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{s}^{\mathfrak{r}}(r,s) = 1$ captures an expression of total similarity between r and s and $\mathfrak{s}^{\mathfrak{r}}(r,s) = 0$ captures an expression of their total dissimilarity.

Example 7 (Continuation of Example 2)**.** Suppose that the agent A believes that *moving with legs* and *traveling with sails* invoke similar feeling. He may express $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}, \mathsf{canTravelWithSails}) = 0.1$, i.e., values should be higher than 0.

Basically, our motivations of both functions $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$ are the same, i.e., we aim at attaching subjective feeling of proximity (about primitive concept names and primitive role names) into a measure. In DLs, different primitive concept names (and also primitive role names) are considered to be total dissimilarity even though they may be recognized as being similar in real-world domains.

Definition 9 (Role Discount Factor)**.** Let $\mathsf{RN}(\mathcal{T})$ be a set of role names occurring in $\mathcal{T}$. Then, a *role discount factor* is a partial function $\mathfrak{d} : \mathsf{RN} \to [0,1]$, where $\mathsf{RN} \subseteq \mathsf{RN}(\mathcal{T})$.

For any $r \in \mathsf{RN}(\mathcal{T})$, $\mathfrak{d}(r) = 1$ captures an expression of total importance on the role (beyond a corresponding nested concept) and $\mathfrak{d}(r) = 0$ captures an expression of total importance on a nested concept (beyond the correspondent role r).

Example 8 (Continuation of Example 2)**.** Suppose that the agent A does not concern much if places permit to either walk or to sail. He would rather consider on actual activities which he can perform. Thus, he may express $\mathfrak{d}(\mathsf{canWalk}) = 0.3$ and $\mathfrak{d}(\mathsf{canSail}) = 0.3$, i.e., values should be close to 0.

Definition 10 (Preference Profile)**.** A *preference profile*, in symbol π, is a quintuple $\langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ where $\mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}$, and $\mathfrak{d}$ are as defined above and the *default preference profile*, in symbol π_0, is the quintuple $\langle \mathfrak{i}^{\mathfrak{c}}_0, \mathfrak{i}^{\mathfrak{r}}_0, \mathfrak{s}^{\mathfrak{c}}_0, \mathfrak{s}^{\mathfrak{r}}_0, \mathfrak{d}_0 \rangle$ where

$$\begin{aligned}
\mathfrak{i}^{\mathfrak{c}}_0(A) &= 1 \text{ for all } A \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}),\\
\mathfrak{i}^{\mathfrak{r}}_0(r) &= 1 \text{ for all } r \in \mathsf{RN}(\mathcal{T}),\\
\mathfrak{s}^{\mathfrak{c}}_0(A,B) &= 0 \text{ for all } (A,B) \in \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}) \times \mathsf{CN}^{\mathsf{pri}}(\mathcal{T}),\\
\mathfrak{s}^{\mathfrak{r}}_0(r,s) &= 0 \text{ for all } (r,s) \in \mathsf{RN}^{\mathsf{pri}}(\mathcal{T}) \times \mathsf{RN}^{\mathsf{pri}}(\mathcal{T}), \text{ and}\\
\mathfrak{d}_0(r) &= 0.4 \text{ for all } r \in \mathsf{RN}(\mathcal{T}).
\end{aligned}$$

Intuitively, the default preference profile π_0 represents the agent's preference in the default manner, i.e., when preferences are not given. That is, every $A \in \mathsf{CN}^{\mathsf{pri}}$ has normal importance and so does every $r \in \mathsf{RN}$. Also, every $(A,B) \in \mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}}$ is totally different and so does every $(r,s) \in \mathsf{RN}^{\mathsf{pri}} \times \mathsf{RN}^{\mathsf{pri}}$. Lastly, every $r \in \mathsf{RN}$ is considered 0.4 importance for the similarity of two existential restriction information. It is interesting to note that changes in the definition of the default preference profile yield different interpretations of the default preference and thereby may produce a different degree of similarity under the default manner. As for its exemplification, the value 0.4 is used by $\mathfrak{d}_0$ to conform with the value of ν used in this work.

In this work, a preference profile of an agent is denoted by subscribing that agent below π, e.g., π_{A} represents a preference profile of the agent A.

4 SIMILARITY MEASURE UNDER PREFERENCE PROFILE

A numerical value determined by CSM indicates a degree of similarity of two concept descriptions w.r.t. the sole objective aspects. That is, either their structures or their interpretations are similar (cf. Section 7). For example, $\mathsf{sim}(\mathsf{ActivePlace}, \mathsf{Mangrove}) = 0.59$ and $\mathsf{sim}(\mathsf{ActivePlace}, \mathsf{Beach}) = 0.59$ indicates that the similarity of ActivePlace and Mangrove, and that of ActivePlace and Beach are equivalently 59 %. However, this information may not be useful for the agent to make decisions.

In this section, we present a conceptual notion for *concept similarity measure under the agent's preferences* (originally introduced in [15]) and its desirable properties. We also present the measure sim^{π} by adopting preference profile onto the measure sim. Our first intuition is to exemplify the applicability of preference profile onto an arbitrary existing measure. This shows that our proposed notion of preference profile can be considered as a collection of noteworthy aspects for the development of concept similarity measure under the agent's preferences.

Definition 11. Given a preference profile π, two concepts $C, D \in \mathsf{Con}(\mathcal{L})$, and a TBox $\mathcal{T}$, a *concept similarity measure under preference profile* w.r.t. a TBox $\mathcal{T}$ is a function $\overset{\pi}{\sim}_{\mathcal{T}}$: $\mathsf{Con}(\mathcal{L}) \times \mathsf{Con}(\mathcal{L}) \to [0,1]$.

When a TBox $\mathcal{T}$ is clear from the context, we simply write $\overset{\pi}{\sim}$. Furthermore, to avoid confusion on the symbols, $\overset{\pi}{\sim}_{\mathcal{T}}$ is used when referring to arbitrary measures.

The notion $\overset{\pi}{\sim}$ may be informally read as *the computation of* $\sim$ *is influenced by* π. That informal interpretation shapes our intuition to consider this kind as a generalization of CSM in DLs. With adopting of this viewpoint of the interpretation, we can agree that sim^π (Subsection 4.1) is informally interpreted as *we compute* sim *(Definition 4) under an existence of a given* π.

Basically, the notion $\overset{\pi}{\sim}$ is a function mapping a pair of two concept descriptions w.r.t. a particular π to a unit interval. We identify a property called *preference invariance w.r.t. equivalence* in our preliminary study [15]. Now, we aim at identifying more important properties of $\overset{\pi}{\sim}$. We start by investigating important properties of CSM existing in the literature (e.g. [16, 9]). Our primary motivation is to identify CSM's properties which are also reasonable for $\overset{\pi}{\sim}$. The following collects fundamental properties for the introduced concept similarity measure under preference profile. They can be used to answer the question *What could be good preference-based similarity measures?* In other words, any preference-based measures satisfying the fundamental properties are considered to be good ones.

Formally, let $C, D, E \in \mathsf{Con}(\mathcal{L})$ and Π be a countably infinite set of preference profile. Then, we call a concept similarity measure under preference profile $\overset{\pi}{\sim}$ is:

1. *symmetric* iff $\forall \pi' \in \Pi : (C \overset{\pi'}{\sim} D = D \overset{\pi'}{\sim} C)$;
2. *equivalence invariant* iff $C \equiv D \Longrightarrow \forall \pi' \in \Pi : (C \overset{\pi'}{\sim} E = D \overset{\pi'}{\sim} E)$;
3. *structurally dependent* iff for any finite sets of concepts C_1 and C_2 with the following conditions:

   - $\mathsf{C}_1 \subseteq \mathsf{C}_2$,
   - concepts $A, B \notin \mathsf{C}_2$,
   - $\mathfrak{i}^{\mathfrak{c}}(\Phi) > 0$ if Φ is primitive and $\Phi \in \mathsf{C}_2$, and
   - $\mathfrak{i}^{\mathfrak{r}}(\varphi) > 0$ if Φ is existential, i.e. $\Phi := \exists \varphi.\Psi$, and $\Phi \in \mathsf{C}_2$,

   the concepts $C := \bigsqcap(\mathsf{C}_1 \cup \{A\})$, $D := \bigsqcap(\mathsf{C}_1 \cup \{B\})$, $E := \bigsqcap(\mathsf{C}_2 \cup \{A\})$ and $F := \bigsqcap(\mathsf{C}_2 \cup \{B\})$ fulfill the condition $\forall \pi' \in \Pi : (C \overset{\pi'}{\sim} D \leq E \overset{\pi'}{\sim} F)$; and
4. *preference invariant w.r.t. equivalence* iff $C \equiv D \Longleftrightarrow \forall \pi' \in \Pi : C \overset{\pi'}{\sim} D = 1$.

Next, we discuss the underlying intuitions of each property subsequently. We note that the properties 1 to 3 are adopted from [16, 9]. However, to the best of our knowledge, the property 4 is first time introduced for *concept similarity measure under preference profile* in this work (originally introduced in [15]).

Let Π be a countably infinite set of preference profile. In the following, we discuss the intuitive interpretation of each property. Firstly, *symmetry* states that an order of concepts in question does not influence the notion $\overset{\pi}{\sim}$ for any $\pi \in \Pi$. For instance, $\mathsf{Mangrove} \overset{\pi}{\sim} \mathsf{Beach} = \mathsf{Beach} \overset{\pi}{\sim} \mathsf{Mangrove}$. This property is controversial as cognitive sciences believes that similarity is asymmetric. However, substantial work in DLs [16, 10, 17, 6, 8, 9, 7, 15, 5] prefers symmetry (merely [4, 18] prefer asymmetry). Here, we also agree on the symmetry because axiomatic information in TBox is not dynamically changed. Furthermore, the notion of preference profile studied in this work is also static, i.e., it can be changed merely by tuning.

Secondly, *equivalence invariance* (alternatively called *equivalence soundness* [9] in the context of dissimilarity measure) states that if two concepts C and D are logically equivalent, then measuring the similarity of each toward the third concept E w.r.t. any $\pi \in \Pi$ must be the same. For instance, let $C \equiv \exists \mathsf{canWalk.Trekking}$ and $D \equiv \exists \mathsf{canWalk.Trekking}$. It is clear that C and D are logically equivalent. Therefore, let $E \in \mathsf{Con}(\mathcal{L})$, $C \overset{\pi}{\sim} E = D \overset{\pi}{\sim} E$ for any $\pi \in \Pi$.

Thirdly, the notion of *structural dependence* is originally introduced by Tversky in [19]. Later, the authors of [16] collect it as another important properties for CSM in their work. Basically, in Tversky's model, an object is considered as a set of features. Then, the similarity of two objects is measured by the relationship between a number of common features and a number of different features. Extending this idea to $\overset{\pi}{\sim}$ gives the meaning that the similarity of two concepts C, D increases if a more number of *equivalent* concepts is shared and each is considered *important*.

Lastly, *preference invariance w.r.t. equivalence* states that if two concepts are logically equivalent, then the similarity degree of two concepts under preference profile π is always one for every $\pi \in \Pi$, and vice versa. Taking the negation both sides, this means $C \not\equiv D \iff \exists \pi \in \Pi : C \overset{\pi}{\sim} D \neq 1$. For instance, let $C \equiv \exists \mathsf{canWalk.Trekking}$ and $D \equiv \exists \mathsf{canWalk.Parading}$. It is clear that C and D are not logically equivalent, then taking $\pi = \pi_0$ obtains $C \overset{\pi_0}{\sim} D \neq 1$; though, taking $\pi = \pi_1$ where $\mathfrak{s}^c(\mathsf{Trekking}, \mathsf{Parading}) = 1$ is defined in π_1 yields $C \overset{\pi_1}{\sim} D = 1$.

There are several properties which are not considered as fundamental properties of concept similarity measure under preference profile because the behaviors may not obey their properties when used under *non-default* preference profiles, e.g. *reverse subsumption preserving*. In [16], a measure $\sim$ satisfies the *reverse subsumption preserving* iff, for any concepts C, D, and E, $C \sqsubseteq D \sqsubseteq E \Longrightarrow C \sim E \leq D \sim E$. The property states that the similarity of D and E is higher than the one of C and E because E is closer to D than C. To refute it, we need only one preference profile π such that the implication does not hold (cf. Example 9), i.e., to show that $(C \sqsubseteq D \sqsubseteq E)$ and $\exists \pi \in \Pi : (C \overset{\pi}{\sim} E > D \overset{\pi}{\sim} E)$.

Example 9. Suppose concepts $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3$, and A_4 are primitive. Query describes features of an item that an agent is searching for. Item_1 and Item_2 are items, which compose of features $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3$ and $\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3, \mathsf{A}_4$, respectively.

$$\mathsf{Query} \sqsubseteq \mathsf{A}_1 \sqcap \mathsf{A}_2$$

$$\mathsf{Item}_1 \sqsubseteq \mathsf{A}_1 \sqcap \mathsf{A}_2 \sqcap \mathsf{A}_3$$

$$\mathsf{Item}_2 \sqsubseteq \mathsf{A}_1 \sqcap \mathsf{A}_2 \sqcap \mathsf{A}_3 \sqcap \mathsf{A}_4$$

The ontology shows the hierarchy: $\mathsf{Item}_2 \sqsubseteq \mathsf{Item}_1 \sqsubseteq \mathsf{Query}$. By taking $\mathfrak{s}^c(\mathsf{A}_2, \mathsf{A}_4) = 1$, it is reasonable to conclude that $\mathsf{Item}_2 \overset{\pi}{\sim} \mathsf{Query} > \mathsf{Item}_1 \overset{\pi}{\sim} \mathsf{Query}$ due to an increased number of totally similar concepts.

Our proceeding paper [5] studies CSM for the DL $\mathcal{FL}_0$. In this paper, we suggest two measures, viz. the skeptical measure $\sim^s$ and the credulous measure $\sim^c$, which are derived from the known structural characterization subsumption through inclusion of regular languages. This fact exhibits that there is not a unique CSM for similarity-based applications. Which CSMs should be used depends on concrete applications, especially the type of a rational agent. For example, when employing the notion $\sim$ to a query answering system, a credulous agent may want to see answers as much as possible; hence, the measure $\sim^c$ is employed. On the other hand, a skeptical agent would like to see sufficient relevant answers; hence, the measure $\sim^s$ is employed. This idea is generalized and is extended toward the notion $\overset{\pi}{\sim}$ to be used under different agent's profiles. In essence, if an arbitrary concept similarity measure under preference profile $\overset{\pi}{\sim}$ is fixed, measuring the similarity of two concepts under different preference profiles may yield different values.

Definition 12. Let Π be a countably infinite set of preference profile and $\pi_1, \pi_2 \in \Pi$. For any fixed measure $\overset{\pi}{\sim}$, the concept similarity measure under π_1 is *more skeptical than* π_2 (denoted by $\overset{\pi_1}{\sim} \preceq \overset{\pi_2}{\sim}$) if $C \overset{\pi_1}{\sim} D \leq C \overset{\pi_2}{\sim} D$ for all $C, D \in \mathsf{Con}(\mathcal{L})$.

4.1 The Measure sim^π

To develop an instance of $\overset{\pi}{\sim}$, we generalize sim for all aspects of preference profile. As a result, the new measure sim^π is also driven by the structural subsumption characterization by means of tree homomorphism for the DL $\mathcal{ELH}$.

We start by presenting each aspect of preference profile in term of *total functions* in order to avoid computing on null values. A *total importance function* is firstly introduced as $\hat{\mathfrak{i}} : \mathsf{CN}^{\mathsf{pri}} \cup \mathsf{RN} \to [0, 2]$ based on the primitive concept importance and the role importance.

$$\hat{\mathfrak{i}}(x) = \begin{cases} \mathfrak{i}^{\mathfrak{c}}(x), & \text{if } x \in \mathsf{CN}^{\mathsf{pri}} \text{and } \mathfrak{i}^{\mathfrak{c}} \text{ is defined on } x, \\ \mathfrak{i}^{\mathfrak{r}}(x), & \text{if } x \in \mathsf{RN} \text{ and } \mathfrak{i}^{\mathfrak{r}} \text{ is defined on } x, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

A *total similarity function* is also presented as $\hat{\mathfrak{s}} : (\mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}}) \cup (\mathsf{RN}^{\mathsf{pri}} \times \mathsf{RN}^{\mathsf{pri}}) \to [0, 1]$ using the primitive concepts similarity and the primitive roles similarity.

$$\hat{\mathfrak{s}}(x, y) = \begin{cases} 1, & \text{if } x = y, \\ \mathfrak{s}^{\mathfrak{c}}(x, y), & \text{if } (x, y) \in \mathsf{CN}^{\mathsf{pri}} \times \mathsf{CN}^{\mathsf{pri}} \text{ and } \mathfrak{s}^{\mathfrak{c}} \text{ is defined on } (x, y), \\ \mathfrak{s}^{\mathfrak{r}}(x, y), & \text{if } (x, y) \in \mathsf{RN}^{\mathsf{pri}} \times \mathsf{RN}^{\mathsf{pri}} \text{ and } \mathfrak{s}^{\mathfrak{r}} \text{ is defined on } (x, y), \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

Similarly, a *total role discount factor function*[4] is presented in the following in term of a function $\hat{\mathfrak{d}} : \mathsf{RN} \to [0,1]$ based on the role discount factor.

$$\hat{\mathfrak{d}}(x) = \begin{cases} \mathfrak{d}(x), & \text{if } \mathfrak{d} \text{ is defined on } x, \\ 0.4, & \text{otherwise.} \end{cases} \tag{8}$$

The next step is to generalize the notion of homomorphism degree hd (Definition 3). Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and $r, s \in \mathsf{RN}$. Also, let $\mathcal{T}_C, \mathcal{T}_D, \mathcal{P}_C$, $\mathcal{P}_D$, $\mathcal{E}_C$, $\mathcal{E}_D$, $\mathcal{R}_r$, and $\mathcal{R}_s$ be as defined in Subsection 2.2. The homomorphism degree under preference profile π from $\mathcal{T}_D$ to $\mathcal{T}_C$ can be formally defined in Definition 13.

Definition 13. Let $\mathbf{T}^{\mathcal{ELH}}$ be a set of all $\mathcal{ELH}$ description trees, and $\pi = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be a preference profile. The *homomorphism degree under preference profile* π is a function $\mathsf{hd}^\pi : \mathbf{T}^{\mathcal{ELH}} \times \mathbf{T}^{\mathcal{ELH}} \to [0,1]$ defined inductively as follows:

$$\mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C) = \mu^\pi(\mathcal{P}_D, \mathcal{E}_D) \cdot \mathsf{p\text{-}hd}^\pi(\mathcal{P}_D, \mathcal{P}_C) + (1 - \mu^\pi(\mathcal{P}_D, \mathcal{E}_D)) \cdot \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_D, \mathcal{E}_C) \tag{9}$$

where

$$\mu^\pi(\mathcal{P}_D, \mathcal{E}_D) = \begin{cases} 1, & \text{if } \sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) + \sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) = 0, \\ \frac{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)}{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) + \sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r)}, & \text{otherwise;} \end{cases} \tag{10}$$

$$\mathsf{p\text{-}hd}^\pi(\mathcal{P}_D, \mathcal{P}_C) = \begin{cases} 1, & \text{if } \sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) = 0, \\ 0, & \text{if } \sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) \neq 0, \\ & \quad \text{and } \sum_{B \in \mathcal{P}_C} \hat{\mathfrak{i}}(B) = 0, \\ \frac{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) \cdot \max_{B \in \mathcal{P}_C} \{\hat{\mathfrak{s}}(A,B)\}}{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)}, & \text{otherwise;} \end{cases} \tag{11}$$

$$\mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_D, \mathcal{E}_C) = \begin{cases} 1, & \text{if } \sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) = 0, \\ 0, & \text{if } \sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) \neq 0 \\ & \quad \text{and } \sum_{\exists s.Y \in \mathcal{E}_C} \hat{\mathfrak{i}}(s) = 0, \\ \frac{\sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r) \cdot \max_{\epsilon_j \in \mathcal{E}_C} \{\mathsf{e\text{-}hd}^\pi(\exists r.X, \epsilon_j)\}}{\sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r)}, & \text{otherwise;} \end{cases} \tag{12}$$

where ϵ_j is an existential restriction; and

$$\mathsf{e\text{-}hd}^\pi(\exists r.X, \exists s.Y) = \gamma^\pi(r,s) \cdot (\hat{\mathfrak{d}}(r) + (1 - \hat{\mathfrak{d}}(r)) \cdot \mathsf{hd}^\pi(\mathcal{T}_X, \mathcal{T}_Y)) \tag{13}$$

[4] We set the default value to 0.4 to comply with the default value of π_0.

where

$$\gamma^{\pi}(r,s) = \begin{cases} 1, & \text{if } \sum_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r') = 0, \\ \frac{\sum_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r') \cdot \max_{s' \in \mathcal{R}_s} \{\hat{\mathfrak{s}}(r',s')\}}{\sum_{r' \in \mathcal{R}_r} \hat{\mathfrak{i}}(r')}, & \text{otherwise.} \end{cases} \tag{14}$$

Intuitively, the function hd^{π} (Equation (9)) is defined as the weighted sum of the degree under preferences of the vertex set commonalities ($\mathsf{p\text{-}hd}^{\pi}$) and the degree under preferences of edge condition matching ($\mathsf{e\text{-}set\text{-}hd}^{\pi}$). Equation (11) calculates the average of the best matching under preferences of primitive concepts in $\mathcal{P}_D$. Equation (13) calculates the degree under preferences of a potential homomorphism of a matching edge. If edge labels share some commonalities under preferences (Equation (14)), i.e. $0 < \gamma^{\pi} \leq 1$, then part of the edge matching is satisfied; but the successors' labels and structures have yet to be checked. This is defined recursively as $\mathsf{hd}^{\pi}(\mathcal{T}_X, \mathcal{T}_Y)$ in Equation (13). Equation (12) calculates the best possible edge matching under preferences of each edge in $\mathcal{E}_D$ and returns the average thereof.

The weight μ^{π} in Equation (9) determines how important the primitive concept names are to be considered for preference-based similarity. For the special case where $D = \top$, i.e. $\mathcal{P}_D = \mathcal{E}_D = \emptyset$, μ^{π} is irrelevant as $\mathcal{T}_{\top}$ is the smallest $\mathcal{ELH}$ description tree and $\mathsf{hd}^{\pi}(\mathcal{T}_{\top}, \mathcal{T}_C) = 1$ for all concepts C.

It is to be mentioned that the function hd^{π} may look similar to $simi_d$ as both are recursive definitions for the same DL $\mathcal{ELH}$. However, they are obviously different caused by the distinction of their inspirations and their viewpoints of the development. While hd^{π} is inspired by the homomorphism-based structural subsumption characterization, $simi_d$ is inspired by the Jaccard Index [20]. Technically speaking, $simi_d$ employs t-conorm instead of fixing an operator. However, unlike $simi_d$, the use of μ^{π} for determining how primitive concepts are weighted and the use of γ^{π} for determining the proportion of shared super roles are employed. Furthermore, $simi_d$ is originated from the viewpoint of CSM, thus some aspects of preference profile are missed; though some may exist. We continue the discussion in Section 7.

The function hd^{π} yields a numerical value that represents structural similarity w.r.t. a particular profile π of a concept against another concept. As both directions constitute the degree of two concepts being equivalent, the measure sim^{π} is also defined by means of these two directional computations.

Definition 14. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathcal{T}_C$ and $\mathcal{T}_D$ be the corresponding description trees, and $\pi = \langle \mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}, \mathfrak{s}^{\mathfrak{c}}, \mathfrak{s}^{\mathfrak{r}}, \mathfrak{d} \rangle$ be a preference profile. Then, the *$\mathcal{ELH}$ similarity measure under preference profile* π between C and D (denoted by $\mathsf{sim}^{\pi}(C,D)$) is defined as follows:

$$\mathsf{sim}^{\pi}(C,D) = \frac{\mathsf{hd}^{\pi}(\mathcal{T}_C, \mathcal{T}_D) + \mathsf{hd}^{\pi}(\mathcal{T}_D, \mathcal{T}_C)}{2}. \tag{15}$$

Intuitively, the degree of similarity under a certain π is the average of the degree of having homomorphism under the same π in both directions. Note that ones may also argue to calculate the value by using alternative binary operators accepting unit intervals, e.g. based on the multiplication (in symbols, $\mathsf{mul\text{-}sim}^\pi$) on both directions of hd^π or the root mean square (in symbols, $\mathsf{rms\text{-}sim}^\pi$) on values of both directions [13]. Unfortunately, those give unsatisfactory values for the extreme cases. For example, $\mathsf{mul\text{-}sim}^\pi(A, \top) = 0 \times 1 = 0$ and $\mathsf{rms\text{-}sim}^\pi(A, \top) = \sqrt{\frac{0^2+1^2}{2}} = 0.707$, whereas $\mathsf{sim}^\pi(A, \top) = \frac{0+1}{2} = 0.5$. Since $\mathsf{mul\text{-}sim}^\pi(C, D) \leq \mathsf{sim}^\pi(C, D) \leq \mathsf{rms\text{-}sim}^\pi(C, D)$ for any concepts C and D, we believe that the average-based definition given above is the most appropriate method. Based on this form, sim^π is basically considered as a generalization of sim which determines similarity under preference profile, i.e., behavioral expectation of the measure will conform to the agent's perception.

We present an example about the calculation of sim^π in the following.

Example 10. (Continuation of Example 1) Let enrich the example. Assume the agent A's preference profile is defined as follows: (i) $\mathfrak{i}^{\mathfrak{c}}(\mathsf{Place}) = 2$; (ii) $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 2$; (iii) $\mathfrak{s}^{\mathfrak{c}}(\mathsf{Trekking}, \mathsf{Kayaking}) = 0.1$; (iv) $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canMoveWithLegs}, \mathsf{canTravelWithSails}) = 0.1$; (v) $\mathfrak{d}(\mathsf{canWalk}) = 0.3$ and $\mathfrak{d}(\mathsf{canSail}) = 0.3$. Let ActivePlace, Mangrove, Beach, Place, Trekking, Kayaking, canWalk, and canSail be rewritten shortly as AP, M, B, P, T, K, cW, and cS, respectively. Using Definition 13,

$$\begin{aligned}
\mathsf{hd}^\pi(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{M}}) &= \left(\frac{3}{6}\right) \cdot \mathsf{p\text{-}hd}^\pi(\mathcal{P}_{\mathsf{AP}}, \mathcal{P}_{\mathsf{M}}) + \left(\frac{3}{6}\right) \cdot \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= \left(\frac{3}{6}\right) \cdot \left(\frac{\mathfrak{i}(X) \cdot \max\{\mathfrak{s}(X, Y), \mathfrak{s}(X, \mathsf{P})\} + \mathfrak{i}(\mathsf{P}) \cdot \max\{\mathfrak{s}(\mathsf{P}, Y), \mathfrak{s}(\mathsf{P}, \mathsf{P})\}}{\mathfrak{i}(X) + \mathfrak{i}(\mathsf{P})}\right) \\
&\quad + \left(\frac{3}{6}\right) \cdot \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= \left(\frac{3}{6}\right) \left(\frac{1 \cdot \max\{0, 0\} + 2 \cdot \max\{0, 1\}}{1 + 2}\right) + \left(\frac{3}{6}\right) \cdot \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_{\mathsf{AP}}, \mathcal{E}_{\mathsf{M}}) \\
&= \left(\frac{3}{6}\right) \left(\frac{2}{3}\right) \\
&\quad + \left(\frac{3}{6}\right) \left[\frac{\mathfrak{i}(\mathsf{cW}) \cdot \max\{\mathsf{e\text{-}hd}^\pi(\exists \mathsf{cW}.\mathsf{T}, \exists \mathsf{cW}.\mathsf{T})\} + 1 \cdot \max\{0.019\}}{\mathfrak{i}(\mathsf{cW}) + \mathfrak{i}(\mathsf{cS})}\right] \\
&= \left(\frac{3}{6}\right) \left(\frac{2}{3}\right) + \left(\frac{3}{6}\right) \left[\frac{2 \cdot \max\{(1)(0.3 + 0.7(1))\} + 1 \cdot \max\{0.019\}}{\mathfrak{i}(\mathsf{cW}) + \mathfrak{i}(\mathsf{cS})}\right] \\
&= \left(\frac{3}{6}\right) \left(\frac{2}{3}\right) + \left(\frac{3}{6}\right) \left[\frac{(2)(1) + (1)(0.019)}{2 + 1}\right] \approx 0.67
\end{aligned}$$

Similarly, we obtain $\mathsf{hd}^\pi(\mathcal{T}_{\mathsf{M}}, \mathcal{T}_{\mathsf{AP}}) = 0.80$. Hence, $\mathsf{sim}^\pi(\mathsf{M}, \mathsf{AP}) \approx 0.74$ by Definition 14. Furthermore, using Definition 13, $\mathsf{hd}^\pi(\mathcal{T}_{\mathsf{AP}}, \mathcal{T}_{\mathsf{B}}) \approx 0.51$ and $\mathsf{hd}^\pi(\mathcal{T}_{\mathsf{B}}, \mathcal{T}_{\mathsf{AP}}) = 0.75$. Hence, $\mathsf{sim}^\pi(\mathsf{B}, \mathsf{AP}) \approx 0.63$ by Definition 14.

The fact that $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP})$ corresponds with the agent A's needs and preferences.

4.2 Properties of sim^{π}

Previously, we theorize a set of desirable properties that a concept similarity measure under preference profile should satisfy and formally introduce the measure sim^{π}. In this subsection, we provide mathematical proofs for the properties of sim^{π}. This gives many benefits to the users of sim^{π} since they can predict its expected behaviors.

Lemma 1. For $\mathcal{T}_D, \mathcal{T}_C \in \mathbf{T}^{\mathcal{ELH}}$, $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$.

Proof. (Sketch) Recall by Definition 10 that the default preference profile π_0 is the quintuple $\langle \mathfrak{i}_0^{\mathfrak{c}}, \mathfrak{i}_0^{\mathfrak{r}}, \mathfrak{s}_0^{\mathfrak{c}}, \mathfrak{s}_0^{\mathfrak{r}}, \mathfrak{d}_0 \rangle$. Also, suppose a concept name D is of the form: $P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n$, where $P_i \in \mathsf{CN}^{\mathsf{pri}}$, $r_j \in \mathsf{CN}$, $D_j \in \mathsf{Con}(\mathcal{ELH})$, $1 \leq i \leq m$, $1 \leq j \leq n$, $P_1 \sqcap \cdots \sqcap P_m$ is denoted by $\mathcal{P}_D$, and $\exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n$ is denoted by $\mathcal{E}_D$. Let d be the depth of $\mathcal{T}_D$. We prove that, for any $d \in \mathbb{N}$, $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ by induction on d.

When $d = 0$, we know that $D = P_1 \sqcap \cdots \sqcap P_m$. To show that $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$, we need to show that $\mu^{\pi_0} = \mu$ and $\mathsf{p\text{-}hd}^{\pi_0}(\mathcal{P}_D, \mathcal{P}_C) = \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C)$. Let us derive as follows:

$$\mu^{\pi_0} = \frac{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A)}{\sum_{A \in \mathcal{P}_D} \hat{\mathfrak{i}}(A) + \sum_{\exists r.X \in \mathcal{E}_D} \hat{\mathfrak{i}}(r)} = \frac{\sum_{i=1}^{m} 1}{\sum_{i=1}^{m} 1 + 0} = \frac{m}{m+0} = \mu.$$

Furthermore, we only need to show $\sum_{A \in \mathcal{P}_D} \max\{\hat{\mathfrak{s}}(A, B) : B \in \mathcal{P}_C\} = |\mathcal{P}_D \cap \mathcal{P}_C|$ in order to show $\mathsf{p\text{-}hd}^{\pi_0}(\mathcal{P}_D, \mathcal{P}_C) = \mathsf{p\text{-}hd}(\mathcal{P}_D, \mathcal{P}_C)$. We know that $\mathfrak{s}_0^{\mathfrak{c}}$ maps name identity to 1 and otherwise to 0. Thus, $\sum_{A \in \mathcal{P}_D} \max\{\hat{\mathfrak{s}}(A, B) : B \in \mathcal{P}_C\} = |\{x : x \in \mathcal{P}_D \text{ and } x \in \mathcal{P}_C\}| = |\mathcal{P}_D \cap \mathcal{P}_C|$.

We must now prove that if $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ holds for $d = h - 1$ where $h > 1$ and $D = P_1 \sqcap \cdots \sqcap P_m \sqcap \exists r_1.D_1 \sqcap \cdots \sqcap \exists r_n.D_n$ then $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$ also holds for $d = h$. To do that, we have to show $\mathsf{e\text{-}set\text{-}hd}^{\pi_0}(\mathcal{E}_D, \mathcal{E}_C) = \mathsf{e\text{-}set\text{-}hd}(\mathcal{E}_D, \mathcal{E}_C)$. This can be done by showing in the similar manner that $\gamma^{\pi_0} = \gamma$ and $\mathsf{hd}^{\pi_0}(\mathcal{T}_X, \mathcal{T}_Y) = \mathsf{hd}(\mathcal{T}_X, \mathcal{T}_Y)$ from $\mathsf{e\text{-}hd}^{\pi_0}(\exists r.X, \exists s.Y) = \mathsf{e\text{-}hd}(\exists r.X, \exists s.Y)$, where $\exists r.X \in \mathcal{E}_D$ and $\exists s.Y \in \mathcal{E}_C$. Consequently, it follows by induction that, for $\mathcal{T}_D, \mathcal{T}_C \in \mathbf{T}^{\mathcal{ELH}}$, $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C)$. □

Theorem 3. Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathsf{sim}^{\pi_0}(C, D) = \mathsf{sim}(C, D)$.

Proof. It immediately follows from Lemma 1, Definition 4, and Definition 14. □

Theorem 3 tells us that sim^{π} is backward compatible in the sense that using sim^{π} with $\pi = \pi_0$, i.e. sim^{π_0}, coincides with sim. Technically speaking, sim^{π_0} can be used to handle the case of similar concepts regardless of the agent's preferences.

Theorem 4. sim^{π} is *symmetric*.

Proof. Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and $C, D \in \mathsf{Con}(\mathcal{ELH})$, we have $\mathsf{sim}^\pi(C,D) = \mathsf{sim}^\pi(D,C)$ by Definition 14. $\square$

Theorem 5. sim^π is *equivalence invariant.*

Proof. Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and $C, D, E \in \mathsf{Con}(\mathcal{ELH})$, we show $C \equiv D \Longrightarrow \mathsf{sim}^\pi(C,E) = \mathsf{sim}^\pi(D,E)$.

Suppose $C \equiv D$, i.e. $C \sqsubseteq D$ and $D \sqsubseteq C$, then we know there exists a homomorphism $h_1 : \mathcal{T}_D \to \mathcal{T}_C$ which maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C$ and $h_2 : \mathcal{T}_C \to \mathcal{T}_D$ which maps the root of $\mathcal{T}_C$ to the root of $\mathcal{T}_D$, respectively, by Theorem 1. This means $\mathcal{T}_C = \mathcal{T}_D$. Thus, $\mathsf{sim}^\pi(C,E) = \mathsf{sim}^\pi(D,E)$. $\square$

Theorem 6. sim^π is *structurally dependent.*

Proof. (Sketch) Let Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$ and any finite sets of concepts C_1 and C_2 with the following conditions:

1. $\mathsf{C}_1 \subseteq \mathsf{C}_2$;
2. concepts $A, B \notin \mathsf{C}_2$;
3. $\mathfrak{i}^{\mathfrak{c}}(\Phi) > 0$ if primitive $\Phi \in \mathsf{C}_2$;
4. $\mathfrak{i}^{\mathfrak{r}}(\varphi) > 0$ if existential $\exists\varphi.\Psi \in \mathsf{C}_2$.

Suppose $C := \bigsqcap(\mathsf{C}_1 \cup \{A\})$, $D := \bigsqcap(\mathsf{C}_1 \cup \{B\})$, $E := \bigsqcap(\mathsf{C}_2 \cup \{A\})$ and $F := \bigsqcap(\mathsf{C}_2 \cup \{B\})$ where $\mathsf{C}_1 = \{P_1, \ldots, P_m, \exists r_1.P'_1, \ldots, \exists r_n.P'_n\}$ and $\mathsf{C}_2 = \{P_1, \ldots, P_i, \exists r_1.P'_1, \ldots, \exists r_j.P'_j\}$, w.l.o.g. we show $\mathsf{sim}^\pi(C,D) \leq \mathsf{sim}^\pi(E,F)$ by following two cases.

Suppose $m \leq i$, $n = j$, and A, B be primitives, we have $\mathsf{p\text{-}hd}^\pi(\mathcal{P}_C, \mathcal{P}_D) = \frac{\sum_{P \in \mathcal{P}_C} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_C} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(A)}$, $\mathsf{p\text{-}hd}^\pi(\mathcal{P}_D, \mathcal{P}_C) = \frac{\sum_{P \in \mathcal{P}_D} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_D} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(B)}$, $\mathsf{p\text{-}hd}^\pi(\mathcal{P}_E, \mathcal{P}_F) = \frac{\sum_{P \in \mathcal{P}_E} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_E} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(A)}$, and $\mathsf{p\text{-}hd}^\pi(\mathcal{P}_F, \mathcal{P}_E) = \frac{\sum_{P \in \mathcal{P}_F} \mathfrak{i}^{\mathfrak{c}}(P)}{\sum_{P \in \mathcal{P}_F} \mathfrak{i}^{\mathfrak{c}}(P) + \mathfrak{i}^{\mathfrak{c}}(B)}$.

Since $m \leq i$, we know $\mathsf{p\text{-}hd}^\pi(\mathcal{P}_C, \mathcal{P}_D) \leq \mathsf{p\text{-}hd}^\pi(\mathcal{P}_E, \mathcal{P}_F)$ and $\mathsf{p\text{-}hd}^\pi(\mathcal{P}_D, \mathcal{P}_C) \leq \mathsf{p\text{-}hd}^\pi(\mathcal{P}_F, \mathcal{P}_E)$. This infers $\mathsf{sim}^\pi(C,D) \leq \mathsf{sim}^\pi(E,F)$.

Suppose $m = i$, $n \leq j$, and A, B be existentials, then with the similar manner, we can show $\mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_C, \mathcal{E}_D) \leq \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_E, \mathcal{E}_F)$ and $\mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_D, \mathcal{E}_C) \leq \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_F, \mathcal{E}_E)$. This also infers $\mathsf{sim}^\pi(C,D) \leq \mathsf{sim}^\pi(E,F)$.

Therefore, we have shown $\mathsf{sim}^\pi(C,D) \leq \mathsf{sim}^\pi(E,F)$. $\square$

Lemma 2. Let $\mathcal{T}_D, \mathcal{T}_C \in \mathbf{T}^{\mathcal{ELH}}$ and Π be a countably infinite set of preference profile. Then, $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longleftrightarrow \forall\pi \in \Pi : \mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C) = 1$.

Proof. Let Π be a countably infinite set of preference profile and π_0 be the default preference profile. Fix any $\pi \in \Pi$, we show $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longleftrightarrow \mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C) = 1$.

($\Rightarrow$) $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$ implies that there exists a homomorphism $h : \mathcal{T}_D \to \mathcal{T}_C$ which maps the root of $\mathcal{T}_D$ to the root of $\mathcal{T}_C$. Consequently, any setting on π does not influence the calculation on $\mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C)$.

($\Leftarrow$) In particular, it suffices to show $\mathsf{hd}^{\pi_0}(\mathcal{T}_D, \mathcal{T}_C) = 1 \Longrightarrow \mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$. By Lemma 1, it is the case that $\mathsf{hd}(\mathcal{T}_D, \mathcal{T}_C) = 1$.

□

Theorem 7. sim^π is *preference invariant w.r.t. equivalence.*

Proof. (Sketch) Let $C, D \in \mathsf{Con}(\mathcal{ELH})$ and Π be a countably infinite set of preference profile. Fix any $\pi \in \Pi$, we show $C \equiv D \Longleftrightarrow \mathsf{sim}^\pi(C, D) = 1$.

($\Rightarrow$) Assume $C \equiv D$, we need to show $\mathsf{sim}^\pi(C, D) = 1$. By Theorem 2, we know $C \equiv D \Longleftrightarrow \mathsf{sim}(C, D) = 1$. With the usage of Lemma 2, Definition 4, and Definition 14, we can derive $\mathsf{sim}^\pi(C, D) = 1$.

($\Leftarrow$) This can be shown similarly as in the forward direction.

□

Theorem 4 to 7 spells out that sim^π satisfies all fundamental properties of concept similarity measure under preference profile.

Another important property of sim^π is that there exists an algorithmic procedure whose execution time is upper bounded by a polynomial expression in the size of the description trees (Theorem 8).

Theorem 8. Assume that a value from any preference functions is retrieved in $\mathcal{O}(1)$. Given $C, D \in \mathsf{Con}(\mathcal{ELH})$, $\mathsf{sim}^\pi(C, D) \in \mathcal{O}(|V_C| \cdot |V_D|)$ where V_C and V_D are set of vertices of the description trees $\mathcal{T}_C$ and $\mathcal{T}_D$, respectively.

Proof. (Sketch) Let $C, D \in \mathsf{Con}(\mathcal{ELH})$, π be any preference profile, and $\mathcal{T}_C, \mathcal{T}_D$ be corresponding description trees. By Definition 14, we show $\mathsf{hd}^\pi(\mathcal{T}_C, \mathcal{T}_D) \in \mathcal{O}(|V_C| \cdot |V_D|)$ and $\mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C) \in \mathcal{O}(|V_D| \cdot |V_C|)$. W.l.o.g. it suffices to show merely hd^π $(\mathcal{T}_C, \mathcal{T}_D) \in \mathcal{O}(|V_C| \cdot |V_D|)$, i.e., we show the computation of each composing part is upper bounded by $|V_C| \cdot |V_D|$. □

Definition 12 suggests that different preference profile settings represent different types of a rational agent. An easy characterization is observed from the aspect of role discount factor ($\mathfrak{d}$). Intuitively, when the settings $\mathfrak{i}^\mathfrak{c}$, $\mathfrak{i}^\mathfrak{r}$, $\mathfrak{s}^\mathfrak{c}$, and $\mathfrak{s}^\mathfrak{r}$ defined by two rational agents A, B are the same, the agent which defines the lower $\mathfrak{d}$ on every $r \in \mathsf{RN}$ is always more skeptical. For instance, if $\mathfrak{d}_\mathrm{A}(\mathsf{canWalk}) = 0.3$ and $\mathfrak{d}_\mathrm{B}(\mathsf{canWalk}) = 0.4$, then $\mathsf{sim}^{\pi_\mathrm{A}}(\exists\mathsf{canWalk.Trekking}, \exists\mathsf{canWalk.Parading}) = 0.3$ and $\mathsf{sim}^{\pi_\mathrm{B}}(\exists\mathsf{canWalk.Trekking}, \exists\mathsf{canWalk.Parading}) = 0.4$. This is clear that the agent A is more skeptical than the agent B.

Proposition 1. Let Π be a countably infinite set of preference profile and $\pi_1, \pi_2 \in \Pi$ such that $\pi_1 = \langle \mathfrak{i}^\mathfrak{c}_1, \mathfrak{i}^\mathfrak{r}_1, \mathfrak{s}^\mathfrak{c}_1, \mathfrak{s}^\mathfrak{r}_1, \mathfrak{d}_1 \rangle$, $\pi_2 = \langle \mathfrak{i}^\mathfrak{c}_2, \mathfrak{i}^\mathfrak{r}_2, \mathfrak{s}^\mathfrak{c}_2, \mathfrak{s}^\mathfrak{r}_2, \mathfrak{d}_2 \rangle$, and RN be a set of role names. The following holds:

$$\forall r \in \mathsf{RN} : (\mathfrak{d}_1(r) \leq \mathfrak{d}_2(r)) \Longrightarrow \equiv \preceq \ \mathsf{sim}^{\pi_1} \ \preceq \ \mathsf{sim}^{\pi_2}$$

for fixed functions $\mathfrak{i}^\mathfrak{c}_1 = \mathfrak{i}^\mathfrak{c}_2$, $\mathfrak{i}^\mathfrak{r}_1 = \mathfrak{i}^\mathfrak{r}_2$, $\mathfrak{s}^\mathfrak{c}_1 = \mathfrak{s}^\mathfrak{c}_2$, and $\mathfrak{s}^\mathfrak{r}_1 = \mathfrak{s}^\mathfrak{r}_2$.

5 IMPLEMENTATION METHODS OF SIM^π

Theorem 8 tells us that sim^π can be computed in the polynomial time. This section exhibits two algorithmic procedures of sim^π belonging to that class.

5.1 Top-Down Implementation of sim^π

Algorithm 1 Pseudo code for hd^π using top-down fashion

```
function $\mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C, \pi)$
    return $(\mu^\pi(\mathcal{T}_D, \pi) \times \mathsf{p\text{-}hd}^\pi(\mathcal{P}_D, \mathcal{P}_C, \pi)) + ((1 - \mu^\pi(\mathcal{T}_D, \pi)) \times \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_D, \mathcal{E}_C, \pi))$
end function

function $\mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_D, \mathcal{E}_C, \pi)$
    if $\sum \mathfrak{i}^{\mathfrak{r}}(\mathcal{E}_D, \pi) = 0$ then
        return 1
    else if $\sum \mathfrak{i}^{\mathfrak{r}}(\mathcal{E}_C, \pi) = 0$ then
        return 0
    else
        $w \leftarrow 0$
        for $\exists r.X \in \mathcal{E}_D$ do
            $m \leftarrow 0$
            for $\exists s.Y \in \mathcal{E}_C$ do
                $e \leftarrow \mathsf{e\text{-}hd}^\pi(\exists r.X, \exists s.Y, \pi)$
                if $e > m$ then
                    $m \leftarrow e$
                end if
            end for
            $w \leftarrow w + (m \times \hat{\mathfrak{i}}(r))$
        end for
        return $w / \sum \mathfrak{i}^{\mathfrak{r}}(\mathcal{P}_D, \pi)$
    end if
end function

function $\mathsf{e\text{-}hd}^\pi(\exists r.X, \exists s.Y, \pi)$
    return $\gamma^\pi(r, s, \pi) \times (\hat{\mathfrak{d}}(r) + ((1 - \hat{\mathfrak{d}}(r)) \times \mathsf{hd}^\pi(\mathcal{T}_X, \mathcal{T}_Y, \pi)))$
end function
```

Algorithm 1 presents the top-down approach for sim^π implementation. Due to the limited space, we omit to show Algorithm 1 in details. The reader may easily observe that the time efficiency of Algorithm 1 is quintic because the computation of $\mathsf{p\text{-}hd}^\pi$ is quadratic and $\mathsf{e\text{-}set\text{-}hd}^\pi$ contains double nested loops which indirectly make recursive calls to hd^{π}. It is also not difficult to observe that the number of recursive calls is upper bounded by the height of the description trees.

It is worth to mention that using hd^{π} requires concept descriptions to be transformed into $\mathcal{ELH}$ description trees. Taking this as an advantage, the next subsection introduces an alternative way to compute hd^{π} from bottom to up, which is approximately three times faster than the counterpart top-down approach in the worst case (cf. Subsection 6.1 for useful discussion).

5.2 Bottom-Up Implementation of sim^{π}

Rather than computing (possibly duplicated) value of hd^{π} again and again, Algorithm 2 employs the classical bottom-up version of dynamic programming technique to compute hd^{π} of the smaller subtrees and records the results in a table (see the variable *result*$[\cdot][\cdot]$ in Algorithm 2) from which a solution to the original computation of hd^{π} can be then obtained (cf. at line No. 20, the function returns value *result*$[0][0]$).

To compute hd^{π} from bottom to up, we need to know the height of the trees in advance. For Algorithm 2, we employ *breath-first search* algorithm (denoted by BFS) to determine the height of each description tree (cf. line No. 4 and 5 of the algorithm). Algorithm 2 reuses the methods μ^{π}, $\mathsf{p\text{-}hd}^{\pi}$, $\mathsf{e\text{-}set\text{-}hd}^{\pi}$, γ^{π}, $\sum \mathfrak{i}^{\mathfrak{c}}$, and $\sum \mathfrak{i}^{\mathfrak{r}}$ from Algorithm 1 and provides pseudo code for $\mathsf{e\text{-}hd}^{\pi}$ since it is merely overridden.

What is the time complexity of Algorithm 2? It should be quintic because the algorithm considers the similarity of all the different pairs of two concept names for h times (cf. line No. 6). More formally, we know *result*$[\mathcal{T}_{\gamma}][\mathcal{T}_{\lambda}] \in \mathcal{O}(v^2)$ where v denotes the set cardinality of $\mathcal{P}_x$ (and $\mathcal{E}_x$) for any description tree x. Let $m(i)$ and $n(i)$ be the number of nodes on level i of description trees D and C, respectively. Then, the number of times operation *result*$[\cdot][\cdot]$ is executed (say C) is equal to:

$$
\begin{aligned}
C &= \sum_{i=0}^{h-1}\sum_{j=0}^{m(i)}\sum_{k=0}^{n(i)} v^2 \\
&= v^2 \sum_{i=0}^{h-1}\sum_{j=0}^{m(i)}\sum_{k=0}^{n(i)} 1 \\
&= v^2 \sum_{i=0}^{h-1}\sum_{j=0}^{m(i)} (n(i)+1) \\
&= v^2 \sum_{i=0}^{h-1} (n(i)+1)(m(i)+1) \\
&= v^2 \big[[(n(0)+1)(m(0)+1)] + [(n(1)+1)(m(1)+1)] \\
&\quad + \ldots + [(n(h-1)+1)(m(h-1)+1)]\big].
\end{aligned}
$$

Algorithm 2 Pseudo code for hd^π using bottom-up fashion

```
1: Initialize a global $result[\cdot][\cdot]$ to store the degree of similarity between 2 concepts.
2:
3: function $\mathsf{hd}^\pi(\mathcal{T}_D, \mathcal{T}_C, \pi)$
4:     Map $< \mathbb{Z}$, List $< \mathcal{T} >>$ $map_D \leftarrow \mathsf{BFS}(\mathcal{T}_D)$    ▷ $map_D$ stores nodes on each level of $\mathcal{T}_D$
5:     Map $< \mathbb{Z}$, List $< \mathcal{T} >>$ $map_C \leftarrow \mathsf{BFS}(\mathcal{T}_C)$    ▷ $map_C$ stores nodes on each level of $\mathcal{T}_C$
6:     $h \leftarrow map_D$.size()
7:     for $i = h - 1$ to 0 do
8:         List $< \mathcal{T} >$ $list_{\mathcal{T}_\Gamma} \leftarrow map_D$.get(i)
9:         List $< \mathcal{T} >$ $list_{\mathcal{T}_\Lambda} \leftarrow map_C$.get(i)
10:        for $\mathcal{T}_\gamma \in list_{\mathcal{T}_\Gamma}$ do
11:            for $list_{\mathcal{T}_\Lambda} \neq$ null and $\mathcal{T}_\lambda \in list_{\mathcal{T}_\Lambda}$ do
12:                if $i = h - 1$ then
13:                    $result[\mathcal{T}_\gamma][\mathcal{T}_\lambda] \leftarrow \mathsf{p\text{-}hd}^\pi(\mathcal{P}_\gamma, \mathcal{P}_\lambda, \pi)$
14:                else
15:                    $result[\mathcal{T}_\gamma][\mathcal{T}_\lambda] \leftarrow (\mu^\pi(\mathcal{T}_\gamma, \pi) \times \mathsf{p\text{-}hd}^\pi(\mathcal{P}_\gamma, \mathcal{P}_\lambda, \pi))$
       $+ ((1 - \mu^\pi(\mathcal{T}_\gamma, \pi)) \times \mathsf{e\text{-}set\text{-}hd}^\pi(\mathcal{E}_\gamma, \mathcal{E}_\lambda, \pi))$
16:                end if
17:            end for
18:        end for
19:    end for
20:    return $result[0][0]$
21: end function
22:
23: function $\mathsf{e\text{-}hd}^\pi(\exists r.X, \exists s.Y, \pi)$
24:     $hd' \leftarrow result[\mathcal{T}_X][\mathcal{T}_Y]$
25:     if $hd' =$ null then
26:         $hd' \leftarrow 0$
27:     end if
28:     return $\gamma^\pi(r, s, \pi) \times (\hat{\mathfrak{d}}(r) + ((1 - \hat{\mathfrak{d}}(r)) \times hd'))$
29: end function
```

Thus, the algorithm makes the similar number of operations as Algorithm 1, plus an additional amount of extra space. On the positive side, the algorithm has never recursively invoked itself to determine the similarity of different pairs of nested concepts, i.e., it directly uses values stored in the table. The algorithm also shows that computing the similarity of nodes from level i, where i is greater than the minimum height of description trees (cf. the condition $list_{\mathcal{T}_\Lambda}! =$ null at line No. 11), is irrelevant to the computation.

Algorithm 2 does work productively in an environment where recursion is fairly expensive. For example, imperative languages, such as Java, C, and Python, are typically faster if using a loop and slower if doing a recursion. On the other hand, for some implementations of functional programming languages, iterations may be very expensive and recursion may be very cheap. In many implementations of them, recursion is transformed into a simple jump but changing the loop variables (which are mutable) requires heavy operations. Subsection 6.1 reports that the practical performance agrees to this theoretical analysis that the bottom-up approach is more efficient when implemented by imperative languages, such as Java.

6 EMPIRICAL EVALUATION

This section evaluates the practical performance of both algorithms against sim[5], reassures pragmatically the backward compatibility of sim^π under π_0 (Theorem 3 already proves this), and discusses the applicability of sim^π in potential use cases.

6.1 Performance Analysis and Backward Compatibility of sim^π

Both versions of sim^π (cf. Subsection 5.1 and Subsection 5.2) are implemented in Java version 1.8 with the usage of Spring Boot version 1.3.3.RELEASE. All the dependencies are managed by Apache Maven version 3.2.5. We also implement unit test cases along with the development of both versions to verify the correctness of their behaviors. In the current state (when we are writing this paper), there are 111 unit test cases. All of them are written to cover important parts of both implementations.

To perform benchmarking, we have selected Snomed ct as a test ontology. As mentioned in the introduction, it is one of the largest and the most widely used medical ontologies currently available, and also, is expressible in $\mathcal{ELH}$. In our experiments, we employ a Snomed ct ontology version from January 2005 (hitherto referred as $\mathcal{O}_{\textsc{Snomed}}$) which contains 379 691 concept names and 62 role names. Moreover, each defined concept is categorized into the 18 mutually exclusive top-level concepts. In the sense of subsumption relation, concepts belonging to the same category should be more similar than those belonging to different categories.

For our experiments, we used a 2.4 GHz Intel Core i5 with 8 GB RAM under OS X El Capitan. Unfortunately, the overall number of concept pairs in $\mathcal{O}_{\textsc{Snomed}}$ is approximately 10^{11}. Suppose an execution of sim^π takes around a millisecond, we still need around 1 158 days in order to complete the entire ontology. According to this reason, we consider 2 out of 18 categories, viz. *Clinical Finding* and *Procedure*, although there are more category pairs. Then, we randomly select 0.5 % of *Clinical Finding*, i.e. 206 concepts, denoted by $\mathbf{C}'_1$. After that, we randomly select the same number of concepts from *Procedure*, i.e. 206 concepts, denoted by $\mathbf{C}'_2$. This sampled set is denoted by $\mathcal{O}'_{\textsc{Snomed}}$, i.e. $\mathcal{O}'_{\textsc{Snomed}} = \mathbf{C}'_1 \cup \mathbf{C}'_2$. Then, we create three test datasets from this sampled set, viz. $\mathbf{C}'_1 \times \mathbf{C}'_1$, $\mathbf{C}'_1 \times \mathbf{C}'_2$, and $\mathbf{C}'_2 \times \mathbf{C}'_2$.

[5] We have re-implemented sim (proposed in [10]) based on the same technologies and techniques as sim^π.

Firstly, we estimate the practical performance of the top-down fashion. For each concept pair in each set, we

1. employ the default preference profile π_0 on (top-down) sim^π;
2. measure the similarity of concepts in $\mathcal{O}'_{\text{SNOMED}}$ by peeking on $\mathcal{O}_{\text{SNOMED}}$ to help unfolding;
3. repeat the previous step with (top-down) sim;
4. repeat steps 2.–3. three times and calculate the statistical results (in milliseconds).

Results are gathered in Table 1. We note that *avg*, *max*, and *min* represent the execution time for measuring similarity of a concept pair in the average case, in the worst case, and in the best case, respectively.

Pairs	Number of Pairs	sim (avg/max/min)	sim^{π_0} (avg/max/min)
$\mathbf{C}'_1 \times \mathbf{C}'_1$	25	2.280/7.000/0.000	1.800/10.000/0.000
$\mathbf{C}'_1 \times \mathbf{C}'_2$	215	2.291/97.000/0.000	2.278/84.000/0.000
$\mathbf{C}'_2 \times \mathbf{C}'_2$	1 849	3.395/45.000/0.000	3.931/128.000/0.000

Table 1. Execution time of top-down sim and top-down sim^{π_0} on $\mathcal{O}'_{\text{SNOMED}}$

Secondly, we estimate the practical performance of the bottom-up fashion by following the same steps as we did previously. Indeed, we exclude the time used to determine the height of each description tree, i.e., our benchmark begins from line No. 7 to 21 of Algorithm 2. Table 2 gathers up the results.

Pairs	Number of Pairs	sim (avg/max/min)	sim^{π_0} (avg/max/min)
$\mathbf{C}'_1 \times \mathbf{C}'_1$	25	2.200/6.000/0.000	1.693/5.000/0.000
$\mathbf{C}'_1 \times \mathbf{C}'_2$	215	2.040/32.000/0.000	1.946/10.000/0.000
$\mathbf{C}'_2 \times \mathbf{C}'_2$	1 849	3.368/55.000/0.000	3.435/45.000/0.000

Table 2. Execution time of bottom-up sim and bottom-up sim^{π_0} on $\mathcal{O}'_{\text{SNOMED}}$

The experiment shows that the practical performance of sim^π is likely equal to the performance obtained by sim – as ones may not expect. The results show that the bottom-up sim^π performs approximately three times faster than the counterpart top-down sim^π (in the worst case) when implemented by imperative languages (e.g. Java as in our case). This conforms to our analysis discussed in Subsection 5.2.

Lastly, we evaluate the backward compatibility of sim^π with sim. Our goal is to ascertain that sim^π can be used interchangeably as the original sim by setting preference profile to the default one (Theorem 3 already proves this). To this point, we have performed an experiment on concept pairs defined in $\mathcal{O}'_{\text{SNOMED}}$. The experiment evaluates results from sim and sim^{π_0} and found that both coincide, as desired.

6.2 Applicability of sim^π

6.2.1 Tuning via $\mathfrak{i}^\mathfrak{c}$ and $\mathfrak{d}$

We show the applicability of $\mathfrak{i}^\mathfrak{c}$ and $\mathfrak{d}$ through similarity measuring on Snomed ct. Figure 2 depicts an example unfoldable terminology extracted from $\mathcal{O}_{\text{Snomed}}$.

$$\begin{aligned}
\mathsf{NeonatalAspirationOfAmnioticFluid} &\equiv \mathsf{NeonatalAspirationSyndromes} \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{causativeAgent}.\mathsf{AmnioticFluid}) \\
\mathsf{NeonatalAspirationOfMucus} &\equiv \mathsf{NeonatalAspirationSyndromes} \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{causativeAgent}.\mathsf{Mucus}) \\
\mathsf{Hypoxemia} &\equiv \mathsf{DisorderOfRespiratorySystem} \sqcap \mathsf{DisorderOfBloodGas} \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{interprets}.\mathsf{OxygenDelivery}) \\
&\quad \sqcap \exists \mathsf{roleGroup}.(\exists \mathsf{findingSite}.\mathsf{ArterialSystemStructure}) \\
\mathsf{BodySecretion} &\sqsubseteq \mathsf{BodySubstance} \\
\mathsf{BodySubstance} &\sqsubseteq \mathsf{Substance} \\
\mathsf{BodyFluid} &\sqsubseteq \mathsf{BodySubstance} \sqcap \mathsf{LiquidSubstance} \\
\mathsf{AmnioticFluid} &\sqsubseteq \mathsf{BodyFluid} \\
\mathsf{Mucus} &\sqsubseteq \mathsf{BodySecretion} \\
\mathsf{causativeAgent} &\sqsubseteq \mathsf{associatedWith}
\end{aligned}$$

Figure 2. Example of $\mathcal{ELH}$ concept definitions defined in $\mathcal{O}_{\text{Snomed}}$

Considering merely objective factors regardless of the agent's preferences, it yields that $\mathsf{sim}^{\pi_0}(\mathsf{NAOAF}, \mathsf{NAOM}) \approx 0.9$[6] and $\mathsf{sim}^{\pi_0}(\mathsf{NAOAF}, \mathsf{H}) = 0.2$. The results yield the quite similar concepts NAOAF and NAOM, which reflect the fact that both are resided in the same cluster of Snomed ct. However, the result yielding that the concepts NAOAF and H share a little similarity controverts the fact that both carry neither implicit nor explicit relationship. This is indeed caused by the usage of the special-purpose role called roleGroup – informally read as *relation group*.

In Snomed ct, the use of relation group is widely accepted to nestedly represent a group of existential information [21]. As a consequence, it increases unintentionally the degree of similarity due to role commonality (i.e. γ^π). Since roleGroup precedes every existential restriction, it is useless to regard an occurrence of this as being similar. The importance contribution of roleGroup in $\mathcal{O}_{\text{Snomed}}$ should be none. Hence, the agent S who measures similarity on Snomed ct should set $\mathfrak{d}_S(\mathsf{roleGroup}) = 0$.

Furthermore, the Snomed ct top concept SCT-TOP subsumes every defined concept of each category. This means this special concept is shared by every expanded concept description. Intuitively, this special top concept is of no importance

[6] Obvious abbreviations are used here for the sake of succinctness.

for measuring similarity on SNOMED CT and we can treat the top-level concepts as directly subsumed by $\top$. As a result, the agent S should also set $\mathfrak{i}^{\mathfrak{c}}_{S}(\mathsf{SCT\text{-}TOP}) = 0$.

Tuning the measure with this expertise knowledge yields more realistic result. That is, the similarity of concepts under the same category which uses roleGroup in their definitions is slightly reduced. Also, the similarity of concepts under different categories is totally dissimilar. Continuing the case, $\mathsf{sim}^{\pi_S}(\mathsf{NAOAF}, \mathsf{NAOM}) \approx 0.84$ and $\mathsf{sim}^{\pi_S}(\mathsf{NAOAF}, \mathsf{H}) = 0.0$, as desired.

6.2.2 Tuning via $\mathfrak{s}^{\mathfrak{r}}$

Let us use the ontology given below to query for places similar to ActivePlace.

$$\mathsf{ActivePlace} \sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canSail}.\mathsf{Kayaking}$$

$$\mathsf{Mangrove} \sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canWalk}.\mathsf{Trekking}$$

$$\mathsf{Supermarket} \sqsubseteq \mathsf{Place} \sqcap \exists \mathsf{canBuy}.\mathsf{FreshFood}$$

Suppose the agent feels *walking* and *sailing* are similar and are *still satisfied much* on both actions. Taking $\mathfrak{s}^{\mathfrak{r}}(\mathsf{canWalk}, \mathsf{canSail}) = 0.6$ yields $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{S}, \mathsf{AP})$, which conforms to the agent's preferences and needs.

6.2.3 Tuning via $\mathfrak{s}^{\mathfrak{c}}$

Let us use the ontology given below to query for a product which offers features the agent is satisfied with most.

$$\mathsf{WantedFeatures} \sqsubseteq \mathsf{F}_0 \sqcap \mathsf{F}_1 \sqcap \mathsf{F}_2$$

$$\mathsf{Item}_1 \sqsubseteq \mathsf{F}_0 \sqcap \mathsf{F}_3$$

$$\mathsf{Item}_2 \sqsubseteq \mathsf{F}_0 \sqcap \mathsf{F}_4$$

According to the ontology, WantedFeatures represents a collection of desired features and F_i (where $i \in \mathbb{N}$) represents a feature. A purchase decision is sometimes affected by satisfied alternations, which are varied by different people. Assume that the agent feels satisfaction to have F_3 if the agent cannot have F_1. Taking $\mathfrak{s}^{\mathfrak{c}}(\mathsf{F}_1, \mathsf{F}_3) = 0.8$ yields $\mathsf{sim}^{\pi}(\mathsf{WF}, \mathsf{I1}) > \mathsf{sim}^{\pi}(\mathsf{WF}, \mathsf{I2})$, which conforms to the agent's perceptions.

6.2.4 Tuning via $\mathfrak{i}^{\mathfrak{r}}$

Let us use the ontology given in Example 1 to query for places which are most similar to ActivePlace. Typically, a human decision is affected by a priority of concerns, which are varied by different people. Suppose that the agent weights more on places which permit to *walk* more than other activities. Taking $\mathfrak{i}^{\mathfrak{r}}(\mathsf{canWalk}) = 2$ yields $\mathsf{sim}^{\pi}(\mathsf{M}, \mathsf{AP}) > \mathsf{sim}^{\pi}(\mathsf{B}, \mathsf{AP})$, which conforms to the agent's preferences.

7 RELATED WORK

As we develop the notion $\overset{\pi}{\sim}_{\mathcal{T}}$ as a generalization of $\sim_{\mathcal{T}}$, this section relates our development to others in two areas, viz. CSMs without regard to the agent's preferences and CSMs with regard to the agent's preferences.

7.1 CSMs Without Regard to The Agent's Preferences

In the standard perception, CSM refers to the study of similar concepts inherited by nature, i.e. the ones similar regardless of the agent's preferences. CSM is widely studied and the techniques are roughly classified into two main groups, viz. path-distance-based approach and DLs-based approach.

In the path-distance-based approach, a degree of similarity is calculated based on the depth of a subsumption hierarchy. The method [22, 23] considers the distance between concepts w.r.t. their least common subsumer. A potential drawback of this approach is its ignorance on concept definitions defined in TBox. Hence, any pair of concepts out of the subsumption relation is always considered as totally dissimilar.

In DLs-based approach, a simple approach is developed in [20] for the DL $\mathcal{L}_0$ (i.e. no use of roles) and is known as Jaccard Index. Its extension to the DL $\mathcal{ELH}$ is proposed in [16]. This work also introduces important properties of CSM and suggests a general framework called *simi* which satisfied most of the properties. In *simi*, functions and operators, such as t-conorm and the fuzzy connector, are to be parameterized and thus left to be specified. The framework also does not contain implementation details. This may cause implementation difficulties since merely promising properties are given and no guideline of how concrete operators are chosen is provided. Similar approaches can be found in [4, 5, 6, 7] for other DLs.

There is another approach which considers their canonical interpretation of concepts in question, such as [8, 9]. A potential drawback of these approaches is that it cannot be applied to an ontology without ABox, e.g. Snomed ct.

The notion of homomorphism degree is originally introduced in [13] and is thereof extended toward the development of sim^{π} in this work. Theorem 3 suggests that sim^{π} can be used to measure similarity of concepts inherently by nature through the setting π_0, i.e. sim^{π_0}. As inspired by the tree homomorphism, the measure differs [16] from the use of μ^{π} to determine how important the primitive concepts are to be considered and the use of γ^{π} to determine a degree of role commonality between matching edges of the description trees.

7.2 CSMs with Regard to The Agent's Preferences

Most CSMs are objective-based. However, there exists work [10, 16] which provides methodologies for tuning. We discuss their differences to ours in the following.

In an extended work of sim [10], a range of number for discount factor (ν) and the neglect of special concept names are used in the similarity application of Snomed

CT. For instance, when $\mathsf{roleGroup}$ is found, the value of ν is set to 0. These ad hoc approaches can be viewed as specific applications of $\mathfrak{d}$ and $\mathfrak{i}^{\mathfrak{c}}$, respectively, of preference profile. Unfortunately, no other aspects of π appear in its use.

In *simi* [16], the function *pm* is used to define the similarity degree of primitive concept pairs and role pairs. Using *pm* with primitive concept pairs invokes the equivalent intuition as $\mathfrak{s}^{\mathfrak{c}}$; however, this does not mean so in the aspect $\mathfrak{s}^{\mathfrak{r}}$. Allowing to define the similarity of defined role names, as in [16], may be not appropriate since defined role names are contributed by primitive role names. For example, let $r_1 \sqsubseteq s_1$ and $r_2 \sqsubseteq s_2$ are defined in $\mathcal{T}$. It is clear that $r_1, r_2 \in \mathsf{RN}^{\mathsf{def}}$. By defining $pm(r_1, r_2)$, the defined similarity should be also propagated to the similarity of s_1 and s_2. However, this point is not discussed in [16]. In respect of this, $\mathsf{RN}^{\mathsf{pri}}$ is merely used in $\mathfrak{s}^{\mathfrak{r}}$ and γ^{π} is defined for the similarity of defined role names. The authors of [16] also define the function $g : N_A \to \mathbb{R}_{>0}$ representing the weight for concept names and existential restriction atoms (based on their definition). Ones may feel the resemblance of g and $\mathfrak{i}^{\mathfrak{c}}, \mathfrak{i}^{\mathfrak{r}}$; however, they are also different in three perspectives. Firstly, the mapping of g is reached to the infinity whereas $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{i}^{\mathfrak{r}}$ are bounded. This characteristic of g is impractical to use as it may lead to the unbalance of weight assignments. For instance, one may define $g(A_1) = 1$ but $g(A_2) = 10^{12}$ where $A_1, A_2 \in \mathsf{CN}^{\mathsf{pri}}$. To avoid this situation, the authors should provide a guideline for weight assignments. Secondly, the mapping of g is lower bounded by one. This clearly makes an impossibility to define the intuition of having no importance. Thus, the situation given in Subsubsection 6.2.4 is not expressible. Lastly, the domain of g is the set of atoms whereas $\mathfrak{i}^{\mathfrak{c}}$ (and $\mathfrak{i}^{\mathfrak{r}}$) is the set of primitive concept names (and the set of role names, respectively). Using the set of atoms as the domain is also impractical since there can be infinitely many existential restriction atoms and the interpretation of functions is slightly dubious. For instance, given $g(\exists r.C) = 2$ and $g(\exists r.D) = 3$, do both r intentionally contribute the equal importance? Thus, this definition is inappropriate to represent the agent's perception. Moreover, the aspect $\mathfrak{d}$ disappears from [16]. Lacking of fully $\mathfrak{i}^{\mathfrak{c}}$ and $\mathfrak{d}$ makes the framework inappropriate to use for SNOMED CT applications. These distinctions of *simi* and sim^{π} are radically caused by their different motivations. Table 3 summarizes this discussion, where ✔ denotes totally identical to the specified function whereas ✓ denotes partially identical to the specified function.

CSM	$\mathfrak{i}^{\mathfrak{c}}$	$\mathfrak{i}^{\mathfrak{r}}$	$\mathfrak{s}^{\mathfrak{c}}$	$\mathfrak{s}^{\mathfrak{r}}$	$\mathfrak{d}$
sim^{π}	✔	✔	✔	✔	✔
the extended work of sim [10]	✔				✔
simi [16]	✓		✔	✓	

Table 3. Concept similarity measures which embed preference elements

Not only distinct on the mathematical representation of *simi* and sim^{π}, the desired properties presented in each work are also different. While the properties introduced in [16] are motivated for CSM, our properties are developed under the consideration of the agent's preferences ($\overset{\pi}{\sim}_{\mathcal{T}}$). Hence, some properties introduced for CSM are revised in subjective manners and the new property is introduced.

8 CONCLUSIONS AND FUTURE WORK

This paper introduces the notion called concept similarity measure under preference profile (in symbol, $\overset{\pi}{\sim}_{\mathcal{T}}$) with the set of its desirable properties, as intended behaviors of good preference-based measures. The measure sim^{π} (cf. Section 4), which is regarded as a measure of the proposed notion, is capable of informing the degree under preferences of similarity of two concepts although they are not in the subsumption relation. At the heart of the measure is the calculation of the degrees under preferences of homomorphism between two description trees in both directions. Proofs of inherited properties are shown in Theorems 4, 5, 6, and 7. The measure can also be used regardless of the agent's preferences. Theorem 3 suggests that this is handled by the default preference profile setting, i.e. sim^{π_0}.

Apart from the mathematical definition, we suggest two concrete algorithms, viz. the top-down approach (cf. Subsection 5.1) and the bottom-up approach (cf. Subsection 5.2), for implementations of sim^{π}. The computational complexity of both algorithms is clearly discussed and is practically evaluated against $\mathcal{O}_{\text{SNOMED}}$ (cf. Subsection 6.1). The usability of possible use cases are discussed in Subsection 6.2.

The proposed measure has great potential use in knowledge engineering, such as the development of recommendation systems based on the agent's preferences, the development of domain-specific knowledge bases, and the ontology engineering. Moreover, it may be used with heterogeneous ontologies by identifying duplicated primitive concepts and primitive roles among ontologies via $\mathfrak{s}^{\mathfrak{c}}$ and $\mathfrak{s}^{\mathfrak{r}}$, respectively.

There are several possible directions for the future research. Firstly, it appears to be a natural step to extend the notion of preference profile to support more expressive DLs, e.g. universal restriction, concept negation, and also, to support an ABox. Secondly, we also aim at devising a concept similarity measure under preference profile which can handle more expressive DLs. Thirdly, we intend to explore the possibility to extend the notion of preference profile beyond $\overset{\pi}{\sim}_{\mathcal{T}}$, e.g. non-standard instance checking under preference profile. Apart from theoretical perspectives, we also intend to explore possibility on optimizing the proposed algorithmic procedures.

Acknowledgment

This research is part of the JAIST-NECTEC-SIIT dual doctoral degree program; it is supported by the Japan Society for the Promotion of Science (JSPS kaken No. 17H02258) and is partly supported by CILS of Thammasat University and the NRU project of Thailand Office of Higher Education Commission.

REFERENCES

[1] Baader, F.—Calvanese, D.—McGuinness, D. L.—Nardi, D.—Patel-Schneider, P. F.: The Description Logic Handbook: Theory, Implementation and Applications. 2nd edition. Cambridge University Press, New York, NY, USA, 2010.

[2] Ashburner, M.—Ball, C. A.—Blake, J. A.—Botstein, D.—Butler, H.—Cherry, J. M.—Davis, A. P.—Dolinski, K.—Dwight, S. S.—Eppig, J. T.—Harris, M. A.—Hill, D. P.—Issel-Tarver, L.—Kasarskis, A.—Lewis, S.—Matese, J. C.—Richardson, J. E.—Ringwald, M.—Rubin, G. M.—Sherlock, G.: Gene Ontology: Tool for the Unification of Biology. Nature Genetics, Vol. 25, 2000, No. 1, pp. 25–29, doi: 10.1038/75556.

[3] Euzenat, J.—Valtchev, P.: Similarity-Based Ontology Alignment in OWL-Lite. In: de Mántaras, R. L., Saitta, L. (Eds.): Proceedings of the 16th European Conference on Artificial Intelligence (ECAI-04), IOS Press, 2004, pp. 333–337.

[4] Janowicz, K.—Wilkes, M.: SIM-DL$_A$: A Novel Semantic Similarity Measure for Description Logics Reducing Inter-Concept to Inter-Instance Similarity. In: Aroyo, L. et al. (Eds.): The Semantic Web: Research and Applications (ESWC 2009). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5554, 2009, pp. 353–367.

[5] Racharak, T.—Suntisrivaraporn, B.: Similarity Measures for $\mathcal{FL}_0$ Concept Descriptions from an Automata-Theoretic Point of View. Proceeding of the 2015 6th International Conference on Information and Communication Technology for Embedded Systems (IC-ICTES), 2015, pp. 1–6.

[6] D'Amato, C.—Fanizzi, N.—Esposito, F.: A Dissimilarity Measure for $\mathcal{ALC}$ Concept Descriptions. Proceedings of the 2006 ACM Symposium on Applied Computing (SAC '06), 2006, pp. 1695–1699, doi: 10.1145/1141277.1141677.

[7] Fanizzi, N.—D'Amato, C.: A Similarity Measure for the $\mathcal{ALN}$ Description Logic. Proceedings of Italian Conference on Computational Logic (CILC 2006), 2006, pp. 26–27.

[8] D'Amato, C.—Fanizzi, N.—Esposito, F.: A Semantic Similarity Measure for Expressive Description Logics. CoRR, abs/0911.5043, 2009.

[9] D'Amato, C.—Staab, S.—Fanizzi, N.: On the Influence of Description Logics Ontologies on Conceptual Similarity. In: Gangemi, A., Euzenat, J. (Eds.): Knowledge Engineering: Practice and Patterns (EKAW 2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5268, 2008, pp. 48–63.

[10] Tongphu, S.—Suntisrivaraporn, B.: Algorithms for Measuring Similarity Between $\mathcal{ELH}$ Concept Descriptions: A Case Study on Snomed ct. Computing and Informatics, Vol. 36, 2017, No. 4, pp. 733–764.

[11] Baader, F.: Terminological Cycles in a Description Logic with Existential Restrictions. Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI '03), San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2003, pp. 325–330.

[12] Baader, F.—Brandt, S.—Küsters, R.: Matching under Side Conditions in Description Logics. Proceedings of the 17th International Joint Conference on Artificial

Intelligence – Vol. 1 (IJCAI '01), San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2001, pp. 213–218.

[13] Suntisrivaraporn, B.: A Similarity Measure for the Description Logic $\mathcal{EL}$ with Unfoldable Terminologies. 2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS), 2013, pp. 408–413.

[14] Racharak, T.—Suntisrivaraporn, B.—Tojo, S.: Identifying an Agent's Preferences Toward Similarity Measures in Description Logics. In: Qi, G., Kozaki, K., Pan, J., Yu, S. (Eds.): Semantic Technology (JIST 2015). Springer International Publishing, Cham, Lecture Notes in Computer Science, Vol. 9544, 2016, pp. 201–208.

[15] Racharak, T.—Suntisrivaraporn, B.—Tojo, S.: sim^{π}: A Concept Similarity Measure under an Agent's Preferences in Description Logic $\mathcal{ELH}$. Proceedings of the 8th International Conference on Agents and Artificial Intelligence (ICAART 2016) – Vol. 2, 2016, pp. 480–487.

[16] Lehmann, K.—Turhan, A.-Y.: A Framework for Semantic-Based Similarity Measures for $\mathcal{ELH}$-Concepts. In: del Cerro, L. F., Herzig, A., Mengin, J. (Eds.): Logics in Artificial Intelligence (JELIA 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7519, 2012, pp. 307–319.

[17] Borgida, A.—Walsh, T. J.—Hirsh, H.: Towards Measuring Similarity in Description Logics. Working Notes of the International Description Logics Workshop, CEUR Workshop Proceedings, Vol. 147, 2005.

[18] Janowicz, K.: Sim-DL: Towards a Semantic Similarity Measurement Theory for the Description Logic $\mathcal{ALCNR}$ in Geographic Information Retrieval. In: Meersman, R., Tari, Z., Herrero, P. et al. (Eds.): OTM Workshops 2006. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4278, 2006, pp. 1681–1692, doi: 10.1007/11915072_74.

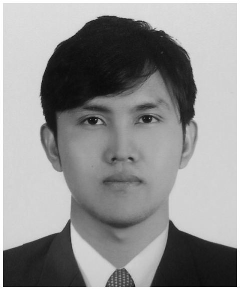
[19] Tversky, A.: Features of Similarity. Psychological Review, Vol. 84, 1977, No. 4, pp. 327–352, doi: 10.1037/0033-295X.84.4.327.

[20] Jaccard, P.: Étude Comparative de la Distribution Florale Dans Une Portion des Alpeset des Jura. Bulletin de la Societe Vaudoise des Sciences Naturellese, Vol. 37, 1901, pp. 547–579 (in French).

[21] Schulz, S.—Suntisrivaraporn, B.—Baader, F.: Snomed ct's Problem List: Ontologists' and Logicians' Therapy Suggestions. Studies in Health Technology and Informatics, Vol. 129, 2007, No. 1, pp. 802–806.

[22] Ge, J.—Qiu, Y.: Concept Similarity Matching Based on Semantic Distance. Fourth International Conference on Semantics, Knowledge and Grid, 2008, pp. 380–383, doi: 10.1109/SKG.2008.24.

[23] Giunchiglia, F.—Yatskevich, M.—Shvaiko, P.: Semantic Matching: Algorithms and Implementation. Journal on Data Semantics IX. Springer-Verlag, Berlin, Heidelberg, 2007, pp. 1–38, doi: 10.1007/978-3-540-74987-5_1.



Teeradaj Racharak received his Bachelor of Engineering with first class honors in software and knowledge engineering from Kasetsart University, Thailand, in 2010 and the Master of Engineering in computer science (with specialization in software engineering) from Asian Institute of Technology, Thailand, in 2012. Currently, he is Ph.D. student in the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST), Japan, and in the School of Information, Computer, and Communication Technology (ICT), Sirindhorn International Institute of Technology (SIIT), Thammasat University, Thailand, under the JAIST-NECTEC-SIIT dual doctoral degree program. His research interest involves artificial intelligence, particularly in knowledge representation and reasoning using diverse formalisms (e.g. description logics and argumentation framework) ranging from theoretical aspects to empirical development.

Boontawee Suntisrivaraporn received his B.Eng. with first class honors in computer engineering from King Mongkut's Institute of Technology Ladkrabang, Thailand, and then graduated with both degrees of M.Sc. and D.Eng. (summa cum laude) from TU Dresden, Germany, in the field of artificial intelligence. He taught full-time at Sirindhorn International Institute of Technology and held a visiting Associate Professor position at School of Information Science, Japan Advanced Institute of Technology. His research interests mainly include description logic, knowledge representation and reasoning, biomedical ontologies, and graph theory and applications. He served as a PC member of various conferences and also received best paper awards in relevant conferences, e.g. Medinfo, ASWC, JIST. From 2016, he moved to the private sectors with his current role as Lead Data Scientist at Business Intelligence and Data Science of Siam Commercial Bank, Thailand.

Satoshi Tojo received his Bachelor of Engineering, Master of Engineering, and Doctor of Engineering degrees from the University of Tokyo, Japan. He joined Mitsubishi Research Institute, Inc. (MRI) in 1983, and the Japan Advanced Institute of Science and Technology (JAIST), Ishikawa, Japan, as Associate Professor in 1995 and became Professor in 2000. His research interest is centered on grammar theory and formal semantics of natural language, as well as logic in artificial intelligence, including knowledge and belief of rational agents. Also, he has studied the iterated learning model of grammar acquisition, and linguistic models of western tonal music.

Computing and Informatics, Vol. 37, 2018, 614–634, doi: 10.4149/cai_2018_3_614

PARAMETER SELECTION AND UNCERTAINTY MEASUREMENT FOR VARIABLE PRECISION PROBABILISTIC ROUGH SET

Weimin Ma

Tongji University
School of Economics and Management
Shanghai 200092, P.R. China
e-mail: mawm@tongji.edu.cn

Lei Yue

Tongji University
School of Economics and Management
Shanghai 200092, P.R. China
&
Shandong University of Finance and Economics
Institute of Business Administration
Jinan 250014, P.R. China
e-mail: yuechaolei@163.com

Bingzhen Sun*

Xidian University
School of Economics and Management
Xi'an 710071, P.R. China
e-mail: bzsun@xidian.edu.cn

Haiyan Zhao

Shanghai University of Engineering Science
Shanghai 200092, P.R. China

* Corresponding author

Abstract. In this paper, we consider the problem of parameter selection and uncertainty measurement for a variable precision probabilistic rough set. Firstly, within the framework of the variable precision probabilistic rough set model, the relative discernibility of a variable precision rough set in probabilistic approximation space is discussed, and the conditions that make precision parameters α discernible in a variable precision probabilistic rough set are put forward. Concurrently, we consider the lack of predictability of precision parameters in a variable precision probabilistic rough set, and we propose a systematic threshold selection method based on relative discernibility of sets, using the concept of relative discernibility in probabilistic approximation space. Furthermore, a numerical example is applied to test the validity of the proposed method in this paper. Secondly, we discuss the problem of uncertainty measurement for the variable precision probabilistic rough set. The concept of classical fuzzy entropy is introduced into probabilistic approximation space, and the uncertain information that comes from approximation space and the approximated objects is fully considered. Then, an axiomatic approach is established for uncertainty measurement in a variable precision probabilistic rough set, and several related interesting properties are also discussed. Thirdly, we study the attribute reduction for the variable precision probabilistic rough set. The definition of reduction and its characteristic theorems are given for the variable precision probabilistic rough set. The main contribution of this paper is twofold. One is to propose a method of parameter selection for a variable precision probabilistic rough set. Another is to present a new approach to measurement uncertainty and the method of attribute reduction for a variable precision probabilistic rough set.

Keywords: Rough set, probabilistic approximation space, relative discernibility, variable precision probabilistic rough set, approximation reduction

1 INTRODUCTION

Rough set [1, 2] theory is a mathematical theory that has been used since the 1980s to handle uncertain, imprecise, and incomplete information. In recent years, rough set theory has been applied successfully to many fields in computer science and management science, e.g., intelligent data processing [3], data mining [4], big data processing [5], pattern identification [6], image processing [3, 7], decision-making support and process control [3, 8]. Against a varied background of management science situations, several extensions of the Pawlak rough set model have been developed, such as a variable precision rough set model [9], rough set model based on tolerance relations [10, 11], Bayesian rough set model [12], fuzzy rough set model [13], rough fuzzy set model [14], probabilistic rough set model [15], etc. The basic model of classical Pawlak rough set is based on equivalence relations (reflexive, symmetric, and transitive are satisfied), and is represented by a precise inclusion relationship of sets. In this model, equivalence relationship is the crucial concept, and the universe of discourse is divided into a positive region, negative region and boundary region.

But the strict equivalence relationship leads to a relatively broad boundary region, i.e., an uncertainty region. Therefore, finding methods to minimize the boundary region has become a highly discussed issue in both the theoretical and applied study of rough set theory.

The core issue of rough set theory is classification analysis based on a binary relationship for a given domain. The Pawlak rough set model is by nature a qualitative classification model. It classifies based on an equivalence relationship and inclusion relationship between approximation sets, with no consideration of possible overlap of information between set objects. In view of this, incomplete inclusion relations and subordination relations among sets have to be considered. Regarding this constraint of classical Pawlak rough set theory, an effective approach to reduce the boundary region is to extend a strict inclusion relation between sets, and to introduce a majority inclusion relation between sets. This leads to an important extension of Pawlak rough set, i.e., variable precision rough set [9]. Subsequently, many valuable extensions of Ziarko's variable precision rough set were established. Based on Ziarko's idea, Beynon [16] and Katzberg [17] define the model of variable precision rough sets with asymmetric bounds by introducing two parameters to the lower and upper approximations. Sun and Gong [18] present a new generalized model of Ziarko's variable precision rough set based on binary relations over the universe of discourse. In addition, probabilistic rough set theory that combines classical probability theory and Pawlak rough set theory is another effective model to reduce the boundary domain of a Pawlak rough set. In 1987, Wong and Ziarko [19] introduced probabilistic approximation space to the study of rough set and then presented the concept of probabilistic rough set. Subsequently, Yao et al. [20] proposed a more general probabilistic rough set called decision-theoretic rough set. There another perspective to deal with the degree of overlap of an equivalence class with the set to be approximated was given, and an approach to select the needed parameters in lower and upper approximations was presented. As far as the probabilistic approach to rough set theory, Pawlak and Skowron [21], Pawlak et al. [22] and Wong and Ziarko [19] proposed a method to characterize a rough set by a single membership function. By the definition of a rough membership function, elements in the same equivalence class have the same degree of membership. The rough membership may be interpreted as the probability of any element belonging to a set, given that the element belongs to an equivalence class. This interpretation leads to probabilistic rough set [23]. Compared with a classical Pawlak rough set model, variable rough set models and probabilistic rough set models belong to the category of quantitative models, and they can effectively overcome the weaknesses of the Pawlak rough set model in terms of tolerance mechanism and generalization capacity when processing imprecise, inconsistent, and incomplete information.

Finally, in [24], the authors defined another quantitative model by combining a variable precision rough set model and a probabilistic rough set model into a variable precision probabilistic rough set model. The three above-mentioned quantitative extension models based on classical Pawlak rough set models share one common point: a precision parameter needs to be set in advance in their definition of lower and upper approximations. While introduction of a precision parameter can improve the defects of the classical Pawlak rough set theory model, studies of the three models so far have only discussed the existence, domain of the value, and a semantic explanation (management background) of precision parameters. They lack discussion of how to determine precision parameters, i.e., they did not propose a definition of parameters and the selection method. This has constrained the application of the models. Besides, the parameters used in certain conditions are not necessary in some cases. In this case, the credibility of decision rules may be reduced. In fact, the selection of parameters is of vital importance to the selection of decision rules in real life during the application of a variable precision probability rough set model. In this paper, based on [25], we discuss the relative discernibility between sets in probabilistic approximation space. We put forward a method of threshold selection of precision parameters based on the relative discernibility of sets under the precondition of relative discernibility of the set in probabilistic approximation space and consistency of the quality of approximation classification. This makes it practical to select precise parameters in a variable precision probabilistic rough set model. Uncertainty measurement of concepts (objects) in approximation space is another important part of the study of Pawlak rough set theory. As in the Pawlak rough set model, the roughness and precision of variable precision probabilistic rough set only describe uncertainties that come from the approximation space. In fact, the uncertainty of a rough set in approximation space comes from both the approximation space and the approximated set. In view of this, we fully discuss both factors. The concept of fuzzy entropy is introduced into probabilistic approximation space, and an axiomatic approach is employed to put forward a new method to address measurement uncertainty in a variable precision probabilistic rough set. Finally, we briefly introduce an approximation reduction of an information system based on a variable precision probabilistic rough set model.

The rest of this paper is as follows: Section 2 provides the basic concept of binary relations over the universe and briefly reviews the Pawlak rough set theory, variable precision rough set and probabilistic rough set. In Section 3, the discernibility of probabilistic approximation space of variable precision probabilistic rough set is discussed first, then a parameter selection method for a variable precision probabilistic rough set is proposed on this basis. Section 4 investigates the uncertainty measurement of a variable precision probabilistic rough set by introducing the concept of classical fuzzy entropy into probabilistic approximation space. Section 5 discusses the attribute reduction of probabilistic approximation space based on variable precision probabilistic rough set and presents several interesting conclusions. At last, we conclude our research and set out further research directions in Section 6.

2 PRELIMINARIES

In this section, we briefly review the concept of binary relations over a universe as well as the Pawlak rough set model over the universe. Also, we will present the definitions of the variable precision rough set model and probabilistic rough set model.

2.1 Pawlak Rough Set

First of all, we present the definition of an equivalence relation in the universe of discourse.

Definition 1 ([1, 2])**.** Let U be a non-empty and finite universe. Denote $U \times U = \{(x_i, x_j) \mid x_i, x_j \in U\}$. Then the subset $R \subseteq U \times U$ is called an equivalence relation on universe U, if R satisfies the following conditions:

1. Reflexivity: $(x_i, x_i) \in R$, $\forall x_i \in U$;
2. Symmetry: $(x_i, x_j) \in R, \Rightarrow (x_j, x_i) \in R, \forall x_i, x_j \in U$;
3. Transitivity: $(x_i, x_j) \in R, (x_i, x_k) \in R \Rightarrow (x_i, x_k) \in R, \forall x_i, x_j, x_k \in U$.

Let U/R be a set consisting of all equivalent classes based on equivalence relation R in the universe, and let $[x]_R$ represent R equivalent classes that include element $x \in U$. Then $K = (U, R)$ is called a knowledge base or a relationship system, where R represents a cluster of equivalence relationships in domain U.

When there is no risk of confusion, we make no distinction as to equivalence relationship cluster R and equivalence relationship R, i.e., $K = (U, R)$. Meanwhile (U, R) is called Pawlak approximation space [26, 27].

Let (U, R) be the approximation space. For any $X \subseteq U$, we define

$$\underline{R}(X) = \cup\{[x]_R \in U/R \mid [x]_R \subseteq X, x \in U\},$$
$$\overline{R}(X) = \cup\{[x]_R \in U/R \mid [x]_R \cap X \neq \emptyset, x \in U\},$$

the lower approximation and upper approximation of R, respectively.

The lower approximation and upper approximation can also be represented as follows:

$$\underline{R}(X) = \{x \in U \mid [x]_R \subseteq X\},$$
$$\overline{R}(X) = \{x \in U \mid [x]_R \cap X \neq \emptyset\}.$$

$Bn_R(X) = \overline{R}(X) - \underline{R}(X)$ is called the boundary region of X. $Pos_R(X) = \underline{R}(X)$ is the positive region of X, and $Neg_R(X) = U - \overline{R}(X)$ is the negative region of X.

Apparently, $\overline{R}(X) = Pos_R(X) \cup Bn_R(X)$.

Based on the above definition, the following conclusion is obviously valid.

Theorem 1 ([27, 29])**.** Define (U, R) as an approximation space. For any $X \subseteq U$, there is:

1. X is a definable set of R when $\overline{R}(X) = \underline{R}(X)$.
2. X is a rough set of R when $\overline{R}(X) \neq \underline{R}(X)$.

The lower approximation $\underline{R}(X)$ is the union of all elementary sets that are the subsets of X, and the upper approximation $\overline{R}(X)$ is the union of all elementary sets that have a non-empty intersection with X.

The lower (upper) approximation $\underline{R}(X)(\overline{R}(X))$ is interpreted as the collection of those elements of U that definitely (possibly) belong to X.

2.2 Variable Precision Rough Set

The Pawlak rough set model is often too strict when including objects into the approximation regions and may require additional information. A lack of consideration for the degree of overlap between an equivalence class and the set to be approximated unnecessarily limits the applications of Pawlak rough set and has motivated a good deal of new generalization of the Pawlak rough set model. In 1993, Ziarko [9, 28] proposed the variable precision rough set model by introducing the majority inclusion relation over the universe of discourse. In the following, we present Ziarko's variable precision rough set model.

Let U be the universe of discourse. For any two subsets $X, Y \subseteq U$, we define

$$mc(X,Y) = \begin{cases} 1 - |X \cap Y|/|X|, & |X| > 0; \\ 0, & |X| = 0. \end{cases}$$

We call $mc(X,Y)$ the relative error classification rate of set X in relation to set Y.

Let (U,R) be the Pawlak approximation space, for any $X \subseteq U$. We define the β lower approximation and upper approximation of X with respect to approximation space (U,R) respectively as follows:

$$\underline{R}_\beta(X) = \{x \in U \mid mc([x]_R, X) \leq \beta\},$$
$$\overline{R}_\beta(X) = \{x \in U \mid mc([x]_R, X) < 1 - \beta\}.$$

Furthermore, the β positive region, boundary region and negative region of X with respect to approximation space (U,R) can be respectively defined as follows:

$$Pos_\beta(X) = \underline{R}_\beta(X) = \{x \in U \mid mc([x]_R, X) \leq \beta\},$$
$$Bn_\beta(X) = \{x \in U \mid \beta < mc([x]_R, X) < 1 - \beta\},$$
$$Neg_\beta(X) = \{x \in U \mid mc([x]_R, X) \geq 1 - \beta\}.$$

Remark 1. If $\beta = 0$, then the following relation holds:

$$\underline{R}_\beta(X) = \{x \in U \mid mc([x]_R, X) \geq 1\} = \{x \in U \mid [x]_R \subseteq X\} = \underline{R}(X),$$
$$\overline{R}_\beta(X) = \{x \in U \mid mc([x]_R, X) > 0\} = \{x \in U \mid [x]_R \cap X \neq \emptyset\} = \overline{R}(X).$$

This is the Pawlak rough set model.

2.3 Variable Precision Probabilistic Rough Set

In this subsection, we introduce another generalized form of the Pawlak rough set model: variable precision probabilistic rough set.

We first give the concept of probabilistic measurement on the universe of discourse [29, 30].

Definition 2 ([30])**.** Let U be a non-empty finite universe of discourse. The set function $P : 2^U \to [0, 1]$ is called the probabilistic measurement on universe U, and satisfies the following conditions:

1. $P(\emptyset) = 0$,
2. $P(U) = 1$,
3. $P(\bigcup_n A_n) = \sum_n P(A_n)$, $A_n \in 2^U, n = 1, 2, \ldots$ and A_n piecewise disjoint.

Let P be the probabilistic measurement on U, $\forall A, B \in 2^U$ and $P(B) > 0$. Then,

$$P(A|B) = \frac{P(A \cap B)}{P(B)},$$

is the conditional probability of occurrence of event A given event B.

Definition 3 ([24])**.** Let U be a non-empty and finite universe of discourse. R is an equivalence relation on U. U/R are equivalence classes formed by R. P is the probabilistic measurement defined on the σ-algebra of measurable subsets of U. Then we call this the probabilistic approximation space.

In the following, we present the definition of variable precision probabilistic rough set with respect to probabilistic approximation space.

Let $A_P = (U, R, P)$ be a probabilistic approximation space. For any $0.5 < \alpha \leq 1$, $X \subseteq U$, the lower approximation $\underline{P}_\alpha(X)$ and upper approximation $\overline{P}_\alpha(X)$ of X with precision parameter α about probabilistic approximation space A_P are, respectively, as follows:

$$\underline{P}_\alpha(X) = \{x \in U \mid P(X|[X]_R) \geq \alpha\},$$
$$\overline{P}_\alpha(X) = \{x \in U \mid P(X|[X]_R) > 1 - \alpha\}.$$

Similarly, the positive region, boundary region and negative region of X about probabilistic approximation space A_P are, respectively, defined as follows:

$$Pos(X,\alpha) = \underline{P}_\alpha(X) = \{x \in U \mid P(X|[X]_R) \geq \alpha\},$$

$$Bn_\alpha(X) = \{x \in U \mid 1-\alpha < P(X|[X]_R) < \alpha\},$$

$$Neg_\alpha(X) = U \backslash \overline{P}_\alpha(X) = \{x \in U \mid P(X|[X]_R) < \alpha\}.$$

3 THE THRESHOLD SELECTION OF VARIABLE PRECISION PROBABILISTIC ROUGH SET

According to the definition of the variable precision probabilistic rough set model, it is well known that roughness of any non-empty subset X ($X \subseteq U$) in approximation space is caused by the existence of boundary region $Bn_\alpha(X)$. Therefore, the boundary region $Bn_\alpha(X)$ affects the discernibility of X, and the boundary region of X varies with parameter α, which further influences the discernibility of the set itself. Because the discernibility of the boundary region of X is relative, a higher discernibility of X in a given classification probability value can be reached if a greater classification probability exists. Based on the above analysis, the following definitions are given.

As is well known, the variable precision probabilistic rough set model [24] is an extension of the existing results. The variable precision probabilistic rough set model was defined by introducing the classical probability measure into the Pawlak approximation space, and then we use the conditional probability of any objects (i.e., the equivalence classes of an element on universe of discourse) with respect to the considered event (i.e., the approximated object set X) instead of the majority include relation used in the original Ziarko's model [9]. Based on the conditional probability, the lower and upper approximations of variable probabilistic rough set model were constructed. That is, the variable precision probabilistic rough set model can be regarded as a probabilistic description of the original Ziarko's model [9] under the framework of probabilistic approximation space, i.e., the variable precision probabilistic rough set model will be degenerated into the original Ziarko's model when we define $P(X|[X]_R) = 1 - \frac{|X \cup [X]_R|}{|[X]_R|} = mc(X,[X]_R)$ (where $|\bullet|$ denotes the cardinality of any set). Therefore, the following conclusions and other results of the variable precision probabilistic rough set model are similar to the results given in Ziarko [9].

Definition 4 ([9])**.** Let (U,R,P) be a probabilistic approximation space. If the α boundary region of X ($X \subseteq U$) about (U,R,P) satisfies $Bn_\alpha(X) = \emptyset$ or, equivalently, $\underline{P}_\alpha(X) = \overline{P}_\alpha(X)$. Then X is called α discernible. Otherwise, X is called α indiscernible.

It is easy to know that the discernibility of X depends on the value of precision parameter α from this definition.

Based on the above definition, the following conclusion can be reached.

Theorem 2 ([9])**.** Let (U, R, P) be a probabilistic approximation space. For any X $(X \subseteq U)$, if X is discernable for parameter α $(0.5 < \alpha \leq 1)$, then X is also discernible for any $\alpha_1 < \alpha$ $(0.5 < \alpha_1 \leq 1)$.

Proof. For $Bn_\alpha(X) = \{x \in U | 1 - \alpha < P(X|[X]_R) < \alpha\}$, if X is discernible on parameter α $(0.5 < \alpha \leq 1)$, then there is $Bn_\alpha(X) = \emptyset$. For any α that satisfies $\alpha_1 < \alpha$ $(0.5 < \alpha_1 \leq 1)$, there is $\{1-\alpha_1 < P(X|[X]_R) < \alpha_1\} \subseteq \{1-\alpha < P(X|[X]_R) < \alpha\}$. This is $Bn_{\alpha_1}(X) \subseteq Bn_\alpha(X)$. So, $Bn_\alpha(X) = \emptyset$. Therefore, X is discernible for any $\alpha_1(\alpha_1 < \alpha)$.

This completes the proof. □

Corollary 1 ([9])**.** Let (U, R, P) be a probabilistic approximation space. For any X $(X \subseteq U)$, if X is indiscernible with respect to parameter α $(0.5 < \alpha \leq 1)$, then X is indiscernible for any $\alpha < \alpha_2 (0.5 < \alpha_2 \leq 1)$.

The proof is same as the proof of Theorem 2.

Theorem 2 and Corollary 1 show that the discernibility of any set X increases with the decreasing of the value of precision parameter α. Otherwise, the discernibility of any set X decreases with the increasing value of precision parameter α. That is, for a probabilistic rough set X, there could be a more highly discernible X if a smaller classification precision parameter α was given.

Definition 5. Let U be a non-empty finite universe, and (U, R, P) be a probabilistic approximation space. For any X $(X \subseteq U)$, if $\alpha = 0.5$, we define the absolute boundary region of X about probabilistic approximation space (U, R, P) as:

$$Bn_{0.5}(X) = \{x \in U | P(X|[X]_R) = 0.5\}.$$

Definition 6. Let (U, R, P) be a probabilistic approximation space. For any X $(X \subseteq U)$, if X is indiscernible for any α $(0.5 < \alpha \leq 1)$, then we call X absolutely indiscernible (or absolutely rough set). Otherwise, we call X relatively rough (or weakly discernible).

Theorem 3 ([9])**.** Let (U, R, P) be a probabilistic approximation space. For any X $(X \subseteq U)$, if $\underline{P}_{0.5}(X) \neq \overline{P}_{0.5}(X)$, then X is indiscernible for any precision parameter α $(0.5 < \alpha \leq 1)$.

Proof. Because $\underline{P}_{0.5}(X) \neq \overline{P}_{0.5}(X)$, i.e. $Bn_{0.5}(X) \neq \emptyset$, and for any α $(0.5 < \alpha \leq 1)$, there is $Bn_{0.5}(X) \subseteq Bn_\alpha(X)$. Therefore, $Bn_{0.5}(X) \neq \emptyset$ according to Theorem 2. From Definition 4, it is known that X is α indiscernible.

This completes the proof. □

Corollary 2. Let (U, R, P) be a probabilistic approximation space. Then any X $(X \subseteq U)$ with a non-empty boundary region on (U, R, P) must be indiscernible.

Corollary 3. Let (U, R, P) be probabilistic approximation space. For any X $(X \subseteq U)$, X is an absolutely rough set if and only if $Bn_{0.5}(X) \neq \emptyset$.

It is easy to see that the converse propositions of Theorem 2 and 3 are still valid for the definition of various precision probabilistic rough sets.

Generally speaking, for any set X $(X \subseteq U)$, the discernibility of X about probabilistic approximation space (U, R, P) depends on the value of precision parameter α. In fact, there is always an α for every relative rough set X that makes X discernible at this threshold value. Then we have the following definition.

Definition 7 ([9])**.** Let U be the non-empty finite universe of discourse, and $R \in U \times U$ an equivalence relation on universe U. Let

$$Ind(R, X) = \{\alpha \mid Bn_\alpha(X) \neq \emptyset, \alpha \in (0.5, 1]\}$$

be the whole set of α values that satisfy X and is indiscernible with respect to probabilistic approximation space (U, R, P).

Furthermore, the maximum value of parameter α that satisfies the condition that X is discernible is called the discernible threshold value, denoted as $\gamma_\alpha(P, X)$.

Definition 8 ([9])**.** Let (U, R, P) be a probabilistic approximation space. For any X $(X \subseteq U)$, the discernible threshold value $\gamma_\alpha(P, X)$ satisfies the following conditions:

1. $\gamma_\alpha(P, X) = \mathit{inf}\ Ind(P, X)$,
2. $\gamma_\alpha(P, X) = \min(n_1, n_2)$

where

$$n_1 = 1 - \max\{P(X|[x])|P(X|[X]_R) < 0.5, x \in U\},$$
$$n_2 = \min\{P(X|[x])|P(X|[X]_R) > 0.5, x \in U\}.$$

For any X $(X \subseteq U)$, if X is relatively discernible, then the empty boundary region (i.e., the discernible threshold value boundary region) of X is as follows:

$$Bn_{\gamma_\alpha}(X) = \{x \in U \mid 1 - \gamma_\alpha(P, X) < P(X|[x]_\alpha) < \gamma_\alpha(P, X)\}.$$

Theorem 4. Let (U, R, P) be a probabilistic approximation space. If there are $Bn_\alpha(X) \neq \emptyset$ for any parameter $\alpha \in (0.5, 1]$ and X $(X \subseteq U)$, then X is indiscernible if and only if

$$Bn_\alpha(X) \neq Bn_{\gamma_\alpha}(X), \alpha \in (0.5, 1].$$

In Theorem 4 we talk about α for which X is discernible, therefore, according to Definition 4 we have $Bn_\alpha(X) = \emptyset$.

Based on the former definitions, the following conclusion is clear.

Theorem 5. Let (U, R, P) be a probabilistic approximation space. For any α $(0.5 < \alpha \leq 1)$, X $(X \subseteq U)$, the domain of the probabilistic value X that makes α discernible is as follows:

$$(0.5, \gamma_\alpha(P, X)].$$

In the following, we present a numerical example to demonstrate the method for precision parameter selection given in this paper.

Example 1. Let $U = \{x_1, x_2, \cdots, x_{20}\}$ and let R be an equivalence relation on universe U. P is the probabilistic measurement defined on the σ-algebra of measurable subsets of universe U. Meanwhile, the elementary classes of elements on U with respect to R are as follows, respectively.

$$E_1 = \{x_1, x_2, x_3, x_4, x_5\}, \qquad E_2 = \{x_6, x_7, x_8\}, \qquad E_3 = \{x_9, x_{10}, x_{11}, x_{12}, x_{19}\},$$

$$E_4 = \{x_{17}, x_{18}, x_{20}\}, \qquad E_5 = \{x_{13}, x_{14}, x_{15}, x_{16}\}.$$

Suppose that

$$X = \{x_3, x_5, x_8, x_{14}, x_{15}, x_{16}, x_{18}, x_{19}\}.$$

Take $P(X|E) = \frac{P(X \cap E)}{P(E)}$. Then we have

$$P(X|E_1) = 0.4, \qquad P(X|E_2) = 0.34, \qquad P(X|E_3) = 0.2,$$

$$P(X|E_4) = 0.33, \qquad P(X|E_5) = 0.75.$$

Based on Definition 7, it is easy to calculate and obtain the following results:

$$\begin{aligned} n_1 &= 1 - \max\{P(X|E_1), P(X|E_2), P(X|E_3)\} \\ &= 1 - \max\{0.4, 0.34, 0.33, 0.2\} \\ &= 1 - 0.4 = 0.6, \end{aligned}$$

$$n_2 = \min\{P(X|E_5)\} = \min\{0.75\} = 0.75.$$

So, there is $\gamma_\alpha(P, X) = \min\{n_1, n_2\} = \min\{0.6, 0.75\} = 0.6$.

That is, the maximum threshold value that makes X discernible is $\gamma_\alpha(P, X) = 0.6$. Therefore, the corresponding empty boundary region when X is discernible is as follows:

$$Bn_{\gamma_\alpha}(X) = \{x \in U \mid 0.4 < P(X|[x]_R) < 0.6\}.$$

By using the conclusion of Theorem 4, we know that the domain of the value of precision parameter α that makes X discernible is calculated as follows:

$$(0.5, \gamma_\alpha(P, X)] = (0.5, 0.6].$$

This completes the example.

4 UNCERTAINTY MEASUREMENT OF VARIABLE PRECISION PROBABILISTIC ROUGH SET

In the classical Pawlak rough set theory [1, 2], accuracy and roughness are used to characterize the uncertainty of a set and approximation accuracy is employed to depict the accuracy of a rough classification. Pawlak [1, 2] developed the uncertainty measurement of an ordinary set in the universe of discourse. Subsequently, Banerjee [34] studied the uncertainty measurement of a fuzzy set with respect to approximation space. Although these measures are effective, several limitations have been pointed out by many scholars when applying them to certain situations. Therefore, several improved methods of uncertainty measurement for various generalized rough set models (or generalized information systems) have been established in recent years [37].

As is well known, the roughness of any object set with respect to the probabilistic approximation space is induced by the non-empty boundary region, from the definition of the variable precision probabilistic rough set model. There could be a fuzzy membership relation between any object set and the elements in the universe of discourse. Moreover, the fuzzy membership degree between any object set and the elements is determined by the probability $P(X|[x])$.

For any α $(0.5 < \alpha \leq 1)$, $X \subseteq U$, we denote the fuzzy set generated by the conditional probability as $\tilde{X}_P^\alpha$. So, its membership function is defined as follows:

$$\tilde{X}_P^\alpha(x) = P(X|[X]_R) = P(X \cap [X]_R)/P([X]_R), x \in U.$$

In particular, if $P([X]_R) = 0$ or $[X]_R = \emptyset$, then we use the convention that $\tilde{X}_P^\alpha(x) = 1$.

Based on the definition of rough membership function $\tilde{X}_P^\alpha(x)$ on probabilistic approximation space (U, R, P), the lower and upper approximations of variable precision probability rough set by rough membership function are represented respectively as follows:

$$\underline{P}_\alpha(X) = \left\{x \in U \mid \tilde{X}_P^\alpha(x) \geq \alpha\right\},$$

$$\overline{P}_\alpha(X) = \left\{x \in U \mid \tilde{X}_P^\alpha(x) > 1 - \alpha\right\}.$$

That is, the lower and upper approximations of variable precision probabilistic rough set are α cut set and strong $1 - \alpha$ cut set of fuzzy set $\tilde{X}_P$, respectively.

The boundary region and negative region of X are similarly described as follows:

$$Bn_\alpha(X) = \left\{x \in U \mid 1 - \alpha < \tilde{X}_P^\alpha(x) < \alpha\right\},$$

$$Neg_\alpha(X) = U \backslash \overline{P}_\alpha(X) = \left\{x \in U \mid \tilde{X}_P^\alpha(x) \leq 1 - \alpha\right\}.$$

In this section, we will present an approach to uncertainty measurement for variable precision probabilistic rough set by using the concept of fuzzy entropy. The concept of entropy, originally developed by Shannon [33] for communication theory, has been a useful mechanism for characterizing the information in various models and applications in diverse fields. By using Shannon entropy, several conclusions can be established about the uncertainty measurement and knowledge granularity of the rough set in the Pawlak approximation space [35]. As discussed above, there is a fuzzy set generated by the conditional probability of the universe of discourse for any target set X ($X \subseteq U$). So, we use the concept of fuzzy entropy to discuss the uncertainty measurement for a variable precision probabilistic rough set.

Here, we first give the definition of fuzzy entropy as follows.

Let U be a non-empty and finite universe of discourse. Denote as $F(U)$ all the fuzzy subsets of universe U.

Definition 9 ([31, 32, 33])**.** Let mapping $E : F(U) \to [0,1]$. If the following conditions are satisfied:

1. $E(A) = 0$ if and only if A is a crisp set on U;
2. $E(A) = 1$ if and only if $\mu_A(x) = 0.5, \forall x \in U, A \in F(U)$;
3. If $D(A, 0.5) \geq D(B, 0.5)$, then $E(A) \leq E(B), \forall A, B \in F(U)$;
4. $E(A) = E(A^c)$ (A^c is the complementary set of A)

where $D(A,B) = \sqrt{\frac{1}{|U|}\sum_{x\in U}(\mu_A(x) - \mu_B(x))^2}$ indicates the distance between two rough sets.

Then E is called an entropy on $F(U)$.

With the axiomatic definition of fuzzy entropy, a roughness measurement of a variable precision probabilistic rough set is put forward.

Definition 10. Let (U, R, P) be a probabilistic approximation space. For any $X \in U$, the roughness measurement of rough set X with respect to (U, R, P) is defined as follows:

$$f(\tilde{X}_P^\alpha) = \frac{4}{|U|}\sum_{x\in U}\tilde{X}_P^\alpha(x)\left(1 - \tilde{X}_P^\alpha(x)\right).$$

Lemma 1. Let (U, R, P) be probabilistic approximation space. For any X ($X \subseteq U$), $f\left(\tilde{X}_P^\alpha\right)$ is a fuzzy entropy over $F(U)$.

Proof. For any X ($X \subseteq U$), it is easy to verify that the relation $0 \leq \tilde{X}_P^\alpha(x) \left(1 - \tilde{X}_P^\alpha(x)\right) \leq \frac{1}{4}$ holds. Therefore, $0 \leq f\left(\tilde{X}_P^\alpha\right) \leq 1$.

In the following, we will verify the conditions given in Definition 9 one by one for the roughness measurement $f\left(\tilde{X}_P^\alpha\right)$ of a variable precision probabilistic rough set.

1. If $\tilde{X}_P^\alpha$ is a crisp set, then for any X there is $\tilde{X}_P^\alpha(x) = 0$ or $\tilde{X}_P^\alpha(x) = 1$. So, $f\left(\tilde{X}_P^\alpha\right) = 0$. On the other hand, if $f\left(\tilde{X}_P^\alpha\right) = 0$, then there is $\tilde{X}_P^\alpha(x)$ $\left(1 - \tilde{X}_P^\alpha(x)\right) = 0$ for any $x \in U$. Thus, there is $\tilde{X}_P^\alpha(x) = 0$ or $\tilde{X}_P^\alpha(x) = 1$, i.e., $\tilde{X}_P^\alpha(x)$ is a crisp set.
2. If $x \in U$, $\tilde{X}_P^\alpha(x) = 0.5$, then there is $1 - \tilde{X}_P^\alpha(x) = 0.5$. Moreover, there is $\tilde{X}_P^\alpha(x)\left(1 - \tilde{X}_P^\alpha(x)\right) = 0.25$. So, $f\left(\tilde{X}_P^\alpha\right) = \frac{4}{|U|}\sum_{x\in U}\frac{1}{4} = 1$.

   On the contrary, suppose that $f\left(\tilde{X}_P^\alpha\right) = 1$. Then, there is $\tilde{X}_P^\alpha(x)\left(1 - \tilde{X}_P^\alpha(x)\right)$ $= \frac{1}{4}$, which holds by the above discussion. This proves that $\tilde{X}_P^\alpha(x) = 0.5$. In other words, $\tilde{X}_P^\alpha(x)$ arrives at the maximum fuzziness.
3. If $D\left(\tilde{X}_P^\alpha, 0.5\right) \geq D\left(\tilde{Y}_P^\alpha, 0.5\right)$, for any $X, Y \subseteq U$, there is

$$\begin{aligned} f\left(\tilde{X}_P^\alpha\right) &= \frac{4}{|U|}\sum_{x\in U}\tilde{X}_P^\alpha(x)\left(1 - \tilde{X}_P^\alpha(x)\right) \\ &= \frac{4}{|U|}\sum_{x\in U}\left(0.5 + \tilde{X}_P^\alpha(x) - 0.5\right)\left(0.5 - \left(\tilde{X}_P^\alpha(x) - 0.5\right)\right) \\ &= \frac{4}{|U|}\sum_{x\in U}\left(0.25 - \left(\tilde{X}_P^\alpha(x) - 0.5\right)^2\right) = 1 - \frac{4}{|U|}\sum_{x\in U}\left(\tilde{X}_P^\alpha(x) - 0.5\right)^2 \\ &\leq 1 - \frac{4}{|U|}\sum_{x\in U}\left(\tilde{Y}_P^\alpha(x) - 0.5\right)^2 = f\left(\tilde{Y}_P^\alpha\right). \end{aligned}$$

4. For any $x \in U$, there is $\left(\tilde{X}^c\right)_P^\alpha(x) = 1 - \tilde{X}_P^\alpha(x)$, $\left(\tilde{X}^c\right)_P^\alpha(x) = 1 - \tilde{X}_P^\alpha(x) = \left(1 - \tilde{X}_P^\alpha(x)\right)\tilde{X}_P^\alpha(x)$. That is, there is $f\left(\tilde{X}_P^\alpha\right) = f\left(\left(\tilde{X}^c\right)_P^\alpha\right)$.

Therefore, according to the results of 1., 2., 3. and 4., we know that $f\left(\tilde{X}_P^\alpha\right)$ is a fuzzy entropy on $F(U)$.

This completes the proof. □

By using Definition 9 and Lemma 1, the following results about the uncertainty measurement of variable precision probabilistic rough set are clear.

Theorem 6. Let (U, R, P) be a probabilistic approximation space. For any $0.5 < \alpha \leq 1$ and $X \subseteq U$, there is:

1. $f(U) = f(\emptyset) = 0$.
2. For any $X \in U$, if $\underline{P}_\alpha(X) = \overline{P}_\alpha(X)$, there is $f\left(\tilde{X}_P^\alpha\right) = 0$.
3. For any $X \in U$, if $[x] \neq \emptyset$, there is $f\left(\tilde{X}_P^\alpha\right) = f\left(\left(\tilde{X}^c\right)_P^\alpha\right)$.

Proof. It can be verified directly by the definitions. □

5 ATTRIBUTE REDUCTION OF VARIABLE PRECISION PROBABILISTIC ROUGH SET

In general, objects are described by different attributes. However, it is not necessary to know all attributes for the classification of information systems. That is, some attributes are unnecessary and do not affect the result of classification when removed from the attribute set. Meanwhile, some attributes are indispensable to the result of classification and affect the result when removed from the attribute set. Furthermore, some attributes are relatively necessary for the classification and may determine the result by associating with other attributes. The attribute reduction presents a minimum attribute subset completely describing the classification as the original attribute set for information systems [26, 36, 37, 38, 39, 40, 41]. This subsection will investigate the problem of attribute reduction for an information system based on variable precision probabilistic rough set.

Let U be a non-empty finite universe of discourse. $\Re$ is a family of equivalence relationships over the universe U. Let $K \subseteq \Re$ ($K \neq \emptyset$) and the intersection of all equivalence relations in K be called indiscernible relations on [3]. We denote the intersection of all equivalence relations as $ind(K)$.

Definition 11. Let $S = (U, V, A, F)$ be an information system. $C, D \subseteq A$ are respectively a conditional attribute and decision attribute of probabilistic approximation space (U, R, P). Then, the α ($0.5 < \alpha \leq 1$) approximation dependence of conditional attribute C and decision attribute D of probabilistic approximation space is defined as follows:

$$d_\alpha(C, D) = \frac{|\bigcup_{P(X|[X]_R)\geq\alpha}\{X \mid X \in U/ind(D)\}|}{|U|}.$$

By the definition of approximation dependence in Definition 11, we can easily know that this concept is a natural generalization of the classical approximation dependence in Pawlak rough set theory.

Specifically, $d_\alpha(C, D)$ will degenerate into the classical approximation dependence in Pawlak rough set theory when $\alpha = 1$.

Here, the attribute reduction means that the minimum attribute subset of the conditional attributes results in the same approximation dependence with respect to the decision attribute. We then present the definition of the α approximation attribute reduction for an information system based on variable precision probabilistic rough set theory by using the concept of α ($0.5 < \alpha \leq 1$) approximation dependence as follows.

Definition 12. Let $S = (U, V, A, F)$ be an information system. $C, D \subseteq A$ are respectively conditional attribute and decision attribute of probabilistic approximation space (U, R, P). Then the α approximation reduction $Red_\alpha(C, D)$ is a minimum attribute subset of conditional attribute set C and satisfies the following conditions:

1. $d_\alpha(C, D) = d_\alpha(Red_\alpha(C, D), D)$;
2. The equation given in (1) will no longer be valid when any one attribute is removed from $Red_\alpha(C, D)$.

Theorem 7. Let $S = (U, V, A, F)$ be an information system. Then the approximation reduction of approximation space S always exists for any precision parameter α $(0.5 < \alpha \leq 1)$.

Proof. If for any $c \in C \subseteq A$, and satisfies $R_{C-\{c\}} \neq R_C$, then C is a reduction of information system $S = (U, V, A, F)$. Otherwise, for any $c \in C \subseteq A$, and satisfies $R_{C-\{c\}} = R_C$ hold. Then, we consider the new attribute subset $C_1 = C - \{c\}$. Meanwhile, if for any $c_1 \in C_1 \subseteq A$, there is $R_{C_1-\{c_1\}} \neq R_C$ hold, then, C_1 is a reduction of information system $S = (U, V, A, F)$. Otherwise, for any $c_1 \in C_1 \subseteq A$, there is $R_{C_1-\{c_1\}} = R_C$ hold. Next, we further consider $C_2 = C_1 - \{c_1\}$ and repeat the above process. So, we will find the minimum attribute subset $C^* \subseteq C$ that satisfies the relationships $R_{C^*} = R_C$ and $R_{C^*-\{c\}} \neq R_C$ for any $c \in C^*$. Therefore, C^* is the reduction of information system $S = (U, V, A, F)$.

This completes the proof. □

In general, there may not be only one reduction for information system $S = (U, V, A, F)$ because there may be different combinations among the elements of the attribute set. In practice, we focus on finding only one of the reductions for the information system $S = (U, V, A, F)$.

6 CONCLUSIONS

By introducing precision parameter α $(0.5 < \alpha \leq 1)$ into classical probabilistic rough set, the variable precision probabilistic rough set converts two parameters in the upper (lower) approximation of probabilistic rough set into one parameter. This further improves the robustness and adaptability of the model, extends classical Pawlak rough set theory, and allows rough set theory to process random, uncertain, and inconsistent data information more effectively. Meanwhile, new models and approaches are proposed in which rough set theory is applied to solve decision-making problems with uncertainty in actual situations in management science.

In this paper, we discuss two issues for the variable precision probabilistic rough set model: the relative discernibility of any object set in the universe and the uncertainty measurement of the variable precision probabilistic rough set. For the first aspect content, it is well known that the variable precision probabilistic rough set model [24] is an extension of the existing results by combining the variable precision rough set model [9] and the probabilistic rough set model [22, 23]. The basic idea of the variable precision probabilistic rough set model was defined by introducing the classical probability measure into the Pawlak approximation space, and the conditional probability of any objects (i.e., the equivalence classes of an element on universe of discourse) with respect to the considered event (i.e., the approximated object set X) instead of the majority include relation used in the original Ziarko's model [9]. However, the variable precision probabilistic rough set model will be degenerated into the original Ziarko's model when we define $P(X|[X]_R) = 1 - \frac{|X \cup [X]_R|}{|[X]_R|} = mc(X, [X]_R)$. At the same time, the variable precision probabilistic rough set model will be degenerated into the probabilistic rough set model when we define $1-\alpha = \beta$ in the upper approximation. So, all the results about the relative discernibility of any object set of universe with the variable precision probabilistic rough set are the generalization of the original Ziarko's model [9]. Similarly, the results will be degenerated into the corresponded conclusions of Ziarko's model when we define $P(X|[X]_R) = 1 - \frac{|X \cup [X]_R|}{|[X]_R|} = mc(X, [X]_R)$. Therefore, the results given in Ziarko's model [9] are based on the classical Pawlak approximation space and the results obtained in this paper are under the framework of probabilistic approximation space. For the second aspect, we investigate the uncertainty measurement of any object set with respect to the variable precision probabilistic rough set model. By introducing the concept of the fuzzy entropy into the probabilistic approximation space, we establish a new approach to measure the uncertainty of the approximation quality of any object set with respect to the variable precision probabilistic rough set model. Further, we explore the attribute reduction for the information systems based on the variable precision probabilistic rough set model. Factually, random information acquisition and decision-making problems under general relations or one kind of certain relation are used more widely, and this offers direction for our further study.

Acknowledgements

The authors are very grateful to the Deputy of the Editor-in-Chief Professor Jacek Kitowski, and the three anonymous referees for their thoughtful comments and valuable suggestions. Some remarks directly benefit from the referees' comments. The work was partly supported by the National Natural Science Foundation of China (No. 71571090, No. 71161016), the National Science Foundation of Shaanxi Province of China (2017JM7022), the Key Strategic Project of Fundamental Research Funds for the Central Universities (JBZ170601), the Interdisciplinary Foundation of Humanities and Information (RW180167), the Social Science Planning Project Fund of Xi'an (17J64).

REFERENCES

[1] Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences, Vol. 11, 1982, No. 5, pp. 341–356, doi: 10.1007/BF01001956.

[2] Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers, London, 1991, doi: 10.1007/978-94-011-3534-4.

[3] Pawlak, Z.—Skowron, A.: Rudiments of Rough Sets. Information Sciences, Vol. 177, 2007, No. 1, pp. 3–27, doi: 10.1016/j.ins.2006.06.003.

[4] Lingras, P. J.—Yao, Y. Y.: Data Mining Using Extensions of the Rough Set Model. Journal of the American Society for Information Science, Vol. 49, 1998, No. 5, pp. 415–422.

[5] Yamaguchi, D.: Attribute Dependency Functions Considering Data Efficiency. International Journal of Approximate Reasoning, Vol. 51, 2009, No. 1, pp. 89–98, doi: 10.1016/j.ijar.2009.08.002.

[6] Li, T. R.—Ruan, D.—Geert, W.—Song, J.—Xu, Y.: A Rough Sets Based Characteristic Relation Approach for Dynamic Attribute Generalization in Data Mining. Knowledge-Based Systems, Vol. 20, 2007, No. 5, pp. 485–494, doi: 10.1016/j.knosys.2007.01.002.

[7] Mac Parthaláin, N.—Shen, Q.: Exploring the Boundary Region of Tolerance Rough Sets for Feature Selection. Pattern Recognition, Vol. 42, 2009, No. 5, pp. 655–667, doi: 10.1016/j.patcog.2008.08.029.

[8] Dash, M.—Liu, H.: Feature Selection for Classification. Intelligent Data Analysis, Vol. 1, 1997, No. 1-4, pp. 131–156, doi: 10.1016/S1088-467X(97)00008-5.

[9] Ziarko, W.: Variable Precision Rough set Model. Journal of Computer and System Sciences, Vol. 46, 1993, No. 1, pp. 39–59, doi: 10.1016/0022-0000(93)90048-2.

[10] Kryszkiewicz, M.: Rough Set Approach to Incomplete Information Systems. Information Sciences, Vol. 112, 1998, No. 1-4, pp. 39–49, doi: 10.1016/S0020-0255(98)10019-1.

[11] Kryszkiewicz, M.: Rules in Incomplete Information Systems. Information Sciences, Vol. 113, 1999, No. 3-4, pp. 271–292, doi: 10.1016/S0020-0255(98)10065-8.

[12] Slezak, D.—Ziarko, W.: The Investigation of the Bayesian Rough Set Model. International Journal of Approximate Reasoning, Vol. 40, 2005, No. 1-2, pp. 81–91, doi: 10.1016/j.ijar.2004.11.004.

[13] Dubois, D.—Prade, H.: Rough Fuzzy Sets and Fuzzy Rough Sets. International Journal of General Systems, Vol. 17, 1990, No. 2-3, pp. 191–209, doi: 10.1080/03081079008935107.

[14] Dubois, D.—Prade, H.: Putting Rough Sets and Fuzzy Sets Together. Intelligent Decision Support, Handbook of Applications and Advances of the Rough Sets Theory. Kluwer Academic Publishers, 1992, doi: 10.1007/978-94-015-7975-9_14.

[15] Yao, Y.: Probabilistic Rough set Approximations. International Journal of Approximate Reasoning, Vol. 49, 2008, No. 2, pp. 255–271.

[16] Beynon, M. J.: The Introduction and Utilization of (l,u)-Graphs in the Extended Variable Precision Rough Sets Model. International Journal of Intelligent Systems, Vol. 18, 2003, No. 10, pp. 1035–1055, doi: 10.1002/int.10130.

[17] Katzberg, J. D.—Ziarko, W.: Variable Precision Rough Sets with Asymmetric Bounds. IEEE International Workshop on Rough Sets and Knowledge Discovery, Springer-Verlag, Heidelberg, 1993, pp. 167–177.

[18] Gong, Z. T.—Sun, B. Z.: Variable Precision Rough Set Model Based on General Relation. Journal of Lanzhou University (Nature Edition), Vol. 41, 2005, No. 2, pp. 110–114.

[19] Wong, S. K. M.—Ziarko, W.: Comparison of the Probabilistic Approximate Classification and the Fuzzy Set Model. Fuzzy Sets and Systems, Vol. 21, 1987, No. 3, pp. 357–362.

[20] Yao, Y. Y.—Wong, S. K. M.—Lingras, P.: A Decision-Theoretic Rough Set Model. The 5th International Symposium on Methodologies for Intelligent Systems, 1990, pp. 17–24.

[21] Pawlak, Z.—Skowron, A.: Rough Membership Functions. Advances in the Dempster-Shafer Theory of Evidence. John Wiley and Sons, New York, 1994, pp. 251–271.

[22] Pawlak, Z.—Wong, S. K. M.—Ziarko, W.: Rough Sets: Probabilistic Versus Deterministic Approach. International Journal of Man-Machine Studies, Vol. 29, 1988, No. 1, pp. 81–95, doi: 10.1016/S0020-7373(88)80032-4.

[23] Yao, Y. Y.—Wong, S. K. M.: A Decision Theoretic Framework for Approximating Concepts. International of Journal Man-Machine Studies, Vol. 37, 1992, No. 6, pp. 793–809.

[24] Sun, B. Z.—Gong, Z. T.: Variable Precision Probabilistic Rough Set Model. Journal of Northwest Normal University (Nature Edition), Vol. 41, 2005, No. 2, pp. 23–26.

[25] Beynon, M.: Reducts within the Variable Precision Rough Sets Model: A Further Investigation. European Journal of Operational Research, Vol. 134, 2001, No. 1, pp. 592–605, doi: 10.1016/S0377-2217(00)00280-0.

[26] Sun, B. Z.—Ma, W. M.: Rough Approximation of a Preference Relation by Multi-Decision Dominance for a Multi-Agent Conflict Analysis Problem. Information Sciences, Vol. 315, 2015, pp. 39–53, doi: 10.1016/j.ins.2015.03.061.

[27] Wei, J. M.—Wang, M. Y.—You, J. P.—Wang, S. Q.—Liu, D. Y.: VPRSM Based Decision Tree Classifier. Computing and Informatics, Vol. 26, 2007, No. 6, pp. 663–677.

[28] Gong, Z. T.—Shi, Z. H.—Yao, H. X.: Variable Precision Rough Set Model for Incomplete Information Systems and Its Beta-Reducts. Computing and Informatics, Vol. 31, 2012, No. 6+, pp. 1385–1399.

[29] Zhang, W. X.—Wu, W. Z.—Liang, J. Y.—Li, D. Y.: Theory and Methodology of Rough Set. Science Press, Beijing, 2001.

[30] Yan, J. A.: Theory of Measure. Science Press, Beijing, 1998.

[31] Liu, X. C.: Entropy Distance Measure and Similarity Measure of Fuzzy Sets and Their Relations. Fuzzy Sets and Systems, Vol. 52, 1992, No. 3, pp. 305–318.

[32] Liang, J. Y.—Li, D. Y.: Knowledge Acquirement and Uncertainty Measurement for Information Systems. Science Press, Beijing, 2005, pp. 39–46.

[33] Shannon, C. E.: The Mathematical Theory of Communication. The Bell System Technical Journal, Vol. 27, 1948, No. 3, pp. 379–423.

[34] Banerjee, M.—Sankar, K. P.: Roughness of a Fuzzy Set. Information Sciences, Vol. 93, 1996, No. 3-4, pp. 235–246, doi: 10.1016/0020-0255(96)00081-3.

[35] Sun, B. Z.—Ma, W. M.: Uncertainty Measure for General Relation-Based Rough Fuzzy Set. Kybernetes, Vol. 42, 2013, No. 6, pp. 979–992.

[36] Sun, B. Z.—Ma, W. M.—Gong, Z. T.: Dominance-Based Rough Set Theory over Interval-Valued Information Systems. Expert Systems, Vol. 31, 2014, No. 2, pp. 185–197.

[37] Sun, B. Z.—Ma, W. M.—Chen, D. G.: Rough Approximation of a Fuzzy Concept on a Hybrid Attribute Information System and Its Uncertainty Measure. Information Sciences, Vol. 284, 2014, pp. 60–80, doi: 10.1016/j.ins.2014.06.036.

[38] Sun, B. Z.—Ma, W. M.—Zhao, H. Y.: Decision-Theoretic Rough Fuzzy Set Model and Application. Information Sciences, Vol. 283, 2014, pp. 180–196, doi: 10.1016/j.ins.2014.06.045.

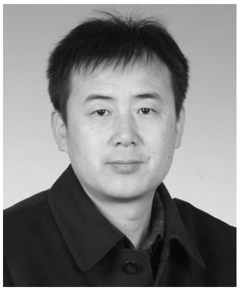
[39] Wu, Q.: Knowledge Granulation, Rough Entropy and Uncertainty Measure in Incomplete Fuzzy Information System. Computing and Informatics, Vol. 33, 2014, No. 3, pp. 633–651.

[40] Zhan, J. M.—Liu, Q.—Dawaz, B. J.: A New Rough Set Theory: Rough Soft Hemirings. Journal of Intelligent and Fuzzy Systems, Vol. 28, 2015, No. 4, pp. 1687–1697.

[41] Sun, B. Z.—Ma, W. M.—Zhao, H. Y.: Rough Set-Based Conflict Analysis Model and Method over Two Universes. Information Sciences, Vol. 372, 2016, pp. 111–125, doi: 10.1016/j.ins.2016.08.030.

Weimin Ma received his B.Sc. degree from the Department of Mechanical Manufacturing Engineering at the Northwest Polytechnic University in Xi'an, China, in 1993. He received his M.Sc. degree and Ph.D. degree in management science and engineering from the School of Economics and Management, Xi'an Jiaotong University, Xi'an, China, in 1999 and 2003, respectively. He has published more than 100 articles in international journals and book chapters. He is currently Professor of the School of Economics and Management, serving as an authorized Ph.D. supervisor in Management Science and Engineering, Tongji University in Shanghai, China. His research interests include on-line computation, fuzzy sets and systems, information systems and technology, decision-making with uncertainty, algorithm design and operations research.

Lei Yue is a Ph.D. candidate in management science and engineering, Tongji University in Shanghai, P.R. China. He received his B.Sc. degree from the School of Business Administration at Shandong Finance Institute in Shandong, P.R. China, in 2002. He received his M.Sc. degree in engineering administration from the School of Management, Tianjin University, Tianjin, P.R. China, in 2007. His research interests include human resource evaluation.



Bingzhen Sun received his B.Sc. degree and M.Sc. degree in mathematics from Northwest Normal University, Lanzhou, China, in 2003 and 2006, respectively. He received his Ph.D. degree in management sciences and engineering from Tongji University, Shanghai, China, in 2013. He is currently Professor of the School of Economics and Management, serving as an authorized Ph.D. supervisor in management science and engineering, Xidian University in Xi'an, China. He has published over 30 articles in international journals. His research interests include rough set theory and applications, fuzzy sets and systems, decision-making under uncertainty and operations research.

Haiyan Zhao received her B.Sc. degree in industrial and commercial management from Jilin University, Jilin, China, in 2004 and her M.Sc. degree in management sciences and engineering from Harbin Institute of Technology, Harbin, China, in 2006. She received her Ph.D. degree in management sciences and engineering from Tongji University, Shanghai, China, in 2017. She is currently Associate Professor of Shanghai University of Engineering Science. Her research interests include soft set theory and applications in decision-making with uncertainty.

Computing and Informatics, Vol. 37, 2018, 635–655, doi: 10.4149/cai_2018_3_635

HIERARCHICAL SYSTEM DESIGN USING REFINABLE RECURSIVE PETRI NET

Messaouda Bouneb

Department of Mathematic and Computer Science
El Arbi ben M'hidi University
Oum el boighi, Algeria
e-mail: bounebm.univ@gmail.com

Djamel Eddine Saidouni

Department of Computer Science
Abed Elhamid Mehri Constantine 2 University
Constantine, Algeria
e-mail: saidounid@hotmail.com

Jean Michel Ilie

Department of Computer Science
Pierre and Marie Curie University
Paris, France
e-mail: jean-michel.ilie@lip6.fr

Abstract. This paper is in the framework of the specification and verification of concurrent dynamic systems. For this purpose we propose the model of Refinable Recursive Petri Nets (RRPN) under a maximality semantics. In this model a notion of undefined transitions is considered. The underlying semantics model is the Maximality Abstract Labeled Transition System (AMLTS). Then, the model supports a definition of a hierarchical design methodology. The example of a cutting flame machine is used for illustrating the approach.

Keywords: Recursive Petri nets, hierarchical design, action refinement, maximality labeled transition system

1 INTRODUCTION

Petri nets model is a graphical and mathematical modelling tool which is used to specify, in clear manner, the behaviors of concurrent systems. The marking graph associated with a given Petri net is used for checking the specified properties of the system. Indeed, this marking graph is seen as a labeled transition system. However, labeled transition systems are based on interleaving semantics. This later represents parallel executions by their interleaved sequential executions. To clarify the ideas, we consider the example of two Petri nets (Figures 1 a) and 1 b)). Figure 1 a) represents a system which can execute transitions t_1 and t_2 in parallel, whereas Figure 1 b) represents a system that executes sequentially transitions t_1 and t_3 or transitions t_2 and t_4.

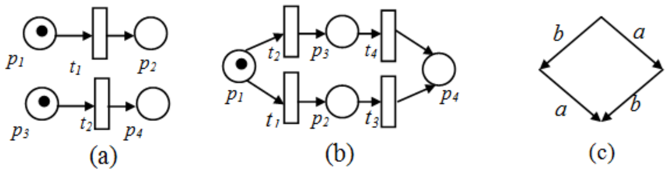


Figure 1. Petri nets

After generating the marking graphs of the two Petri nets, where transitions t_1 and t_4 are labeled by action a and transitions t_2 and t_3 are labeled by action b, the two marking graphs become isomorphic. Therefore the parallel execution of action a and action b is interpreted as their interleaved executions in time. This result is acceptable under the assumption that the firing of each transition corresponds to the execution of an indivisible action with null duration (structural and temporal atomicity of actions). Nevertheless, in reality this assumption is not accepted. In order to accept the verification results, the realization constraints should be taken into account at both specification and semantic level. To clarify the idea, let us consider that the transition t_1 (resp. t_4) consists of two sequential transitions t_{1-1} and t_{1-2} (resp. t_{4-1} and t_{4-2}). Transitions t_{1-1} and t_{4-1} are labeled by action a_1 whereas transitions t_{1-2} and t_{4-2} are labeled by action a_2. The refined Petri nets and their labeled transition systems are represented by Figure 2. It is clear that the behaviors of both Petri nets are different.

Indeed, in the first system, the execution of action b can occur between the execution of actions a_1 and a_2; which is not the case in the second system. Taking into account the non atomicity of actions in a system has been deeply studied in the literature through the definition of several semantics supporting the concept of action refinement [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. Considering such semantics allows a hierarchical design of the systems by refining actions (actions are seen as abstract processes). An other interest of these semantics is the characterization of parallel executions of non instantaneous actions.

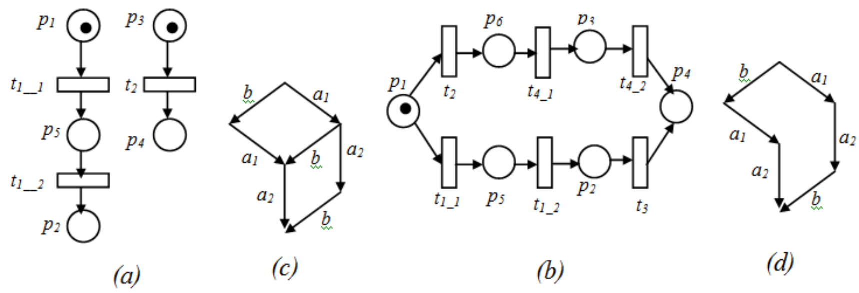


Figure 2. Non structural atomicity of actions

In this context the maximality semantics, through the model of the maximality labeled transition systems, was used for the characterization of concurrent systems. This semantics was defined for process algebras and place transition Petri nets [13, 14, 15, 16, 17]. However, the limits of Petri net model have been highlighted for the specification of systems with dynamic structures such as multi agent systems. For this reason, recursive Petri nets have been defined. Dynamic behaviors are considered through abstract transitions. Since abstract transitions represent activities, the association of true concurrency semantics to the model becomes more appropriate than the use of interleaving semantics. For this purpose, in [18] a maximality operational semantics has been proposed for recursive Petri net model. In order to design concurrent systems, several methodologies have been proposed in the literature, around process algebra and Petri net specification models. As an example we can cite the lotosphere design methodology which is based on the formal description technique Lotos. This design methodology consists in specifying firstly the architecture of the system in terms of its observable behavior, then the specification is refined along the design trajectory until obtaining the more detailed specification. This later takes into account all system functionalities and the environment constraints. As the model is based on the interleaving semantics, the design methodology is not based on action refinement but on transforming a specification, subject of refinement, to an other following some directives. An other design methodology has been defined for hierarchical Petri nets [19]. In [20] a Petri net approach to refining object behavioral specifications has been proposed. As for Lotos, these models are based on the interleaving semantics, too. Consequently design methodologies are not formal.

Since in [18] a true concurrency semantics has been defined for recursive Petri net, it seems interesting to define a design methodology based on refining abstract transitions. Thus, in this paper we extend the model of recursive Petri net by considering refinable transitions. The model will be named Refinable Recursive Petri net. The proposed model is based on the maximality semantics. As for recursive Petri net, dynamic behaviors are considered through abstract transitions. These abstract transitions can be used for a hierarchical system design. Indeed, at a level of abstraction, the details of abstract behavior of a transition may be hidden. They will be exhibited in a further level of abstraction. In the first step, details of abstract transitions behaviors are undefined. These behaviors are gradually introduced along the design trajectory. In this manner, system components are integrated gradually; the initial specification is then the most abstract. This remark leads us to label abstract transitions, the behaviors of which are undefined, by the symbol $\perp$. As an example, let us consider the systems of Figure 1 where transitions t_1 and t_4 are abstract transitions, at this level the difference between the systems may not be seen. The labeled transition systems associated with the two Petri nets are given by Figures 3 a) and 3 b), respectively. To consider the details of refinement of abstract transitions, it is necessary to define relations on behaviors that consider the indefinite character of abstract transitions interstates. It is clear that two undefined transitions may become different after their refinement.

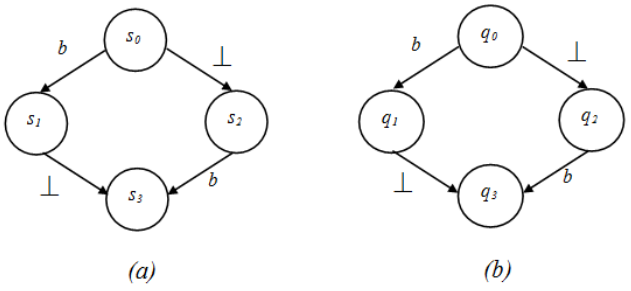


Figure 3. Interleaving approach for recursive Petri nets

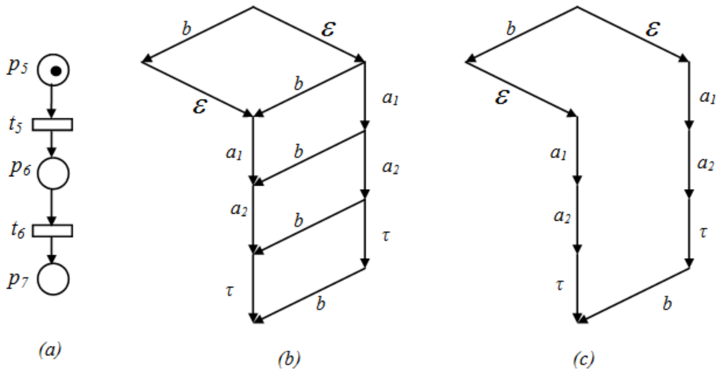


Figure 4. Refinement of abstract transitions

2 REFINABLE RECURSIVE PETRI NETS

A refinable recursive Petri net is a recursive Petri net on which all transitions are labeled by actions and abstract transitions may be labeled by $\perp$ (the abstract transitions the behaviors of which are undefined).

Formal Definitions

Definition 1. A refinable recursive Petri net is 9-uple $R = (P, T, I, W^-, W^+, \Omega, \gamma, K, \lambda)$ such that:

- P is a finite set of places.
- T is a finite set of transitions such that: $T = T_{el} \cup T_{ab}$ and $T_{el} \cap T_{ab} = \emptyset$. T_{el} denotes elementary transitions and T_{ab} denotes the abstract transitions, knowing that: $T_{ab} = T_{abd} \cup T_{abi}$ and $T_{abd} \cap T_{abi} = \emptyset$. The set of abstract transitions where behavior is defined are noted T_{abd} and the set of abstract transitions where behavior is indefinite are noted T_{abi}.
- $I = I_c \cup I_p$ is a finite set of indexes, indicates the cut steps and preemptions. $I \subset \mathbb{N}$.
- $W^- : P \times T \longrightarrow N$ is the matrix of precondition.
- $W^+ : P \times [T_{el} \cup (T_{ab} \times I) \longrightarrow N]$ is the matrix of post-condition.
- $\Omega : T_{ab} \longrightarrow \mathbb{N}^P$ is a function which associates to each abstract transition an ordinary marking (starting marking).
- γ is a family indexed by the set of termination I_c. Each set is specified as an effective representation of semi linear set of final markings (markings of termination on which the standard operations like union, intersection, projection and complementation, member test are applicable).
- $K : T_{el} \times T_{ab} \longrightarrow I_p$ is a partial function of control preemption.
- $\lambda : T \longrightarrow \mathbb{L} \cup \{\perp\}$ such that $\perp$ is the undefined label. λ is the labeling function which associates to each transition an action name. $\mathbb{L}$ ranged over by a, b, ... In practice the transition label is the name of an action.

  - $\forall t \in T_{abd}, \lambda(t) \in \mathbb{L}$.
  - $\forall t \in T_{abi}, \lambda(t) = \perp$

Definition 2. Let $R_1 = \left(P_1, T_1, I_1, W_1^-, W_1^+, \Omega_1, \gamma_1, K_1, \lambda_1\right)$ and $R_2 = (P_2, T_2, I_2, W_2^-, W_2^+, \Omega_2, \gamma_2, K_2, \lambda_2)$ be two refinable recursive Petri nets such that:

- $t \in T_{abd1}$ is an abstract transition, knowing that $T_{abd1} \subset T_{ab1} \subset T_1$.
- I_{ct} is the indexes set of the cuts of transition t.
- I_{pt} is the indexes set of preemption of transition t.
- $v \in \mathbb{N}^P$ is the start marking of transition t.
- $\gamma_t = \{\gamma_i / i \in I_{ct}\}$ is the termination set of transition t.

Then $\rho(t, R_1, R_2, I_{ct}, I_{pt}, v, \gamma_t) = \left(P_3, T_3, I_3, W_3^-, W_3^+, \Omega_3, \gamma_3, K_3, \lambda_3\right)$ is the refinable recursive Petri net obtained after the refinement of t in R_1 by R_2, such that:

- $P_3 = P_1 \cup P_2$.

- $T_3 = T_{el3} \cup T_{ab3}$ such that:
  - $T_{el3} = T_{el1} \cup T_{el2}$.
  - $T_{ab3} = T_{abd3} \cup T_{abi3}$ with:
    * $T_{abd3} = T'_{abd1} \cup T_{abd2}$ with $T'_{abd1} = T_{abd1} \cup \{t\}$.
    * $T_{abi3} = T'_{abi1} \cup T_{abi2}$ with $T'_{abi1} = T_{abi1} - \{t\}$.
- $I_3 = I_{c3} \cup I_{p3}$ such that:
  - $I_{c3} = I'_{c1} \cup I_{c2}$ with $I'_{c1} = I_{c1} \cup I_{ct}$.
  - $I_{p3} = I'_{p1} \cup I_{pt}$.
- $W_3^- : P_3 \times T_3 \longrightarrow \mathbb{N}$ such that $t' \in T_3$:

$$W_3^-\left(p', t'\right) = \begin{cases} W_1^-\left(p', t'\right), & \text{if } p' \in P_1, t' \in T_1, \\ W_2^-\left(p', t'\right), & \text{if } p' \in P_2, t' \in T_2. \end{cases}$$

- $W_3^+ : P_3 \times [T_{el3} \cup (T_{abd3} \times I_3) \cup T_{abi3}] \longrightarrow \mathbb{N}$ such that: $\forall t' \in T_3$:

$$W_3^+\left(p', t', i'\right) = \begin{cases} W_1^+\left(p', t', i'\right), & \text{if } p' \in P_1, t' \in T_1, \\ W_2^+\left(p', t', i'\right), & \text{if } p' \in P_2, t' \in T_2. \end{cases}$$

- $\Omega_3 : T_{abd3} \longrightarrow \mathbb{N}^P$ such that $\forall t' \in T_{abd3}$:

$$\Omega_3\left(t'\right) = \begin{cases} \Omega_1\left(t'\right), & \text{if } t' \in T_{abd1}, \\ \Omega_2\left(t'\right), & \text{if } t' \in T_{abd2}, \\ v, & \text{if } t' = t. \end{cases}$$

- $\gamma_3 = \gamma'_1 \cup \gamma_2$ such that: $\gamma'_1 = \gamma_1 \cup \gamma_t$.
- $K_3 : T_{el3} \times T_{abd3} \longrightarrow I_{p3}$ such that: $\forall t_1 \in T_{el3}, t_2 \in T_{abd3}$:

$$K_3\left(t_1, t_2\right) = \begin{cases} K_1\left(t_1, t_2\right), & \text{if } t_1 \in T_{el1} \text{ and } t_2 \in T_{abd1}, \\ K_2\left(t_1, t_2\right) & \text{if } t_1 \in T_{el2} \text{ and } t_2 \in T_{abd2}, \\ i, & \text{such that } i \in \left(\mathbb{N} - \left(I_1 \cup I_2\right)\right). \end{cases}$$

- $\lambda_3 : T_3 \longrightarrow \mathbb{L} \cup \{\perp\}$: such that $\forall t' \in T_3$:

$$\lambda_3\left(t'\right) = \begin{cases} \lambda_1\left(t'\right), & \text{if } t' \in T'_{abi1} \cup T'_{abd1}, \\ \lambda_2\left(t'\right), & \text{if } t' \in T_2, \\ a, & \text{otherwise with } a \in \mathbb{L}. \end{cases}$$

3 MAXIMALITY-BASED LABELED TRANSITION SYSTEMS

A maximality-based labeled transitions system is a graph labeled on both states and transitions. Each state is labeled by a set of event names. Each event name identifies the start of execution of an action which occured before this state. This action is said to be potentially under execution in this state. A transition between two states s_i and s_j is labeled by a 3-uple (G, a, x) (noted ${}_{G}a_{x}$) where x is the event name identifying the start of execution of the action a and G identifies the set of event names representing the causes of the action a. Elements of G belong to state s_i. Occurence of this transition terminates actions identified by G, thus, the set of event names corresponding to state s_j is that of s_i from which the set G is substructed and the event name x is added. The formal definition of a maximality-based labeled transition system is given in Definition 3.

Formal Definitions

Definition 3. Let $\mathcal{H}$ be a countable set of event names. $2^{\mathcal{H}}$ denotes the set of part-set of $\mathcal{H}$.

A maximality-based labeled transitions system of support $\mathcal{H}$ is a fivefold $(\eta, \varphi, \mu, \xi, \theta)$ with: $\eta = \langle S, TR, \alpha, \beta, S_0 \rangle$ is a system of transitions such that:

- S is the set of states in which the system may be found, this set can be finite or infinite.
- TR is the set of transitions indicating the change of states which the system can do; this set can be finite or infinite.
- α and β are two applications of TR in S such that for any transition $tr \in TR$ we have: $\alpha(tr)$ is the origin of the transition tr and $\beta(tr)$ its goal.
- S_0 is the initial state of the transition system η.
- (η, φ) is a system of transitions labeled by the function φ on an alphabet $\mathbb{L}$, called support of (η, φ). $(\varphi : TR \longrightarrow \mathbb{L})$ such that $\mathbb{L}$ ranged over by a, b, ... In practice a transition label is a name of an action.
- $\theta : S \longrightarrow 2^{\mathcal{H}}$ is a function which associates to each state a finite set of maximal event names. With the assumption that $\theta(S_0) = \emptyset$.
- $\mu : TR \longrightarrow 2^{\mathcal{H}}$ is a function which associates to each transition a finite set of event names corresponding to the actions which began their execution and their terminations cause the execution of this transition.
- $\xi : TR \longrightarrow \mathcal{H}$ is a function which associates to each transition the event name identifying its occurrence.

With the condition that for each transition $tr \in TR$ $\mu(tr) \subseteq \theta(\alpha(tr))$, $\xi(tr) \notin \theta(\alpha(tr)) - \mu(tr)$ and $\theta(\beta(tr)) = (\theta(\alpha(tr)) - \mu(tr)) \cup \{\xi(tr)\}$.

Notation 1. Let $mlts = (\eta, \varphi, \mu, \xi, \theta)$ be a maximality-based labeled transitions system such that $\eta = \langle S, TR, \alpha, \beta, S_0 \rangle$. $tr \in TR$ is a transition such that $\alpha(tr) = s$, $\beta(tr) = s'$, $\varphi(tr) = a$, $\mu(tr) = E$ and $\xi(tr) = x$. The transition tr will be noted $s \xrightarrow{E a_x} s'$.

Definition 4. Let $mlts_1 = (\eta_1, \varphi_1, \mu_1, \xi_1, \theta_1)$ and $mlts_2 = (\eta_2, \varphi_2, \mu_2, \xi_2, \theta_2)$ be two maximality labeled transition systems with: $\eta_1 = (S_1, TR_1, \alpha_1, \beta_1, s_{01})$ and $\eta_2 = (S_2, TR_2, \alpha_2, \beta_2, s_{02})$. $mlts_1$ and $mlts_2$ are isomorphic if there exists a bijection $h : S_1 \longrightarrow S_2$ such that: $\forall s, s' \in S_1 / s = \alpha_1(tr)$ and $s' = \beta_1(tr)$ then: $tr \in TR_1 \Longleftrightarrow tr' \in TR_2$ with:

- $\alpha_2(tr') = h(s)$ and $\beta_2(t) = h(s')$.
- $\theta_1(s) = \theta_2(h(s))$ and $\theta_1(s') = \theta_2(h(s'))$.
- $\varphi_1(tr) = \varphi_2(tr')$.
- $\mu_1(tr) = \mu_2(tr')$.
- $\xi_1(tr) = \xi_2(tr')$.

Definition 5. An abstract maximality labeled transition system $amlts = (\eta, \varphi, \mu, \xi, \theta)$ is a MLTS labeled by function φ on the alphabet $\mathbb{L} \cup \{\perp\}$. $\varphi : TR \longrightarrow \mathbb{L} \cup \{\perp\} / \perp$ denotes an undefined labeling.

Definition 6. An abstract maximality labeled transition system $amlts = \eta, \varphi, \mu, \xi, \theta)$ is said $\perp$-free if and only if $\forall tr \in TR : \varphi(tr) \neq \perp$. A $\perp$-free abstract maximality labeled transition system is a maximality labeled transition system.

Definition 7. The isomorphism on maximality labeled transition systems is extended to abstract maximality labeled transition system by extending the function φ to $\mathbb{L} \cup \{\perp\}$.

4 MAXIMALITY SEMANTICS FOR PETRI NET

In this section we recall the maximality approach of place transition Petri nets, proposed in [14, 16]. We introduce through simple example useful notations and functions for the definition of marking graph associated to a Petri net in a maximality-based approach.

Consider the example of the marked Petri net of Figure 5. With the launch of the transition t_1, it is clear that the firings of transitions t_2 and t_3 are conditioned by the end of the action related to t_1. To capture this causal dependence between firings of transitions, we consider that tokens produced by the firing of the transition t_1 are bound to this transition, namely the token in place p_2 and the token in place p_3 (Figure 6 b)). We can see that, in the initial state, the token in p_1 is not bound to any transition; this token is called free in this state, then the marked Petri net of Figure 6 a). In the case when t_2 would be fired, it could be argued that the action associated with the firing of t_1 has finished its execution. As a result, the token in p_3

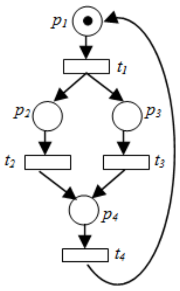


Figure 5. Marked Petri net

will become free. Resulting marking after the firing of the transition t_2 is given in Figure 6 c).

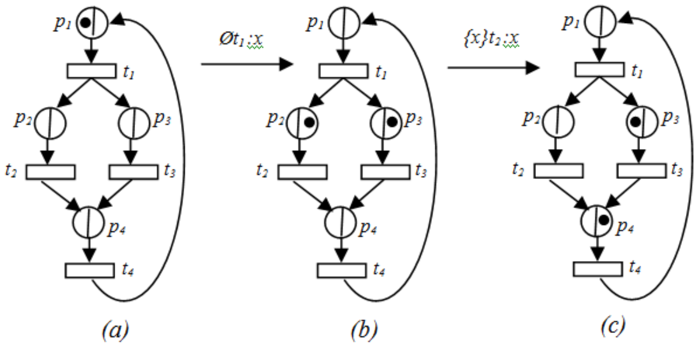


Figure 6. Free and bound tokens in a marking

To distinguish between free and bound tokens in a place, we can imagine that a place is composed of two separated parts. The left part contains free tokens while the right one will contain bound tokens. In a place, the number of free tokens will be denoted by $\mathcal{FT}$, while bound tokens set will be noted $\mathcal{BT}$. So each place is marked by $(\mathcal{FT}, \mathcal{BT})$. Hence, we obtain the succession of markings of Figure 6. Each bound token identifies an action that is eventually being executed (this token corresponds to a maximal event). Also each transition of marking graph corresponds to the start of execution of an action which is identified by an event name. Since a weight of an edge linking a transition to a place may be grater than one, a firing transition may produce more than one bound token, the bound token is identified by a tuple (n, t, x) where n is a number of instance of a bound token, t is a firing transition producing this bound token and x is an event name identifying the transition firing in time. Note that the firing condition of a transition is only conditioned by the number of free and bound tokens in places.

Preliminary Definitions

A Petri net is a tuple (P, T, W) where:

- P : is a finite set of places.
- T : is a finite set of transition such that $P \cap T = \emptyset$.
- $W : (P \times T) \cup (T \times P) \longrightarrow \mathbb{N}$ is the weight function.

Let (P, T, W) be a Petri net with a marking M:

- The set of maximal event names in M is the set of all event names identifying bound tokens in the marking M. Formally, the function δ will be used to calculate this set, it can be defined as: $\delta : M \longrightarrow PR(\mathcal{H})$. $\delta(M) = \cup_{p\in P}\{x_1, x_2, .., x_m\}$ such that $M(p) = (\mathcal{FT}, \mathcal{BT})$ with: $\mathcal{BT} = \{(n_1, t_1, x_1), \ldots, (n_m, t_m, x_m)\}$.
- Let $X \subset \mathcal{H}$ be a finite set of event names. The operation of transforming bound tokens defined by X to free tokens in the marking M is defined by the inductive function makefree as follows:
  - $\textit{makefree}(\{x_1, x_2, \ldots, x_n\}, M) = \textit{makefree}(\{x_2, \ldots, x_n\}, \textit{makefree}(\{x_1\}, M))$
  - $\textit{makefree}(\{x\}, M) = M'$ such that for all $p \in P$, if $M(p) = (\mathcal{FT}, \mathcal{BT})$ then:
    - If there is $(n, t, x) \in \mathcal{BT}$ then $M'(p) = (\mathcal{FT} + n, \mathcal{BT} - \{(n, t, x)\})$ (conversion of n bound tokens identified by the event name x to free tokens).
    - Otherwise, $M'(p) = M(p)$.
- $\mid M(p) \mid= \mathcal{FT} + \sum_{i=1}^{m} n_i$ such that $M(p) = (\mathcal{FT}, \mathcal{BT})$ with $\mathcal{BT} = \{(n_1, t_1, x_1), \ldots, (n_m, t_m, x_m)\}$.
- Let t be a transition of T; t is said to be enabled by the marking M iff $\mid M(p) \mid\geq W(p, t)$ for all $p \in P$. The set of all transitions enabled by the marking M will be noted *enabled*(M).
- The marking M is said minimal for the firing of the transition t iff $\mid M(p) \mid= W(p, t)$ for all $p \in P$.
- Let M_1 and M_2 be two markings of the Petri net (P, T, W). $M_1 \subseteq M_2$ iff $\forall p \in P$, if $M_1(p) = (\mathcal{FT}_1, \mathcal{BT}_1)$ and $M_2(p) = (\mathcal{FT}_2, \mathcal{BT}_2)$ then $\mathcal{FT}_1 \geq \mathcal{FT}_2$ and $\mathcal{BT}_1 \subseteq \mathcal{BT}_2$ such that the relation $\subseteq$ is extended to bound tokens sets as follows: $\mathcal{BT}_1 \subseteq \mathcal{BT}_2$ iff $\forall (n_1, t, x) \in \mathcal{BT}_1, \exists (n_2, t, x) \in \mathcal{BT}_2$ such that $n_1 \leq n_2$.
- Let M_1 and M_2 be two markings of the Petri net (P, T, W) such that $M_1 \subseteq M_2$. The difference $M_2 - M_1$ is a marking M_3 $(M_2 - M_1 = M_3)$ such that for all $p \in P$, if $M_1(p) = (\mathcal{FT}_1, \mathcal{BT}_1)$ and $M_2(p) = (\mathcal{FT}_2, \mathcal{BT}_2)$ then $M_3(p) = (\mathcal{FT}_3, \mathcal{BT}_3)$ with $\mathcal{FT}_3 = \mathcal{FT}_2 - \mathcal{FT}_1$ and $\forall (n_1, t, x) \in \mathcal{BT}_1$, $(n_2, t, x) \in \mathcal{BT}_2$, if $n_1 \neq n_2$ then $(n_2 - n_1, t, x) \in \mathcal{BT}_3$.
- $\textit{get} : 2^{\mathcal{H}} \longrightarrow \mathcal{H}/$ for any $\textit{elt} \in 2^{\mathcal{H}}$, $\textit{get}(\textit{elt}) \in \textit{elt}$ is the function which associates to any transition an events name.

- Given a marking M, a transition t and an event name $x \notin \delta(M)$, $occur(t, x, M) = M'$ such that for all $p \in P$, if $M(p) = (\mathcal{FT}, \mathcal{BT})$ then $M'(p) = (\mathcal{FT}, \mathcal{BT}')$ with $\mathcal{BT}' = \mathcal{BT} \cup \{W(t,p), t, x\}$ if $W(t,p) \neq 0$ and $\mathcal{BT}' = \mathcal{BT}$, otherwise. Hence, M' is the resulting marking obtained by the addition of bound tokens related to the firing of transition t to the marking M.
- $\lambda : T \longrightarrow \mathbb{L}$ is a function which associates to any transition an action name, such that $\mathbb{L}$ ranged over by a, b, ... In practice a transition label is a name of an action.

5 MAXIMALITY SEMANTICS FOR REFINABLE RECURSIVE PETRI NETS

All definitions remain valid for refinable recursive Petri net. Other notations and functions will be given for the definition of the operational maximality semantics. The proposed approach and the interest of hierarchical design are illustrated through simple examples. Since abstract transitions behaviors are introduced gradually, conflict, sequencing and parallelism relations linking an abstract transition to other transitions are extended to transitions of refinement Petri nets.

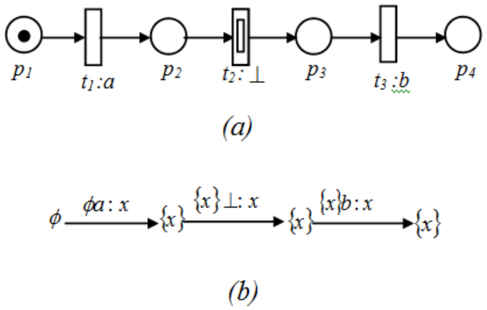


Figure 7. Start and end of undefined abstract transition

Consider the refinable recursive Petri net of Figure 7 a) in which t_2 is an abstract transition with an undefined behavior. The firing of this transition is caused by the end of the execution of the action associated to the transition t_1. Since the behavior of this transition is undefined, it will be fired as an elementary transition labeled by $\perp$. For this fact the generation of the marking graph for this net consists in the generation of the marking graph for classical Petri nets. Note that the event x identifies the beginning of the execution of an undefined process $\perp$.

Consider now the behavior of the process associated to the transition t_2 is modeled by the Petri net of Figure 8 a). The firing of this abstract transition starts the execution of its associated thread. The ordinary marking defined by the semilinear set will be prolonged to the marking of the instance of the Petri net son, this is interpreted by the addition of a token in the place p_5. The passing from the Petri

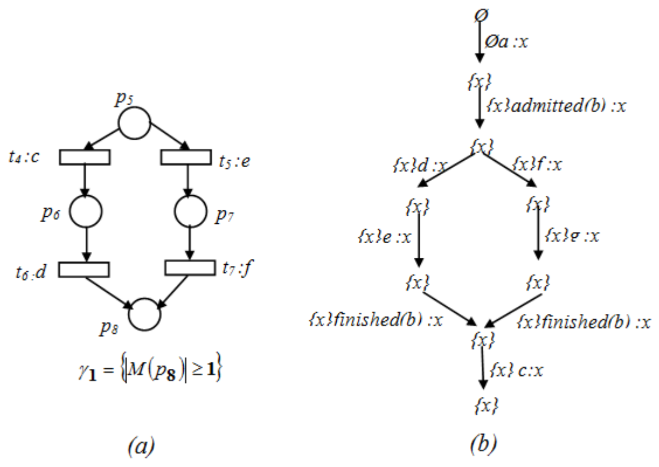


Figure 8. Refinement of the abstract transition t_2

net father to the Petri net son is made through the firing of a virtual transition called *admitted*. The firing of the transition *admitted* is causally dependent on the action a, the start of execution of the action *admitted*(b) is identified by the event x. This firing is similar to the firing of an elementary transition, it is followed by the deposition of a token related to this action in the right part of the place p_5.

After the generation of a linked token in the place p_5, any transition that can be fired from this thread will be immediately executed. But it is necessary to take into account the satisfaction of the predicate of termination$\gamma_1 = \{| \ M\left(p_8\right) \ |\geq 1\}$. This condition will be satisfied, when the transition t_6 or exclusively the transition t_7 deposits at least a token in the right part of the place p_8. When this predicate becomes true, a transition called *finished* will be fired, it makes the return to the father thread, indeed this transition represents the cut step of the son thread τ. Generally, a transition *finished* is regarded as an elementary transition. Its firing causes the emersion of the tokens defined by the post condition of the abstract transition in the right part of all places which belong to the post set of this one. Just after the end of the execution of abstract transition, the firing of the transition t_3 can happen. Figure 8 b) represents the maximality labeled transitions system generated from this Petri net. Note that the event x identifies the action *admitted*(b) as well as the start of the execution of the thread itself, thus it can be re-used within this thread. Once the thread is finished, this event name can be re-used in the father thread.

5.1 Comparison of Abstract MLTS

Definition 8. Let sys_1 and sys_2 be two systems such that: $[| \ sys_1 \ |]_{\{mlts\}}$ is not $\perp$-free and $[| \ sys_2 \ |]_{\{mlts\}}$ is not $\perp$-free. If $[| \ sys_1 \ |]_{\{mlts\}}$ and $[| \ sys_2 \ |]_{\{mlts\}}$ are abstractly isomorphs $\nRightarrow$ $[| \ \rho\left(sys_1\right) \ |]_{\{mlts\}}$ and $[| \ \rho\left(sys_2\right) \ |]_{\{mlts\}}$ are abstractly isomorphs such as: ρ is the process of refinement of an abstract transition $[||]_{\{mlts\}}$ is the process which interprets a Petri net to an abstract maximality labeled transition system.

A proposition of equality is said decidable if we can demonstrate this proposition or prove its negation. By definition two systems are equal if they have the same semantics representation. In the example of Figure 9, the two Petri nets seem that describe the same system, their amlts are abstractly isomorph (Figure 9 c)). But this does not mean that their refined systems stay equal.

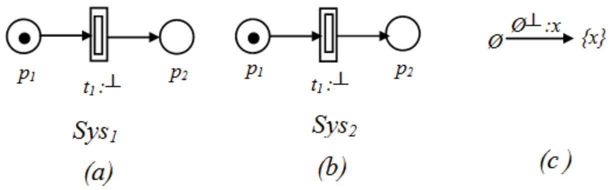


Figure 9. Comparison of two recursive Petri nets

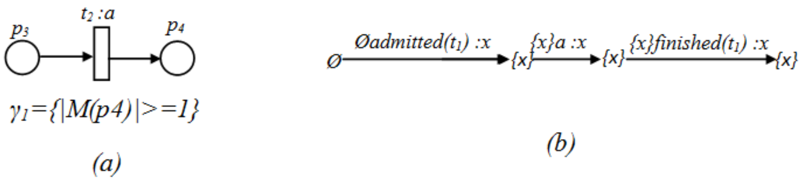


Figure 10. Case of equality after refinement

When we refine the two occurrences of the abstract transitions t_1 in sys_1, t_1 in sys_2 by the Petri net of Figure 10 a), we get two abstract isomorph maximality labeled transition systems given by the Figure 10 b). In this case, both systems are equal. However, if we refine the transition t_1 in sys_1 by the Petri net of Figure 11 a) and we refine the transition t_1 in sys_2 by the Petri net of Figure 11 b), we get two maximality labeled transition systems, which are not isomorphs at this level.

5.2 Maximality Operational Semantics for Refinable Recursive Petri Nets

Preliminary Definitions

- $\mathcal{THR}$: is the set of all threads.
- We call thread any configuration of the form: $\left(TH_i, (M_i)^{N_i}_{ref(T_i)}\right)$ such that:
  - M_i is the father marking.
  - TH_i is the set of the son threads where each thread is identified by a event name.
  - *ref* (T_i) is the Petri net corresponding to this thread, it describes the behavior of abstract transition T_i.

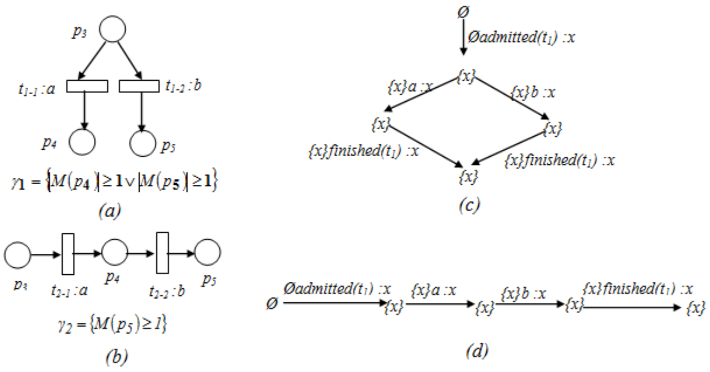


Figure 11. Case of inequality after refinement

– N_i is the event's name which identifies this thread.
– The initial configuration noted $(\emptyset, (M_0)_R^{x_0})$ is built from an initial marking M_0 of the principal Petri net R.

• Let the labeled refinable recursive Petri net $R = (P, T, I, W^-, W^+, \Omega, \gamma, K, \lambda)$ provided with a marking M:

– $\psi : \mathcal{THR} \to 2^{\mathcal{H}}$. The function which determines the events names in a thread is recursively defined by:

  * $\psi\left(\emptyset, (M)_R^N\right) = \delta(M)$.
  * $\psi\left(TH, (M)_R^N\right) = \left(\cup_{i=1}^n \psi(th_i)\right) \cup \delta(M)$ with $TH = \{th_1, th_2, .., th_n\}$.

– $\forall \left(TH, (M)_R^N\right) \in \mathcal{THR}$. $X \subset \mathcal{H}$ is a finished set of events names. $\mathit{clean}\left(X, \left(TH, (M)_R^N\right)\right)$ is recursively defined by:

  * $\mathit{clean}\left(X, \left(\emptyset, (M)_R^N\right)\right) = \left(\emptyset, (\mathit{makefree}(X, M))_R^N\right)$.
  * $\mathit{clean}\left(X, \left(TH, (M)_R^N\right)\right) = \left(\cup_{i=1}^n \mathit{clean}(X, th_i), (\mathit{makefree}(X, M))_R^N\right)$ with $TH = \{th_1, th_2, \ldots, th_n\}$.

– $\forall \left(TH, (M)_R^N\right) \in \mathcal{THR}$, $t \in T$ is sensitized by this thread if and only if:

  * $\forall p \in P :| M(p) | \geq w(p, t)$ or
  * $\exists th_i = \left(TH_i, (M_i)_{\mathit{ref}(T_i)}^{N_i}\right) \in TH$ such that: $| M_i(p) | \geq w(p, t)$ for all $p \in \mathit{ref}(T_i)(p)$.

– The function $\mathit{cutstep} : \mathcal{THR} \times \gamma \to \mathit{boolean}$ is defined as follows:

$$\forall th_i = \left(TH_i, (M_i)_{\mathit{ref}(T_i)}^{N_i}\right) \in \mathcal{THR}, \forall \gamma_i \in \gamma$$

then

$$\begin{cases} \mathit{true}, & \text{if } \forall p \in \mathit{ref}\,(T_i)\,(p)\,, M_i\,(p) \geq n \text{ with } \gamma_i = \{\mid M\,(p) \mid \geq n/n \in \mathbb{N}\}\,, \\ \mathit{false}, & \text{otherwise.} \end{cases}$$

- $TH \mid t > TH'$ means that the execution of the transition t from TH leads to TH'.
- A sequence $TH_0 t_1 TH_1 t_2 \dots$ is an occurrence sequence iff $TH_{i-1} \mid t_i > TH_i$ for $i \geq 1$. A sequence $\sigma = t_1 t_2 \dots$ is a transition sequence starting with TH_0 iff there is an occurrence sequence $TH_0 t_1 TH_1 t_2 \dots$. If a finite sequence $t_1 t_2 \dots t_n$ leads from TH to TH', we write $TH \mid t_1 t_2 \dots t_n > TH'$.

Semantics Rules

The operational semantics of labeled refinable recursive Petri net allowing the generation of a maximality labeled transitions system is defined by the following rules:

1. $\dfrac{\text{For } M_1 \text{ a marking}, t \in \mathit{enabled}\,(M_1) \text{ with } t \in T_{el} \cup T_{abi}}{\left(TH_1, (M_1)_R^N\right) \overset{E\lambda(t)_x}{\longrightarrow} \left(TH_1, (M_2)_R^N\right)}$ such that:
   $\forall M_3 \in \min\,(M_1, t)$:
   - $E = \delta\,(M_3)\,, M_4 = \mathit{makefree}\,(E, M_1 - M_3)$
   - $M_2 = \mathit{occur}\,(t, x, M_4)$ such that:
     - $\forall p \in P$ if $M_4\,(p) = (\mathcal{FT}_4, \mathcal{BT}_4)$ then: $M_2\,(p) = (\mathcal{FT}_4, \mathcal{BT}_2)$ with
     $$\mathcal{BT}_2 = \begin{cases} \mathcal{BT}_4 \cup \{(w\,(t,p,i)\,, \lambda\,(t)\,, x)\}\,, & \text{if } w\,(t,p,i) \neq 0, \\ \mathcal{BT}_4, & \text{otherwise.} \end{cases}$$
   - $x = \mathit{get}\left(\mathcal{H} - \left(\psi\left(\mathit{clean}\left(E, \left(TH_1, (M_1)_R^N\right)\right)\right)\right)\right)$.
2. $\dfrac{\text{For } M_1 \text{ a marking }, T_i \in \mathit{enabled}\,(M_1) \wedge T_i \in T_{abd}}{\left(TH_1, (M_1)_R^N\right) \overset{E\,\mathit{admitted}(\lambda(T_i))_x}{\longrightarrow} \left(TH_2, (M_2)_R^N\right)}$ such that:
   $\forall M_3 \in \mathit{min}\,(M_1, T_i)$:
   - $E = \delta\,(M_3)\,, M_4 = \mathit{makefree}\,(E, M_1 - M_3)$.
   - $\forall p \in P : M_2\,(p) = M_4\,(p)$.
   - $TH_2 = TH_1 \cup \left\{\left(\emptyset, (M_0)_{\mathit{ref}(T_i)}^{\{x\}}\right)\right\}$ such that:
   $$\left((M_0)_{\mathit{ref}(T_i)}^{\{x\}}\right)(p) = \begin{cases} (0, \{(\Omega\,(T_i)\,(p)\,, \mathit{admitted}\,(\lambda\,(T_i))\,, x)\})\,, & \text{if } \Omega\,(T_i)\,(p) \neq 0 \\ (0, \emptyset)\,, & \text{otherwise.} \end{cases}$$
   - $x = \mathit{get}\left(\mathcal{H} - \psi\left(\mathit{clean}\left(E, \left(TH_1, (M_1)_R^N\right)\right)\right)\right)$.

3. $\dfrac{th_i, \exists \gamma_i \in \gamma / cutstep\,(th_i, \gamma_i)}{\left(TH_1, (M_1)_R^N\right) \overset{\{x\} finished(\lambda(T_i))_x}{\longrightarrow} \left(TH_2, (M_2)_R^N\right)}$ such that:
$\forall th_i \in TH_1 / th_i = \left(TH_i, (M_i)_{ref(T_i)}^{N_i}\right), N_i = \{x\}$:

- $M_2 = occur\,(finished\,(T_i)\,, \{x\}\,, M_1)$.
- $\forall p \in P$: if $M_2\,(p) = (\mathcal{FT}_2, \mathcal{BT}_2)$ then

$$\mathcal{BT}_2 = \begin{cases} \mathcal{BT}_1 \cup \{(w\,(t,p,i)\,, finished\,(\lambda\,(T_i))\,, x)\}\,, & \text{if } w\,(t,p,i) \neq 0, \\ \mathcal{BT}_1, & \text{otherwise.} \end{cases}$$

- $TH_2 = TH_1 - \{th_i\}$.

4. $\dfrac{M_1, M_1 \in enabled\,(t)\,, t \in T_{el} \wedge K\,(t, T_i) \in I_p}{\left(TH_1, (M_1)_R^N\right) \overset{E\lambda(t)_x}{\longrightarrow} \left(TH_2, (M_2)_R^N\right)}$ such that:
$\forall th_i \in TH_1 / th_i = \left(TH_i, (M_i)_{ref(T_i)}^{N_i}\right)$, $\forall M_3 \in \min\,(M_1, t)$:

- $E = \delta\,(M_3)\,, M_4 = makefree\,(E, M_1 - M_3)$.
- $M_2 = occur\,(occur\,(t, E, M_3)\,, finished\,(\lambda\,(T_i))\,, N_i)$.
- $\forall p \in P : if M_2\,(p) = (\mathcal{FT}_2, \mathcal{BT}_1)$ then, $M_2\,(p) = (\mathcal{FT}_4, \mathcal{BT}_2)$ with:

$$\mathcal{BT}_2 = \begin{cases} \mathcal{BT}_4 \cup \{(w\,(t,p,i)\,, t, x)\}\,, & \text{if } w\,(t,p,i) \neq 0, \\ \mathcal{BT}_4 \cup \{(w\,(T_i,p,i)\,, finished\,(\lambda\,(T_i))\,, N_i)\}\,, & \text{if } w\,(t,p,i) \neq 0, \\ \mathcal{BT}_4, & \text{otherwise.} \end{cases}$$

- $TH_2 = TH_1 - \{th_i\}$.
- $x = get\left(\mathcal{H} - \left(\psi\left(clean\left(E, \left(TH_1, (M_1)_R^N\right)\right)\right) - \psi\,(th_i)\right)\right)$.

6 CASE STUDY

As an example, we consider a flame cutting machine used to fabricate pieces for vehicles: this machine consists of: the reading head, the blowtorches and the template which is a diagram dimensioning pieces. It has a movable table on which the template is located, and then transmits it as a dimensional information. The read information is transmitted to blowtorches controller, as a result the blowtorches cut the piece from metal sheet according to that as defined on the template. Because some external events may cause errors (modification of table position) during the cutting process, in such a case the following actions are immediately produced:

- Displaying output indicating the occurrence of a problem using a lamp.
- The blowtorches will be stopped.
- The table is reported in its initial position.

The assessment of this system is done by verifying some properties, expressed in CTL logic, using the formal verification environment FOCOVE (Formal Concurrent Verification Environment).

Step 01: In this first step we abstractly model the system tasks. Consequently tasks considering behaviors are not yet known. In the specification of Figure 12 a), transitions t_2 and t_3 are associated respectively to fabricate and maintain processes. These transitions are labeled by "bottom" since their behaviors are unknown.

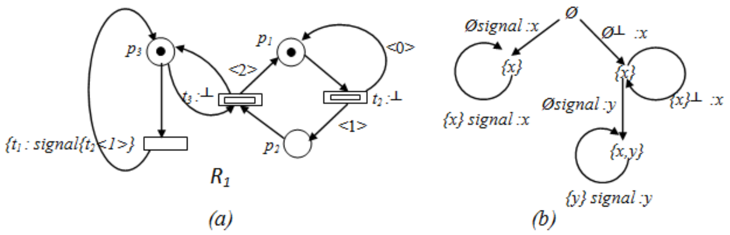


Figure 12. Modelling of flame cutting machine in step 01

The abstract maximality labeled transition system corresponding to this refinable recursive Petri net is shown by Figure 12 b).

Verification

Using the CTL logic in the context of the maximality semantics where actions represent activities has been studied in [16]. Actions are considered as atomic propositions. Then an action name a in a formula associated to a given state means that action a may be in execution at this state. At this level of abstraction we can, for example, verify that always after each execution of the undefined process related to the transition *fabricate* we can launch this process again. This property is expressed in CTL logic as follows: $AG(\perp => \perp)$.

Following the semantics of CTL formula in the context of the maximality semantics, the formula $AG(\perp => \perp)$ means that a not well known activity at a given state leads to the execution of this activity again. Following the refinement process, this activity will be specified in the following refinement steps.

Step 02: In this step we label the abstract transition t_2 by the action *fabricate*, so now we will give details of this abstract transition. This is done by refining the transition t_2 in Petri net R_1 by the process described in Petri net R_2. $\rho\left(t_2, R_1, R_2, \{0\}, \{1\}, \langle p_4\rangle, \{\gamma_0 = \{M/ \mid M(p_6) \mid\geqslant 1\}\}\right) = R_3$

The abstract maximality labeled transition system of this refinable recursive Petri net is shown by Figure 14.

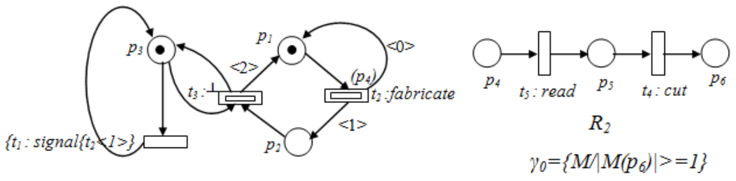


Figure 13. Modelling of flame cutting machine in step 02

Verification

- After each execution of process "fabricate" we can again lunch it.

$$AG\,(finished\,(fabricate) \Longrightarrow admitted\,(fabricate))\,.$$

- When a problem of cutting appears, the undefined process $\perp$ corresponding to the transition maintain will be automatically launched.

$$AF\,(signaler \Longrightarrow EX\perp)\,.$$

- When a problem of cutting appears, the process fabricate will be preempted.

$$AG\,(signaler \Longrightarrow finished\,(fabricate))\,.$$

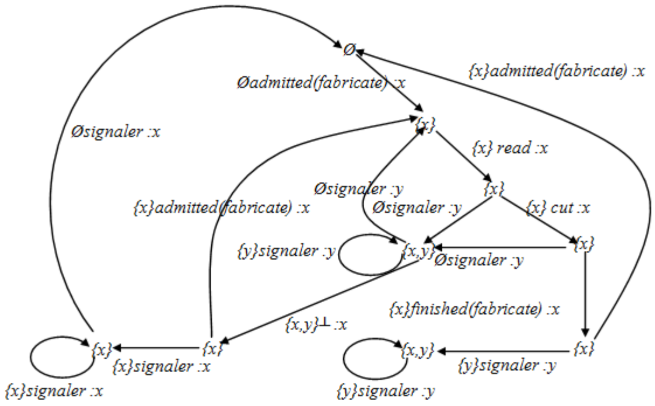


Figure 14. Abstract maximality labeled transition system in step 2

Step 03: Now we give details of the process corresponding to the abstract transition t_3, it will be labeled by the action *maintain*.

$$\rho\,(t_3, R_3, R_4, \{2\}\,, \emptyset, \langle p_7\rangle\,, \{\gamma_2 = \{M/ \mid M\,(p_9) \mid \geqslant 1\}\}) = R_5.$$

The maximality labeled transition system obtained by applying the proposed approach consists of 12 states and 19 transitions. Due to its size it cannot be depicted in this paper.

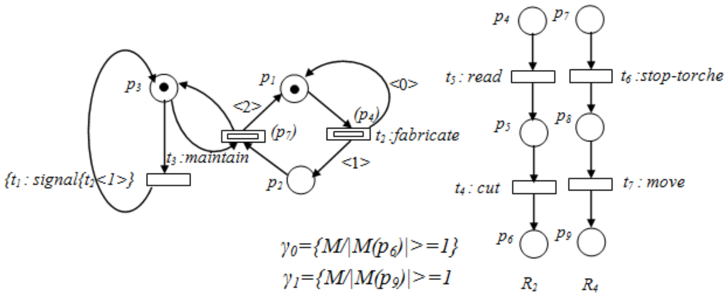


Figure 15. Modelling of flame cutting machine in step 03

Verification

All proprieties which are verified in step 2 are still to be verified in this step 3. In addition we can verify other properties like:

- When an error will occur while the process *fabricate* is running the flame cutting will be stopped.

$$AG\,((signaler\ \ and\ \ finished\,(fabricate)) \Longrightarrow EG\ \ stop\text{-}torche)$$

7 CONCLUSION

In this paper, we have proposed a new approach to modelling concurrent systems by defining a new model named refinable recursive Petri nets, which permits a hierarchical design, so the functionalities and features of systems can be added gradually. Also we have proposed an operational method for generating a maximality labeled transition system associated to the refinable recursive Petri nets. This will make it possible to benefit from the developed results of verification around the model of maximality labeled transition systems. For this fact, the properties related to the good performance of a system specified by a refinable recursive Petri net can be checked on its corresponding maximality labeled transition system. It should be noted that the structure of the maximality labeled transition system represents, in a natural way, the parallel execution of actions, as well as the parallel execution of threads.

REFERENCES

[1] Aceto, L.—Hennessy, M.: Adding Action Refinement to Finite Process Algebra. In: Albert, J. L., Monien, B., Artalejo, M. R. (Eds.): Automata, Languages and Programming (ICALP '91). Springer, Lecture Notes in Computer Science, Vol. 510, 1991, pp. 506–519.

[2] Andrews, D.—Groote, J.—Middelburg, C. (Eds.): Semantics of Specification Languages (SoSL). Springer, London, Workshops in Computing, 1993.

[3] Best, E.—Devillers, R.—Kiehn, A.—Pomello, L.: Concurrent Bisimulations in Petri Nets. Acta Informatica, Vol. 28, 1991, pp. 231–264, doi: 10.1007/BF01178506.

[4] Boudol, G.—Castellani, I.: Concurrency and Atomicity. Theoretical Computer Science, Vol. 59, 1988, No. 1-2, pp. 25–84, doi: 10.1016/0304-3975(88)90096-5.

[5] Courtiat, J. P.—Saidouni, D. E.: Action Refinement in LOTOS. In: Danthine, A., Leduc, G., Wolpe, P. (Eds.): Protocol Specification, Testing and Verification (PSTV '93). North-Holland, 1994, pp. 341–354.

[6] Darondeau, P.—Degano, P.: Causal Trees. In: Ausiello, G., Dezani-Ciancaglini, M., Della Rocca, S. R. (Eds.): Automata, Languages and Programming (ICALP '89). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 372, 1989, pp. 234–248.

[7] Degano, P.—Gorrieri, R.: Atomic Refinement in Process Description Languages. In: Tarlecki, A. (Ed.): Mathematical Foundations of Computer Science (MFCS 1991). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 520, 1991, pp. 121–130.

[8] Devillers, R.: Maximality Preservation and the ST-Idea for Action Refinement. In: Rozenberg, G. (Ed.): Advances in Petri Nets. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 609, 1992, pp. 108–151, doi: 10.1007/3-540-55610-9_170.

[9] Courtiat, J.-P.—Saïdouni, J.-E.: Relating Maximality-Based Semantics to Action Refinement in Process Algebras. In: Hogrefe, D., Leue, S. (Eds.): IFIP TC6/WG6.1, 7th International Conference on Formal Description Techniques (FORTE '94), Chapman, Hall, IFIP Conference Proceedings 6, 1994, pp. 293–308.

[10] Janssen, W.—Poel, M.—Zwiers, J.: Action Systems and Action Refinement in the Development of Parallel Systems. In: Baeten, J. C. M., Groote, J. F. (Eds.): CONCUR '91. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 527, 1991, pp. 298–316.

[11] Saidouni, D. E.—Courtiat, J. P.: Syntactic Action Refinement in Presence of Multiway Synchronization. Semantics of Specification Languages (SoSL), 1994, pp. 289–330, doi: 10.1007/978-1-4471-3229-5_16.

[12] Van Glabbeek, R. J.: The Refinement Theorem for ST-Bisimulation Semantics. IFIP Working Conference on Programming Concepts and Methods, North-Holland, 1990.

[13] Saidouni, D. E.: Maximality Semantic: Application to Actions Refinement in LOTOS. Ph.D. thesis, LAAS-CNRS, 7 av. du Colonel Roche, 31077 Toulouse Cedex France, 1996 (in French).

[14] Saidouni, D. E.—Belala, N.—Bouneb, M.—Boudjadar, A.—Ouchène, B.: Using Maximality-Based Labeled Transitions as Model for Petri Nets. The International Arab Conference on Information Technology (ACIT '08), 2008.

[15] Saidouni, D. E.—Belala, N.—Bouneb, M.: Aggregation of Transitions in Marking Graph Generation Based on Maximality Semantics for Petri Nets. Proceedings

of the Second International Conference on Verification and Evaluation of Computer and Communication Systems (VECoS '08), 2008, pp. 6–16.

[16] Saidouni, D. E.—Belala, N.—Bouneb, M.: Using Maximality-Based Labelled Transitions as Model for Petri Nets. The International Arab Journal of Information Technology (IAJIT), Vol. 6, 2009, No. 5, pp. 440–446.

[17] Saidouni, D. E.—Belala, N.—Bouneb, M.: Maximality-Based Structural Operational Semantics for Petri Nets. 2nd Mediterranean Conference on Intelligent Systems and Automation (CISA), 2009.

[18] Saidouni, D. E.—Bouneb, M.—Ilie, J. M.: Maximality Semantic for Recursive Petri Net. Proceedings of 27th Europeen Conference on Modelling and Simulation (ECMS '13). 2013, pp. 544–550. ISBN 978-0-9564944-7-4, doi: 10.7148/2013-0544.

[19] Buchholz, P.: Hierarchical High Level Petri Nets for Complex System Analysis. Computer Science IV, Dortmund university, D-44221, Dortmund Germany, 1994, doi: 10.1007/3-540-58152-9_8.

[20] Cheung, K.-S.—Chow, P. K.-O.: A Petri-Net Approach to Refining Object Behavioural Specifications. Informatica, Vol. 33, 2009, No. 2, pp. 221–232.

Messaouda Bouneb received her B.Eng. degree from the University of Mentouri Constantine, Algeria (2005). In February 2009, she received her M.Sc. degree in computer science from the University of El Arbi Ben-M'hidi Oum El-Bouaghi, Algeria. Her research domain is formal specification and verification of real-time systems using Petri nets.

Djamel Eddine Saidouni received his B.Eng. degree from the University of Mentouri Constantine, Algeria (1990). He received his Ph.D. in theoretical computer science from the University of Paul Sabatier, Toulouse, France (1996). His domain research is the formal specification and verification of complex distributed and real time systems.

Jean Michel Ilie received several degrees in electronics and informatics along with his Ph.D. thesis from the UPMC University of Paris (1990). Currently, he is a member of the Paris Descartes University in its conference on a master higher grade (2009), he is also Permanent Researcher of the LIP6 laboratory, UPMC. The fields of his research concern the formal validation of complex embedded systems.

Computing and Informatics, Vol. 37, 2018, 656–672, doi: 10.4149/cai_2018_3_656

BREAKOUT LOCAL SEARCH FOR THE TRAVELLING SALESMAN PROBLEM

Mehdi El Krari

Faculty of Science, Mohammed V University in Rabat
Computer Science Laboratory
Rabat, Morocco
e-mail: mehdi@elkrari.com

Belaïd Ahiod

Faculty of Science, Mohammed V University in Rabat
LRIT, Associated Unit to CNRST (URAC 29)
Rabat, Morocco
e-mail: ahiod@fsr.ac.ma

Bouazza El Benani

Faculty of Science, Mohammed V University in Rabat
Computer Science Laboratory
Rabat, Morocco
e-mail: elbenani@hotmail.com

Abstract. The travelling salesman problem (TSP), a famous NP-hard combinatorial optimisation problem (COP), consists of finding a minimum length tour that visits n cities exactly once and comes back to the starting city. This paper presents a resolution of the TSP using the breakout local search metaheuristic algorithm (BLS), which is based on the iterated local search (ILS) framework and improves it by introducing some fundamental features of several well-established metaheuristics such as tabu search (TS) and variable neighbourhood search (VNS). BLS moves from a local optimum of a neighbourhood to another by applying perturbation jumps whose type and number are determined adaptively. It has already been applied to many COP and gives good results. This innovative hybridisation resolved well 41 instances from the commonly used benchmark library TSPLIB. The high quality of experimental results shows the competitiveness of the proposed algorithm compared to other algorithms based on local search.

Keywords: Travelling salesman problem, breakout local search, adaptive perturbation strategy, iterated local search

1 INTRODUCTION

The travelling salesman problem (TSP) [1, 3] is one of the most universally studied combinatorial optimisation problems (COP). It is for more than half a century the focus of many researchers from all around the world. Work on the TSP had an enormous impact on the emergence and evolution of many important areas of research (stochastic local search [9], integer programming [22], complexity theory [11] ...). Besides its importance in practice, the TSP has also become a standard testbed for new algorithmic ideas. The problem was introduced for the first time in 1859 by William Rowan Hamilton. In its classic form, the statement is as follows: "A travelling salesman must visit once and only once a finite number of cities and return to its point of origin. Find the order of visiting cities that minimises the total distance travelled by the salesman."

Application domains of TSP are numerous: logistic problems of transportation of goods, as well as people, and more generally all kinds of scheduling problems. Some issues in the industry are modelled as a travelling salesman problem as the optimisation of trajectories of machine tools: How to drill several points on an electronic card as quickly as possible? The manufacturing of VLSI chips [23] and X-ray crystallography [21] are just a few examples of several applications of the TSP. Its simplicity and adaptability have made this problem for decades a starting point for more work and research. This COP belongs to NP-complete problems [11].

We introduce the breakout local search metaheuristic algorithm (BLS) for solving the TSP. While iterated local search (ILS) [19] may suffer from lack of effectiveness in escaping attractions, BLS follows the basic scheme of this framework and improves it by combining the features of other robust and efficient methods, including variable neighbourhood search (VNS) [15] adapted on perturbations [8]. The main idea of BLS is to use a descent-based local search to find local optima, and use the most appropriate perturbations in order to move (without being blocked) from one neighbourhood to another in the search space. Perturbation strategy of BLS is based on both the history and state of search; it introduces a variable degree of diversification by determining perturbation dynamically jumps and performing adaptive selection from several types of dedicated movements.

BLS was developed by Benlic and Hao in 2012. Since then, it has been used for solving some COP, such as the minimum sum coloring problem [4], maximum clique problems [5], quadratic assignment problem [7] and max-cut problem [6] and has given very good results. This paper deals with a resolution of the TSP, whose performance will be evaluated by solving 41 benchmark instances of the TSPLIB [16].

The reminder of this paper is organised as follows: Section 2 introduces the TSP and local search approaches. Section 3 describes in detail the BLS metaheuristic.

Section 4 first reports the computational results and comparisons, which are based on the TSPLIB benchmark instances; then it provides justification of the choice for some of BLS parameter settings. Section 5 is devoted to a discussion around BLS highlights which makes it different from other ILS algorithms. Finally, Section 6 concludes the paper.

2 THE TRAVELLING SALESMAN PROBLEM

2.1 TSP Formulation and Landscape Analysis

Given n cities and a matrix $D = (d_{ij})_{n\times n}$ of distances between all pairs of these cities. The TSP aims to find a shortest closed tour (i.e. Hamiltonian cycle) in which each city is visited once and only once. Each tour can be represented by a permutation $\pi = (\pi(1), \pi(2), \dots, \pi(n))$ of integers from 1 to n where $j = \pi(i)$ denotes the city j to visit at step i, $i = 1, 2, \dots, n$. Therefore, the goal of TSP is to search a permutation π (tour) that minimises the tour length given by the following equation:

$$\sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \tag{1}$$

where d_{ij} is the distance between city i and city j. In this paper, we consider the symmetric TSP where the distances satisfy $d_{ij} = d_{ji}$ for $1 \leq i, j \leq n$.

The study of TSP landscape [2] (or for any COP) allows us to know which operator exploits better the search space of the problem. Changing neighbourhood must be made such that it gives a new search area without escaping too far from the previous neighbourhood, in which case would provide an independent neighbourhood from those before. Such a study requires knowledge of some notions such as the fitness function and correlation.

As mentioned earlier, the most common fitness function is that using the length of the tour. A good fitness is a fitness with a low value, so a shorter tour. The landscape of the problem depends on the fitness function of the different solutions, hence the concept of fitness landscape.

The second important concept to know is that of correlation [14, 18], introduced to provide a measure of the difficulty of a problem taking in account some operators. It shows how much a tour is linked to its neighbours.

Experiments done previously on the TSP [10, 2, 18] showed that the landscape is more correlated for 2-opt move than others. That is why it has been chosen in this paper, especially in the local search phase.

2.2 Local Search Approaches for the TSP

Local search methods start from an initial configuration and apply successive transformations to the current solution while a stopping criterion is not verified. Therefore, the implementation requires the choice of an initial solution(s) and local transformation(s), also known as moves. These algorithms are frequently used for solving the TSP problem. They improve iteratively the current solution seeking better in its predefined neighbourhood. The algorithm stops when he reaches a maximal number of iterations or when there is no better solution in a given neighbourhood: a local minimum is attained. Historically, 2-opt [9] is one of the first algorithms to solve instances of the TSP. It is a local search algorithm whose neighbourhood is defined by removing two non-adjacent edges of the current solution. The two parts obtained from this solution are reconnected by two other edges to obtain a new solution. The iterated local search (ILS) proposed by T. Stützle [19] is a framework based on local searches; it is a stochastic method that produces a sequence of solutions generated by an introduced heuristic, leading to better results than if we use repeated random testing of this heuristic. One of the main steps of ILS is perturbations [8], it is to avoid to be trapped in a local optimum by switching to another more distant. Local search in such a case will run more easily and will be more efficient.

However, perturbations applied by ILS for some TSP instances may not escape attraction from some neighbourhoods. Based on the later point, we introduced a new approach based on diversification of perturbations and moves, named "Breakout Local Search".

3 BREAKOUT LOCAL SEARCH FOR THE TSP

3.1 An Overview of BLS

The overall approach of BLS is a move from a local optimum of a neighbourhood to another one by applying perturbation jumps type and number of which are determined adaptively. Algorithm 1 below is a pseudo-code of the BLS algorithm for solving the TSP. BLS starts from an initial solution π_0 (having a cost C_0) and performs local search (the steepest descent) to reach a new local optimum π (lines 13–18). Each iteration of the local search algorithm browses the whole neighbourhood and chooses the best improving solution to replace the current neighbourhood solution. If no improvement is made in the neighbourhood, the local optimality is reached. BLS tries firstly to escape the neighbourhood attraction of the current local optimum to move to a new neighbourhood attraction (line 38). BLS applies then a number of moves (or jumps) starting from L_0 and dedicated to the current local optimum π which becomes disturbed and serves as a starting point for the next descent of the local search procedure. When the local search procedure returns the same neighbour π, BLS disturbs it more strongly by selecting a stronger perturbation, and (when different perturbations are not able to move to a neighbourhood with a new best local optimum) by increasing the number of jumps L by 1 to apply for this perturbation (lines 30–32). After visiting some neighbourhoods without improving the best solution found so far (lines 23–24) T times, BLS performs a much stronger perturbation with L_{max} jumps to permanently direct search into a new and more distant region in the search space (lines 26–30).

Algorithm 1 Breakout Local Search for TSP

```
Require: Maximal descents to perform $Desc_{max}$, initial number of jumps $L_0$, maximal
    consecutive visited local optima without any improvement $T$, number of jumps in
    strong perturbation $L_{max}$.
Ensure: A solution $\pi_{best}$
 1: $S \leftarrow 0$
 2: $\pi \leftarrow initialSolution()$      /* generates a solution with greedy or random algorithm */
 3: $C \leftarrow Cost(\pi)$
 4: $\pi_{best} \leftarrow \pi$      /* $\pi_{best}$ saves best solution found */
 5: $c_{best} \leftarrow c$      /* $c_{best}$ saves best objective value */
 6: $\omega \leftarrow 0$      /* $\omega$ gives the number of consecutive non-improving local optima */
 7: $L \leftarrow L_0$      /* $L$ saves number of jumps to perform, set to its minimal value $L_0$ */
 8: $L\omega \leftarrow 0$      /* $L\omega$ is an indicator to guess which move to use in next perturbation */
 9: $c_p \leftarrow c$      /* $c_p$ saves objective value of last descent */
10: $Desc \leftarrow 0$      /* $Desc$ saves current number of descents */
11: $Iter \leftarrow 0$      /* global iteration counter */
12: while $Desc < Desc_{max}$ do
13:     while $\exists 2optMove(x,y)$ such that $(c + delta2Opt(\pi,x,y) < c)$ do
14:         $\pi \leftarrow \pi \oplus 2optMove(x,y)$      /* perform the best improving move */
15:         $c \leftarrow c + delta2Opt(\pi,x,y)$      /* cost variation of $\pi$ with (x,y) move */
16:         $update\_H(Iter,x,y)$      /* update iteration number when edges move was last performed */
17:         $Iter \leftarrow Iter + 1$
18:     end while
19:     if $c < c_{best}$ then
20:         update $\pi_{best}$ and $c_{best}$
21:         $\omega \leftarrow 0$
22:         $Desc \leftarrow Desc \times \frac{1}{2}$      /* reduce current number of descents */
23:     else
24:         $\omega \leftarrow \omega + 1$
25:     end if
26:     if $\omega > T$ then      /* performing strong perturbation */
27:         $\pi \leftarrow dbmPerturb(\pi, L_{max})$      /* Double Bridge Move perturbation with $L_{max}$ moves */
28:         $\omega \leftarrow 0$
29:         $Desc \leftarrow Desc \times \frac{7}{8}$      /* reduce current number of descents */
30:     else if $c = c_p$ then      /* search returned the previous local optimum */
31:         $L\omega \leftarrow L\omega + 1$      /* increment indicator for the type of perturbation */
32:         $L \leftarrow L_0 + \frac{L\omega}{3}$      /* increment moves if the 3 moves do not improve */
33:     else      /* Search escaped from the previous local optimum, reinitialize indicator */
34:         $L\omega \leftarrow 0$
35:     end if
```

```
36:     $c_p \leftarrow c$     /* update the objective value of the previous local optimum */
37:     if Strong perturbation was not performed then
38:         $\pi \leftarrow Adaptive\_Perturbation(\pi, L, L\omega, H, Iter, \omega)$     /* see algorithm 4 */
39:     end if
40: end while
41: return $\pi_{best}$
```

3.2 Exploring Solution Space by Neighbourhood

BLS is a metaheuristic based on the ILS framework, the process of descent/perturbation is redone as we have not yet reached a number ($Desc_{max}$) of descents. BLS uses the steepest descent with a 2-opt neighbourhood in local search and is called different perturbations, each of them introduces a different neighbourhood as shown in Algorithm 2. One of these perturbations (depending on $L\omega$ value) is applied L times to a local optimum π with moves chosen from the set of candidates M.

Algorithm 2 Dynamic_Perturbation($\pi, L, L\omega, H, Iter, \omega, M$)

Require: Initial solution π which is a local optimum, number of jumps L, indicator of perturbation type $L\omega$, matrix of history moves H, global iteration counter $Iter$, number of consecutive non-improving local optima ω, set of candidate moves M.

Ensure: A perturbed solution π

```
1: if ($L\omega$ mod 3) = 0 then     /* Call Perturbation_Operator with 2Opt move */
2:     $\pi \leftarrow Perturbation\_Operator(\pi, L, H, Iter, \omega, M, 2Opt)$
3: else if ($L\omega$ mod 3)= 1 then     /* Call Perturbation_Operator with insert move */
4:     $\pi \leftarrow Perturbation\_Operator(\pi, L, H, Iter, \omega, M, insert)$
5: else     /* Call Perturbation_Operator with swap move */
6:     $\pi \leftarrow Perturbation\_Operator(\pi, L, H, Iter, \omega, M, swap)$
7: end if
8: return $\pi$
```

Perturbations play a key role in BLS since the steepest descent cannot escape the local optimum. BLS tries then to exit the current neighbourhood by introducing different parameterised perturbations, starting with

1. the nodes/cities to move, and
2. how many times to perturb and
3. which move type to make.

All these steps will be detailed in the next sections.

3.3 Principle of Adaptive Perturbation

3.3.1 Main Idea

A variety of perturbations in BLS resides in the number of jumps and the (three) types of perturbations implemented. The number of jumps increases once all the perturbations attempted without any improvement.

- The first perturbation is performed with a 2-opt move, wich is characterised by an exchange of two non-adjacent edges as it is shown in Algorithm 3.
- Secondly we perform an insert move. It comes to move a node from one position to another. This move results in a change of three edges.
- Finally a swap move. It comes to move an edge from one position to another. This move results in a change of four edges.

We present below in Algorithm 3 a perturbation launched by the 2-opt move. The same algorithm is adapted to *insert* and *swap* moves, by applying the appropriate move and saving the affected edges.

Algorithm 3 Perturbation_Operator$(\pi, L, H, Iter, \omega, M, mvt)$

```
Require: Initial solution $\pi$ which is a local optimum, number of jumps $L$, matrix of
    history moves $H$, global iteration counter $Iter$, number of consecutive non-improving
    local optima $\omega$, set of candidate moves $M$, move type $mvt$.
Ensure: A perturbed solution $\pi$
 1: for $i \leftarrow 1, L$ do
 2:     take a pair $(x, y) \in M$
 3:     if $mvt = 2Opt$ then
 4:         $\pi \leftarrow \pi \oplus 2OptMove(x, y)$
 5:         $c \leftarrow c + delta2Opt(\pi, x, y)$
 6:     else if $mvt = insert$ then
 7:         $\pi \leftarrow \pi \oplus insertMove(x, y)$
 8:         $c \leftarrow c + deltaInsert(\pi, x, y)$
 9:     else
10:         $\pi \leftarrow \pi \oplus swapMove(x, y)$
11:         $c \leftarrow c + deltaSwap(\pi, x, y)$
12:     end if
13:     $update\_H(Iter, x, y)$
14:     $Iter \leftarrow Iter + 1$
15:     if $c < c_{best}$ then
16:         update $\pi_{best}$ and $c_{best}$
17:         $\omega \leftarrow 0$
18:         $Desc \leftarrow Desc \times \frac{1}{2}$
19:     end if
20: end for
21: return $\pi$
```

Rather than performing random jumps all the time, BLS switches between three types of perturbations: directed, recency-based and random. Each perturbation generates, as shown in Algorithm 4, a set M of pairs that will be used in perturbations.

3.3.2 The Three Types of Perturbation Moves

The directed perturbation aims to build a set of candidates with the lowest degradation during perturbation move. These candidates should not have been solicited in the last γ moves: they are saved in a tabu list [12, 13], with the corresponding length γ. However an edge can be part of the tabu list but still selected if it leads to a solution that improves the best solution found so far. Directed perturbation is built based on the tabu list, and the quality of the moves to be applied. Eligible candidates for this perturbation are defined by the following set A:

$$\begin{aligned} A = \{&\text{2OptMove}(u, v) \mid \min\{\text{delta2Opt}(\pi, u, v)\}, \\ &(H_{u,v} + \gamma) < \text{Iter} \vee (\text{delta2Opt}(\pi, u, v) + c) < c_{best}, u \neq v\}. \end{aligned} \tag{2}$$

The recency-based perturbation builds a set of candidates by using only the matrix H of historical moves. The moves are those which have been least recently used. These moves are identified by the set B such as:

$$B = \{\text{2OptMove}(u, v) \mid \min\{H_{uv}\}, u \neq v\}. \tag{3}$$

Finally, **the random perturbation** simply makes moves that are picked uniformly at random. Those moves are identified as:

$$C = \{\text{2OptMove}(u, v), u \neq v\}. \tag{4}$$

The three above formula adapt to both insert and swap movements, by applying the appropriate movement and delta operations.

Depending on the state of search, BLS selects one of these three perturbations pseudo-randomly with a probability. This state is determined by the parameter ω that gives the number of consecutive non-improving local minima. The aim is to give priority to the directed perturbation at the beginning of the search (when ω is still small), and reduce the chances of running when the neighbourhood has important attraction, to use other perturbations and have stronger diversifications.

We force the probability P of applying the directed perturbation to get values no smaller than a threshold P_0:

$$P = \begin{cases} e^{-\omega/T}, & \text{if } (e^{-\omega/T} > P_0), \\ P_0, & \text{otherwise.} \end{cases} \tag{5}$$

Given the probability P of using the directed perturbation, the probability of applying both the recency-based and the random perturbations is determined respectively by $(1 - P) \times Q$ and $(1 - P) \times (1 - Q)$ where Q is a constant from $[0, 1]$. Algorithm 4 links the three probabilities to their respective perturbation.

Algorithm 4 Adaptive_Perturbation$(\pi, L, L\omega, H, Iter, \omega)$

```
Require: A tour $\pi$ which is a local optimum, number of jumps $L$, determinant of perturbation type $L\omega$, matrix of history moves $H$, global iteration counter $Iter$, number of consecutive non-improving local optima $\omega$.
Ensure: A perturbed solution $\pi$
1: Determine probability $P$ according to Formula (5)    /* section above */
2: with a probability $P$,    /* directed perturbation */
3: $\pi \leftarrow Dynamic\_Perturbation(\pi, L, L\omega, H, Iter, \omega, A)$
4: with a probability $(1-P) \times Q$,    /* recency-based perturbation */
5: $\pi \leftarrow Dynamic\_Perturbation(\pi, L, L\omega, H, Iter, \omega, B)$
6: with a probability $(1-P) \times (1-Q)$,    /* random perturbation */
7: $\pi \leftarrow Dynamic\_Perturbation(\pi, L, L\omega, H, Iter, \omega, C)$
8: return $\pi$
```

3.3.3 Variation Jumps and Perturbation Moves

BLS varies (between two consecutive blocks) the number of jumps and performed moves. These changes are executed in a particular order: the first change performed is *2-opt* move, characterised by the exchange of two edges, the second change tried is *insert* move with three edges exchanged and finally the *swap* move defined by an exchange of four edges. The main idea is to move to the nearest neighbourhood and allow, at the same time, to escape the local optimum attractor and find a new and better solution. In the case where the three changes fail to escape from the optimum attractor, the number of jumps L is incremented.

The worst case is to redo the perturbation T times (consecutively) without being able to leave this attractor. A strong perturbation defined by L_{max} jumps with a Double Bridge [20] move is then performed. The current number of descents ($Desc$) is then reduced by $\frac{1}{8}$ in order to give a chance to the new neighbourhood to be enough scanned.

4 EXPERIMENTAL RESULTS

4.1 Experimental Protocol

BLS algorithm is programmed in Java 1.7, and compiled on a Pentium Dual-Core CPU T4400 with 2.20 GHz and 2.8 GB. 41 instances from the commonly used TSPLIB benchmark are considered in the experiments, their sizes range from 14 to 442 cities. Each instance is run 20 times, with the parameters listed in Table 1. The choice of parameter values was carried out after many preliminary tests. This is justified in Subsection 4.3.

We observed (in the worst cases) that BLS may never reach stopping conditions if maximum number of non-improving attractors visited T before strong perturbation is set to a small value: this deadlock is due to the repetitive reduction of current number of runs ($Desc$) which prevents reaching $Desc_{max}$. T must be greater than

Parameter	Value	Meaning
$Desc_{max}$	$50n$, $25n$	Maximal number of descents. $50n$ if $n < 200$, $25n$ otherwise
L_0	1	Initial jump magnitude
L_{max}	$0.5n$	Jump magnitude during strong perturbation
γ	n	Tabu tenure/length
P_0	0.75	Smallest probability to perform directed perturbation
Q	0.7	Probability to perform random over recency-based perturbation
T	$((Desc_{max} - 1)/8) + 1$	Maximal number of consecutive non-improving local minima

Table 1. Settings of important parameters

$(Desc - 1) \times (1 - \frac{7}{8})$, so we set T to $\frac{Desc_{max}-1}{8} + 1$.

4.2 Computational Results and Comparisons

In the following, there will be listed the used performance measures of BLS algorithm:

1. the average deviation of obtained solutions from the best known solution, denoted δ:
$$\delta = 100 \times (\bar{c} - bks)/bks\,[\%] \qquad (6)$$
where $\bar{c}$ is the average tour length over 20 runs of BLS, and *bks* is the best known value which can be found in the TSPLIB [16];
2. the number of solutions of which the deviation does not exceed 1 % (over 20 runs), denoted $C_{1\,\%}$;
3. the number of solutions where the cost is equal to the best known solution – C_{opt}. Instances with a zero in the two (merged) columns means that all executions found the best known solution;
4. the CPU time in seconds.

We compare the performance of BLS with the standard local search using 2-opt neighbourhood [17] (LS 2-opt) from the literature, and basic ILS[1] with a 2-opt descent and a Double Bridge Move perturbation. The comparison reported in Table 2 is based on the Local Search framework used in BLS.

The results above shows the great contribution of BLS on both standard local search and ILS. 38 of the 41 instances did not exceed a deviation of 1 % at least once over the 20 executions, of which 30 have never exceeded. 27 instances have reached

[1] Thomas Stützle. TSP-TEST, Version 0.9. Available from `http://www.sls-book.net`, 2004.

Instance	n	bks	LS-2OPT (δ)	ILS		BLS		
				δ	$C_{1\,\%}/C_{opt}$	δ	$C_{1\,\%}/C_{opt}$	CPU Time (sec)
burma14	14	3 323	–	**0**		**0**		0.00
ulysses16	16	6 859	–	**0**		**0**		0.00
ulysses22	16	7 013	–	**0**		**0**		0.00
eil51	51	426	–	0.469	14/5	0.199	20/8	0.58
berlin52	52	7 542	–	0.106	19/19	**0**		0.00
st70	70	675	–	0.741	13/3	**0**		0.97
eil76	76	538	–	0.929	11/3	0.492	18/5	2.67
pr76	76	108 159	–	0.518	20/7	**0**		1.41
gr96	96	55 209	0.997	0.466	18/4	0.215	20/5	6.00
rat99	99	1 211	0.614	1.652	7/0	0.057	20/12	5.72
kroA100	100	21 282	0.073	0.282	20/8	**0**		4.02
kroB100	100	22 141	0.379	0.632	14/5	0.012	20/18	6.58
kroC100	100	20 749	0.546	0.882	13/2	**0**		4.38
kroD100	100	21 294	1.538	0.977	13/0	0.053	20/14	5.16
kroE100	100	22 068	0.983	0.770	15/1	0.164	20/5	5.89
rd100	100	7 910	0.961	0.619	14/3	0.010	20/16	10
eil101	101	629	1.657	1.749	3/0	1.017	11/0	6.07
lin105	105	14 379	0.642	0.285	19/7	**0**		2.10
pr107	107	44 303	0.093	0.950	9/0	0.042	20/11	5.89
pr124	124	59 030	0.953	0.281	19/7	**0**		7.91
bier127	127	118 282	0.649	0.686	15/1	0.139	20/4	31
ch130	130	6 110	0.999	1.244	7/0	0.324	20/2	14
pr136	136	96 772	–	1.775	7/0	0.675	20/0	15
gr137	137	69 853	0.824	1.266	10/0	0.347	20/0	20
pr144	144	58 537	–	0.091	20/7	**0**		4.39
ch150	150	6 528	–	1.241	8/0	0.412	20/2	18
kroA150	150	26 524	–	1.421	6/0	0.374	20/0	26
kroB150	150	26 130	–	1.309	8/0	0.237	20/1	22
pr152	152	73 682	–	0.415	19/1	0.030	20/8	15
u159	159	42 080	–	1.694	5/1	**0**		25
rat195	195	2 323	–	2.927	0/0	1.157	1/0	40
d198	198	15 780	–	0.754	16/0	0.581	20/0	46
gr202	202	40 160	–	1.805	0/0	1.314	5/0	35
ts225	225	126 643	–	0.706	16/15	0.110	20/10	50
gr229	229	134 602	0.911	1.306	6/0	1.403	2/0	49
gil262	262	2 378	1.099	2.397	0/0	1.345	6/0	70
a280	280	2 579	–	3.373	1/0	0.866	12/0	68
lin318	318	42 029	1.202	2.253	0/0	1.384	2/0	152
rd400	400	15 281	1.543	3.030	0/0	2.493	0/0	280
gr431	431	171 414	3.045	2.357	0/0	2.243	0/0	294
pcb442	442	50 778	5.185	3.096	0/0	2.315	0/0	205

Table 2. Comparative results between BLS and standard local search using 2-opt

at least once the optimum, of which 12 have always reached within a reasonable time.

4.3 Justification for Parameter Settings

The good quality of the results obtained by BLS is due in part to the choice of the parameters (see Table 1), each of these is justified by its role and influence in BLS. The most influential of these will be confirmed by comparative tests on three TSPLIB instances (*eil*51, *eil*76 and *eil*101). Results will be represented on two superposed charts: the first is a stacked bar charts, each stack gives the best, average and the worst solution. It will show only two results if the best found solution is equal to the optimum. Second chart is a line chart showing evolution of running time of BLS. The first two parameters below are defined according to the instance size (N).

4.3.1 Maximum Number of Descents ($Desc_{max}$)

The maximum number of descents decides how many times the local search will be performed, the higher is the number of descents, the better are the results; each one is a new chance to find better tour or escaping the attraction. In return, the running time will be greater as shown in Figure 1. It was noted while testing that over a certain number of descents, the results become quite satisfactory and adding more only rises execution time without significant improvements.

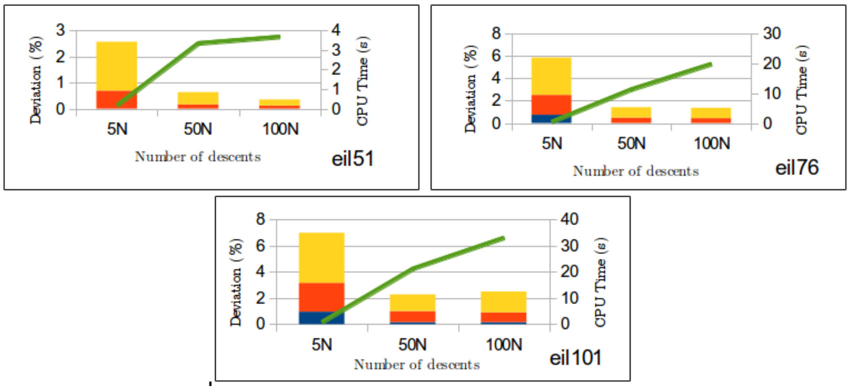


Figure 1. Varying number of descents for eil instances

4.3.2 Initial Number of Moves/Jumps (L_0)

This number gives the minimum of moves to be performed for each of the three types of perturbation; it has its impact on the change of neighbourhood and therefore the chances of unlocks. The larger it is, the more important is the changing neighbourhood, which may sometimes give the next descent a feeling of independence of the previous descents.

The best results were obtained as shown in Figure 2 when starting with a single jump, given the high effectiveness of diversification perturbations.

4.3.3 Maximum Number of Non-Improving Local Optima (T)

This variable indicates when we should perform the strong perturbation; it is reset to zero once this perturbation performed. The smaller T is, the higher is the number of perturbations, which increases the chances of unlocking. As mentioned earlier, the number T must be greater than $\frac{Desc_{max}-1}{8}+1$ to avoid that the current number

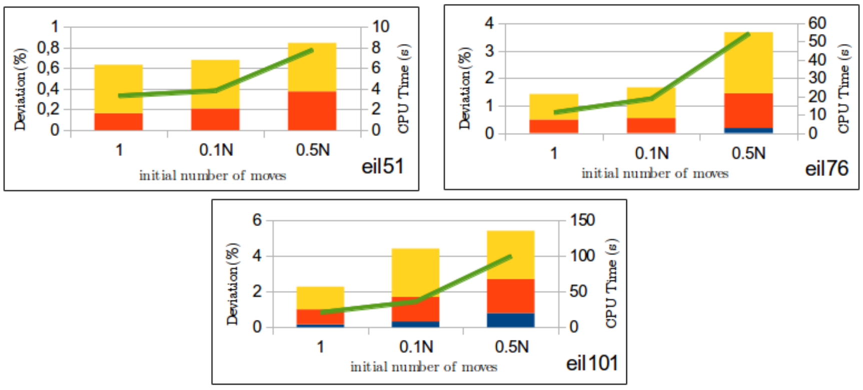


Figure 2. Varying initial number of moves for eil instances

of descents never reaches $Desc_{max}$. The tests above in Figure 3 are performed with variation of T three times: choosing firstly time the minimal value for each instance, and then doubling and tripling these values.

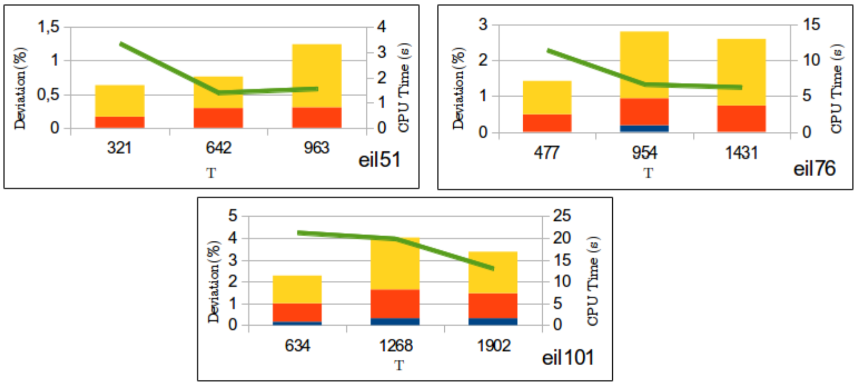


Figure 3. Varying maximum number of non-improving local optima for eil instances

4.3.4 Performing 3-Opt as Descent

BLS can be improved by changing neighbourhood in the steepest descent by switching to the 3-opt, which is larger due to the number of edges that are affected in each movement. This change will leave an impact on the performance of BLS: the history of changes is wider than the history constructed with the 2-opt steepest descent (three edges affected instead of two with 2-opt), then the perturbations become more diversified.

Table 3 shows BLS execution results with 3-opt (BLS 3-OPT), by comparing it with the standard local search using 3-opt neighbourhood (LS 3-OPT) [17] and BLS using 2-opt in the descent (BLS 2-OPT).

Instance	n	bks	LS-3OPT (δ)	BLS 2-OPT			BLS 3-OPT		
				δ	$C_{1\,\%}/C_{opt}$	CPU (sec)	δ	$C_{1\,\%}/C_{opt}$	CPU (sec)
burma14	14	3 323	–	**0**		0.00	**0**		0.00
ulysses16	16	6 859	–	**0**		0.00	**0**		0.00
ulysses22	16	7 013	–	**0**		0.00	**0**		0.00
eil51	51	426	–	0.199	20/8	0.58	**0**		2.69
berlin52	52	7 542	–	**0**		0.00	**0**		0.47
st70	70	675	–	**0**		0.97	**0**		11
eil76	76	538	–	0.492	18/5	2.67	**0**		8.55
pr76	76	108 159	–	**0**		1.41	**0**		21
gr96	96	55 209	0.997	0.215	20/5	6.00	**0**		23
rat99	99	1 211	0.614	0.057	20/12	5.72	0.033	20/12	16
kroA100	100	21 282	0.073	**0**		4.02	**0**		18
kroB100	100	22 141	0.379	0.012	20/18	6.58	**0**		62
kroC100	100	20 749	0.546	**0**		4.38	**0**		25
kroD100	100	21 294	1.538	0.053	20/14	5.16	**0**		35
kroE100	100	22 068	0.983	0.164	20/5	5.89	0138	20/10	52
rd100	100	7 910	0.961	0.010	20/16	10	**0**		54
eil101	101	629	1.657	1.017	11/0	6.07	0.063	20/12	39
lin105	105	14 379	0.642	**0**		2.10	**0**		29
pr107	107	44 303	0.093	0.042	20/11	5.89	**0**		36
pr124	124	59 030	0.953	**0**		7.91	**0**		21
bier127	127	118 282	0.649	0.139	20/4	31	0.102	20/9	247
ch130	130	6 110	0.999	0.324	20/2	14	0.216	20/7	196
pr136	136	96 772	–	0.675	20/0	15	0.374	20/0	244
pr144	144	58 537	–	**0**		4.39	**0**		12
ch150	150	6 528	–	0.412	20/2	18	0.117	20/14	284
kroA150	150	26 524	–	0.374	20/0	26	0.162	20/0	400
kroB150	150	26 130	–	0.237	20/1	22	0.185	18/0	626
pr152	152	73 682	–	0.030	20/8	15	0.041	20/6	374
u159	159	42 080	–	**0**		25	**0**		287
rat195	195	2 323	–	1.157	1/0	40	0.499	20/0	624

Table 3. Comparative results between BLS and standard local search using 3-opt

3-Opt neighbourhood brings many improvements to BLS. All running instances did not exceed a deviation of 1 %, the worst average does not even exceed 0.5 %. Only 4 out of 30 instances could not reach the optimum, while 21 have always reached the optimum.

5 DISCUSSIONS

Observing the overall framework of ILS, BLS uses local search to get local optima, and perturbation to vary the search. However, BLS differentiates itself from most ILS algorithms by the combination of various perturbation strategies of different strengths, triggered according to the search status. As explained in Section 3.3, BLS uses a perturbation of weaker diversification with a higher probability P as the search progresses toward improved new local optima.

By neglecting the maximum number of descents set in our algorithm, BLS always succeeds in finding the optimal solution. Indeed, the BLS search space expands after each series of perturbations until it finds the neighbourhood that his local optimum is the optimal solution. Below in Table 4 are shown the best and average running

CPU time execution (in seconds) of some instances of a BLS execution where we ignored this stopping condition criteria.

Instances	n	*bks*	Best	Average	Instances	n	*bks*	Best	Average
eil51	51	426	0	2.2	**kroC100**	100	20 749	1.4	20
berlin52	52	7 542	0	0.3	**kroD100**	100	21 294	24	433
st70	70	675	0.6	13	**kroE100**	100	22 068	1.19	47
eil76	76	538	1.2	9.4	**rd100**	100	7 910	1.5	55
pr76	76	108 159	0.3	6.9	**eil101**	101	629	2	257
rat99	99	1 211	2.1	960	**lin105**	105	14 379	0.6	41
kroA100	100	21 282	0.5	35	**pr107**	107	44 303	0.6	78
kroB100	100	22 141	8.1	95	**pr124**	124	59 030	1.3	27

Table 4. Execution time required for BLS to find the optimum

6 CONCLUSION

We explained in this paper the breakout local search approach for solving the TSP. This algorithm uses the ILS framework and brings improvements in the perturbation: it performs a local search and a perturbation-based diversification phase (to jump from a local optimum to another one). The local search procedure uses the steepest descent with 2-opt move strategy. To visit a local optima of high quality, the jumps toward new neighbourhood are adaptively controlled according to the state of search. Perturbation is achieved by varying the type of moves and then the size of a jump and selecting the most fitting perturbation for each diversification period.

The quality of BLS results reported in Section 4, proves its competitiveness compared to other algorithms. The repeated constructions (on each jump) of the set of candidate couples (formula (2), (3) and (4)) lead to a slowness of BLS compared to its competitors. For a better compromise of results' quality and execution time, we kept the 2-opt algorithm in the local search step so that BLS will not be penalised by the 3-opt slowness as shown in the comparison of Table 3.

In order to overcome this problem of slowness, BLS can be improved by introducing accelerated descent from the same neighbourhood and implementing efficient data structures to reduce the searching time in the construction of all candidates.

REFERENCES

[1] Reinelt, G.: The Traveling Salesman: Computational Solutions for TSP Applications. Springer-Verlag, Berlin, Heidelberg, 1994.

[2] Stadler, P. F.—Schnabl, W.: The Landscape of the Traveling Salesman Problem. Physics Letters A, Vol. 161, 1992, No. 4, pp. 337–344, doi: 10.1016/0375-9601(92)90557-3.

[3] Cook, W. J.: In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation. Princeton University Press, 2014.

[4] Benlic, U.—Hao, J.-K.: A Study of Breakout Local Search for the Minimum Sum Coloring Problem. In: Bui, L. T., Ong, Y. S., Hoai, N. X., Ishibuchi, H., Suganthan, P. N. (Eds.): Simulated Evolution and Learning (SEAL 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7673, 2012, pp. 128–137.

[5] Benlic, U.—Hao, J.-K.: Breakout Local Search for Maximum Clique Problems. Computers and Operations Research, Vol. 40, 2013, No. 1, pp. 192–206, doi: 10.1016/j.cor.2012.06.002.

[6] Benlic, U.—Hao, J.-K.: Breakout Local Search for the Max-Cut Problem. Engineering Applications of Artificial Intelligence, Vol. 26, 2013, No. 3, pp. 1162–1173.

[7] Benlic, U.—Hao, J.-K.: Breakout Local Search for the Quadratic Assignment Problem. Applied Mathematics and Computation, Vol. 219, 2013, No. 9, pp. 4800–4815, doi: 10.1016/j.amc.2012.10.106.

[8] Codenotti, B.—Manzini, G.—Margara, L.—Resta, G.: Perturbation: An Efficient Technique for the Solution of Very Large Instances of the Euclidean TSP. INFORMS Journal on Computing, Vol. 8, 1996, No. 2, pp. 125–133, doi: 10.1287/ijoc.8.2.125.

[9] Croes, G. A.: A Method for Solving Traveling-Salesman Problems. Operations Research, Vol. 6, 1958, No. 6, pp. 791–812, doi: 10.1287/opre.6.6.791.

[10] Fonlupt, C.—Robilliard, D.—Preux, P.: Fitness Landscape and the Behavior of Heuristics. Evolution Artificielle, Vol. 97, 1997.

[11] Garey, M. R.—Johnson, D. S.—Stockmeyer, L.: Some Simplified NP-Complete Problems. Proceedings of the Sixth Annual ACM Symposium on Theory of Computing (STOC '74), ACM, New York, NY, USA, 1974, pp. 47–63, doi: 10.1145/800119.803884.

[12] Glover, F.: Tabu Search – Part I. ORSA Journal on Computing, Vol. 1, 1989, No. 3, pp. 190–206, doi: 10.1287/ijoc.1.3.190.

[13] Glover, F.: Tabu Search – Part II. ORSA Journal on Computing, Vol. 2, 1990, No. 1, pp. 4–32, doi: 10.1287/ijoc.2.1.4.

[14] Hordijk, W.: A Measure of Landscapes. Evolutionary Computation, Vol. 4, 1996, No. 4, pp. 335–360, doi: 10.1162/evco.1996.4.4.335.

[15] Mladenović, N.—Hansen, P.: Variable Neighborhood Search. Computers and Operations Research, Vol. 24, 1997, No. 11, pp. 1097–1100, doi: 10.1016/S0305-0548(97)00031-2.

[16] Reinelt, G.: TSPLIB – A Traveling Salesman Problem Library. ORSA Journal on Computing, Vol. 3, 1991, No. 4, pp. 376–384, doi: 10.1287/ijoc.3.4.376.

[17] Blažinskas, A.—Lenkevičius, A.—Misevičius, A.: Modified Local Search Heuristics for the Symmetric Traveling Salesman Problem. Information Technology and Control, Vol. 42, 2013, No. 3, pp. 217–230, doi: 10.5755/j01.itc.42.3.1301.

[18] Arbel Krakhofer, B.: Local Optima in Landscapes of Combinatorial Optimization Problems. Master's thesis, University of Vienna, Austria, 1995.

[19] Lourenço, H. R.—Martin, O. C.—Stutzle, T.: Iterated Local Search. In: Glower, F. W., Kochenberger, G. A. (Eds.): Handbook of Metaheuristics. Springer,

International Series in Operations Research and Management Science, Vol. 57, 2003, pp. 320–353.

[20] Martin, O.—Otto, S. W.—Felten, E. W.: Large-Step Markov Chains for the Traveling Salesman Problem. Complex Systems, Vol. 5, 1991, No. 3, pp. 299–326.

[21] Bland, R. G.—Shallcross, D. F.: Large Travelling Salesman Problems Arising from Experiments in X-Ray Crystallography: A Preliminary Report on Computation. Operations Research Letters, Vol. 8, 1989, No. 3, pp. 125–128, doi: 10.1016/0167-6377(89)90037-0.

[22] Dantzig, G. B.: Discrete-Variable Extremum Problems. Operations Research, Vol. 5, 1957, No. 2, pp. 266–288, doi: 10.1287/opre.5.2.266.

[23] Lawler, E. L.—Lenstra, J. K.—Rinnooy Kan, A. H. G.—Shmoys, D. B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley and Sons, New York, 1985.

Mehdi El Krari is Ph.D. candidate at Mohammed V University in Rabat, Morocco. He is working on metaheuristics for combinatorial optimization problems related to logistics and transportation. His main interests are combinatorial optimisation, discrete algorithms, stochastic local search, evolutionary computation, etc.

Belaïd Ahiod is Professor in the Computer Science Department at Faculty of Science of the Mohammed V University in Rabat, Morocco. His research interests include NP-hard combinatorial optimization problems, multi-objective optimization, metaheuristics, nature-inspired algorithms, etc.

Bouazza El Benani is Professor in Mohammed V University of Rabat, Morocco, Department of Computing Science since 1994. He holds his Ph.D. in computer science from Montreal University, Canada. His research interests include artificial intelligence, software engineering, evolutionary algorithms, big data, metaheuristic algorithms, healthcare.

Computing and Informatics, Vol. 37, 2018, 673–692, doi: 10.4149/cai_2018_3_673

SMART DOCUMENT-CENTRIC PROCESSING OF HUMAN ORIENTED INFORMATION FLOWS

Magdalena Godlewska

Institute of Informatics
Faculty of Mathematics, Physics and Informatics
University of Gdansk
Wita Stwosza 57, 80-308 Gdansk, Poland
e-mail: `maggod@inf.ug.edu.pl`

Abstract. Usually people prefer to focus on creative rather than repetitive and schematic work patterns. Still, they must spend a lot of time complying with the procedures, selecting the information they receive and repeatedly restoring the previous state of work. This paper proposes the Mobile INteractive Document architecture (MIND) – a document-centric uniform interface to provide both effective communication of content and coordination of activities performed on documents. MIND documents are proactive, capable of initiating process activities, interacting with individuals on their personal devices and migrating on their own between collaborators. Each MIND document is a mobile agent that has built-in migration policy to control its own workflow and services enabling proper processing of contained information. The architecture supports users in the implementation of procedures, and selection of services needed to work on the document. A Personal Document-Agent (PDA) is a further development of MIND aimed at preserving continuity of state of individuals' work to support their creativity and comfort of their daily work.

Keywords: Human-computer interaction, electronic documents, multi-agent systems, collaborative work, workflow management, task coordination, knowledge-based organization

Mathematics Subject Classification 2010: 68-M10, 68-M11, 68-M12, 68-M14, 68-U35

1 INTRODUCTION

Despite of the intensive development of artificial intelligence and machine learning, people are still the key intellectual resource in almost all areas of life. Yet, supporting people's interaction with various knowledge resources would contribute to their productivity and well-being, and help them to focus their activities on creative work, and to put less effort to perform simple, structured tasks.

Working in a group, as well as individually, humans perform certain processes. In an organization of many people, in particular in a *knowledge-based organization* [1], a collaboration process is often implementation of the established procedure. Moreover, collaboration with other workers, in accordance with the organizational procedures, enables converting knowledge of individuals to a collective organization knowledge. The purpose of the knowledge-based organization is to implement the *knowledge process*, in which the human mind is an important element.

There are several problems that hinder effective collaboration, in particular:

1. The worker needs to know the organizational procedures, which are often very confusing. Especially in procedures that are rarely performed, it could take a long time before the worker finds out to whom a particular document should be sent.
2. Collaborators perform procedures manually, and can make mistakes such as sending (receiving) some information several times to (from) a wrong person. This leads to *information overload* phenomenon, that often leaves the worker confused and unable to make a decision [2].
3. Workflow process automation often requires large amounts of work to define the entire process before performing. Moreover, this is often infeasible as the process flow often depends on individual decisions made during its performance.
4. Documents usually play a passive role in the process, which means that they are opened, filled, sent, etc. They generally do not give any support to users, or do that only in a very limited form.

Also in individual work, there are some problems affecting the efficiency of the performed tasks:

5. A user is the only one person who knows his devices and applications installed on them. Nowadays, this is a problem because users use multiple devices to continually perform the same task at work and home including PCs, tablets and smartphones which must be synchronized. A question arises how to synchronize entire systems, not just separate files.
6. A user needs to install the same or similar applications and configure peripherals for each device or OS.
7. Even if a person uses cloud services, like as Google Drive or One Drive, he has to find the right documents and recreate the state of his recent work after changing a device or rebooting its operating system.

8. A user enacts his own accustomed processes while working or resting, which are not supported in any way.

This paper presents the Mobile INteractive Document architecture (MIND) [3] and a special workflow enactment application, a Local Workflow Engine (LWE) [4], enabling a loosely-coupled agent system, capable of coping with the above points.

The MIND architecture is a model of a document-centric uniform interface to provide both effective communication of content and coordination of activities performed on documents. MIND documents are *proactive*, i.e. they are capable of initiating process activities, interacting with individual workers on their personal devices and migrating on their own between collaborators. Thus, each MIND document is a mobile agent, called a *document-agent*. Document-agents have built-in migration policy to control their own workflow and services to properly process contained information. Section 3 provides a more detailed overview of the MIND architecture.

The migration path of a document-agent contains all information and status of the workflow process to perform it locally on users' devices. A document is transferred between users in a serialized form via an available protocol – the MIND architecture does not impose any specific implementation of that. The choice of a concrete protocol depends on the requirements of the organization, in particular on security levels, a number of employees, a comfort of use, etc. In virtual knowledge-based organizations, email can be used as basic medium for exchanging digital documents of any kind. Implementation of the document transfer protocol is discussed in Section 7.

The LWE application mention before is installed on each worker's device participating in the process. It has a workflow enactment capability, i.e. a functionality to activate document-agents and switch documents between the activity and transition phases of the workflow. All LWEs participating in the process, and performing independently, form together both a technologically independent loosely-coupled agent system and a distributed workflow enactment service. Section 4 outlines generic functionality of LWE and the idea of a distributed workflow enactment service.

In the LWE-based MIND system, workers perform activities on documents independently, using their personal devices, and yet collaborate on achieving a common goal. It is possible owing to a migration policy embedded in each document. This policy defines for each document a workflow process composed of specific document-flow patterns [5] that provide process-wide coordination. The document-flow patterns are a result of analysis of the coordination patterns proposed by van der Aaalst [6]. This shows that a relatively small and well defined set of collaboration patterns contains building blocks of arbitrary complex workflow processes in real organizations. Thus, document-flow patterns proposed in this paper, which are based on these collaborations patterns enable modelling and coordination of any workflow process with MIND documents. Moreover, the proposed distributed workflow enactment service allows defining dynamically the workflow process during its actual execution. Section 5 discusses briefly the document-flow patterns and shows how to modify a workflow dynamically.

The proposed solution allows for significant reduction of the problems with group and individual work mentioned above. Points 1–4 present the problems for which the MIND architecture was developed. The system enabling group work through performing of knowledge processes has been implemented and validated as a part of the MENAID (Methods and Tools for Next Generation Document Engineering) project [7] – Section 7 presents the results of this work. An attempt to solve the problems introduced in points 5–8 is based on a concept to apply the model of the MIND architecture to improve individual work on different devices, with a Personal Document-Agent (PDA) outlined in Section 6.

Section 2 opens this paper by reviewing work related to the presented research, while Section 8 concludes the paper and introduces the possibilities of further development of the presented solutions.

2 RELATED WORK

The idea of an active document is not new. Already in 1996, the Multivalent Document architecture MVD [8] was presented and it was the first significant step in the document-based processing. MVD allowed for treating the document as an object the content of which can be manipulated dynamically. It introduced active functionality with dynamically loaded objects called *behaviors*. The MIND embedded services are similar to the concept of behaviors, however MIND introduces also local and external services, to manipulate a document content, but are not components of documents. This gives documents more flexibility, adjusting them to exploit local resources of visiting devices and to easily add a new functionality.

The Placeless Documents [9] extend document functionality with active properties that can not only allow to manipulate a document content, but also can manage of a document structure and its workflow. These are also the main features of the MIND architecture. However, the Placeless Documents are reactive, i.e., they respond to external events, while MIND documents are proactive – they initiate their own behavior as they have their own embedded functionality (services).

It is worth mentioning the document-agent MobiDoc platform [10] as the concept of a proactive document, capable of travelling between computers under its own control. This platform was closely related to the particular technology, thus lacked forward compatibility, and, consequently was difficult to implement in a large-scale. The idea of a document-agent is very interesting, as openness and technological independence are very important features of modern systems. In the LWE-based MIND system, a technological independence is one of its priorities. In a special case, when each user has a different operating system, each LWE may be implemented in a different technology. The document transfer protocol can be also adapted to the requirements of implementation. It gives the opportunity to create an agent platform with all benefits of multi-agent systems, but without a need to implement a full-size agent platform that depends on the chosen technology, has to be updated regularly and requires additional skills from administrators.

Among more recent solutions that allow agents to perform a workflow is WADE (Workflow and Agents Development Environment) [11] agent platform based on JADE (Java Agent DEvelopment framework) [12]. WADE agents embed a micro-workflow engine, capable of executing workflows compiled before launching a workflow. Performing activities may be delegated by one agent to another and in principle it is not related to agent mobility. This solution follows a classic central workflow enactment philosophy, and differs from it only in decentralization of a global process state into a process states controlled by micro-workflow engines running inside agents. This solution makes the WADE platform different from the MIND architecture. In MIND, a workflow object specified formally with XPDL (XML Process Definition Language) [13] is bundled with a document and contains its internal state of which it is of full control. Then, a workflow is enacted outside of agents by local workflow engines (LWEs). More details on the concept of distributed workflow enactment can be found in Section 4. The advantage of such a solution is that a respective XPDL file may be modified during process execution. Moreover, MIND document-agent is the only communication interface, which makes it technologically independent in a loosely coupled and heterogeneous distributed system.

There are some interesting solutions, such as AMODIT [14], that use elements of artificial intelligence to improve workflow processes. The main concept is to analyse the content of the documents and previous decisions, in order to suggest the next steps of the workflow. The mentioned system is a commercial product based on a client-server model, which does not exhibit openness, technological independence and multi-agent approach, as the MIND architecture does. The idea of adopting a Multi-Agent System and machine learning to support cooperation based on documents was introduced in [15]. The presented idea was based on the analysis of the documents and users' behaviors, omitting the problem of process definition. The MIND architecture, allowing for the dynamic modification of the process, opens the possibility of learning the flow of the processes and behavior of users.

One of the problems in individual work, mentioned in point 7, is saving a state of a recent work. This problem occurs, for example, when the system needs to be rebooted. The user often loses the information about all documents or applications opened before. Sometimes, it can be really frustrating. Users of the Mac OS receive the greatest support on this issue. They can simply decide if they want to re-open applications in the same state they left them before logging out [16]. It is also possible, although not so easy, in Linux systems (e.g. Ubuntu). There is the `dconf-editor`, where the `auto-save-session` check box may be selected. The `dconf-editor` is not preinstalled by default, so even in experienced users may have a problem using it. Finally, in the Windows systems it is not possible to re-open active programs on reboot without installing an external application. There is only the possibility to automatically re-open any Explorer windows that were opened before rebooting. There is a list of external applications that enable to restore programs or folders after system reboot, e.g. Cache My Work [17] or SmartClose [18].

Restoring applications after a system reboots solves the problem on just one operating system. Currently, users would like to work on many devices, e.g. to start work in the office, continue it on the way home, and end at home. On each device, users would have to recreate a state of their current work. Google Drive [19] or One Drive [20] cloud based solutions could be very useful for sharing files through different devices, but they cannot ensure continuing work at the point where it previously has been interrupted.

The MIND architecture allows for implementation of a Personal Document-Agent (PDA), which can store a global state of the work, interact with a user, operating systems and services. This solution differs from the idea of Virtual Personal Assistant (VPA), like Apple Siri [21], Google Assistant [22] or Amazon Alexa [23], which are based on interaction with a user. The latter act as local applications or web services and they support a user in everyday duties, e.g. by turning on an alarm, checking the weather, reminding a meeting or making purchases. PDA proposed in this paper is a smart middleware between operating systems and applications rather than a yet another application like VPA (it can, however, use VPA as a local or external service).

3 THE MIND ARCHITECTURE

Traditionally, electronic documents have been static objects downloaded from a server or sent by an email. MIND allows static documents to be converted into a set of dynamic components that can migrate between collaborative workers according to their migration policy.

The concept of the MIND document lifecycle is illustrated in Figure 1.

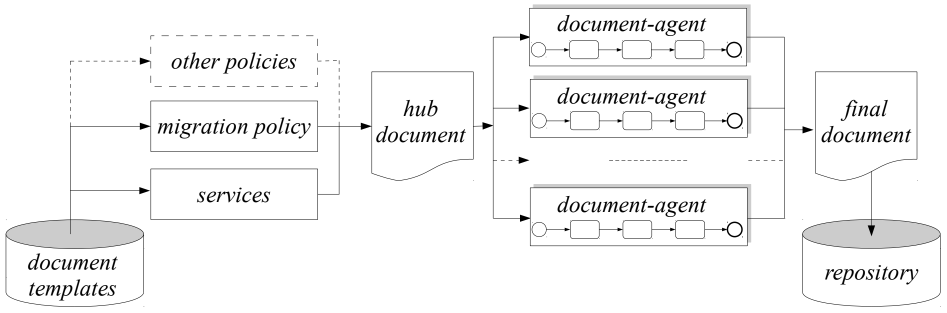


Figure 1. The MIND document lifecycle [4]

A *hub document* is formed on the basis of document templates that includes *migration policy*, which specifies steps of the knowledge process and services that will be performed on different parts of a document during the process. The hub document that contains static elements (e.g. XML files), is *unmarshalled* to mobile objects called *document-agents*, which perform their mission in a distributed agent system. Each document-agent migrates across an organization and interacts with its workers.

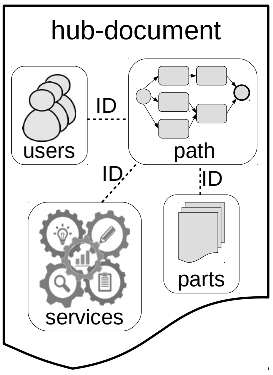


Figure 2. Components of a MIND document-agent

The *hub document* contains five components, outlined in Figure 2:

hub-document, a headline of the MIND document. It contains its ID and other related information necessary to identify a document-agent with a given process.

users, containing data about users who participate in a process.

parts, containing information about constituent documents, which are actual documents on which work will be performed, e.g. PDF files, MSOffice documents, HTML forms, etc. Such constituent documents are parts of the MIND document, which does not mean that they always migrate with document-agents. Sometimes, a part can describe an external location of a constituent document.

services, containing information on a document functionality available during a process. Three types of services are possible: *embedded*, transferred together with a document-agent, *local* which may be acquired by a document-agent from a local user's device, and *external*, to be called on remote hosts by a user's system at the request of an arriving document-agent. Services provide document-agent functionality and make it proactive as it was mentioned in Section 2.

path, defining migration policy (workflow) of each part of a document. It specifies steps of a process and activities that should be performed at each step of a process. The other components refer to the path and are distributed according to the path during a process.

The presented document-agent model complies with the Belief-Desire-Intention (BDI) definition of an agent [24]. BDI is an agent architecture that reflects model of human practical reasoning, developed by Bratman [25]. The main goal of the MIND architecture is to support users in making reasonable decisions, so BDI model fits well to document-agents, because it represents the natural knowledge processes of a human mind.

A model considers *beliefs* as a knowledge about the world, *desires* as goals of an agent and future-directed *intentions*, which are composed of plans, as an important and irreducible concept. Particularly, for the MIND document-agents:

- the *world* is a knowledge-based organization, in which users have devices and applications that can be an environment for agents and these devices are from time to time connected to a network enabling migration or communication of agents.
- *beliefs* are components of document-agents: hub-document, users, parts and services. An agent does not need to know all users or services of an organization. It is enough for the agent to know a subset of the world that is needed to perform a designated process. Especially, during a process, a document-agent can "discover" the world, i.e., add users, parts or services.
- *desires* are represented by the path component that reflects steps of a knowledge process striving to achieve an organization's goal.
- *intentions* are related to the agent's autonomy. In the Bratman's model, plans are initially only partially conceived, with details being filled in as they progress. The MIND document-agent autonomously follows its path, which can be modified dynamically during the process. It is also able to designate the services needed to perform the appropriate action on a constituent document.

Although MIND meets requirements of the BDI model, it is not its direct implementation, as in the latter, agents do not have any specific mechanisms to learn from past behavior and adapt to new situations. The MIND document collects all information about its performance, which can be an appropriate training set to teach agents to perform a process better in future, for example negotiation [26].

4 DISTRIBUTED WORKFLOW ENACTMENT

A key feature of the MIND architecture is physical distribution of business process activities, performed dynamically on a system of independent personal devices. The MIND documents have built-in process definition and functionality (the respective path and embedding service components mentioned in the previous section). This makes them document-agents, which are autonomous and mobile. Especially, they are independent of any particular platform supporting workflow enactment and they are capable of launching individual activities onto various users' devices, which maintain process coordination across an organization.

A standard WfMC *Workflow enactment service* [27] interprets the process description and controls sequencing of activities through one or more cooperating workflow engines. Even if workflow engines are distributed, workflow enactment is centralized in most of the implementations, because control data must be available to all engines. Contrary to that, in the MIND architecture, all data needed for workflow enactment are embedded in documents, and allow for implementation of a really distributed workflow enactment service, consisting of Local Workflow Engines (LWEs).

The idea of the distributed workflow enactment service built on top of LWEs is illustrated in Figure 3.

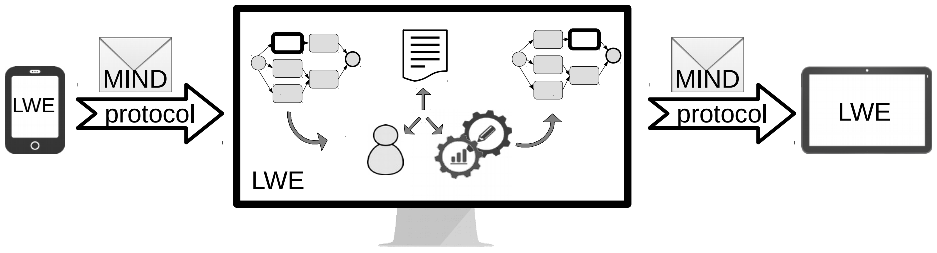


Figure 3. Distributed workflow enactment service based on LWEs

Each LWE is independent of other LWEs, so it can be implemented in any technology and adapted to requirements of particular devices, especially mobile devices such as tablets and smartphones.

A MIND document-agent is sent in a static, serialized form via any network protocol available, enabling transmitting a document-size data. For example, it can be a FIPA MTP standard protocol [28], ordinary HTTP protocol, or just email protocols such as SMTP or IMAP. Section 7 outlines several possible implementations of transport layer for the MIND documents.

The initial state of workflow enactment is when LWE downloads a serialized document on a local device and activates it, which means unmarshalling and launching its embedded functionality. The activated document-agent begins its mission by obtaining the path component and determining the current activity that should be performed in this particular step of the process. The document-agent contains all data needed to determine the state of the workflow process locally. Thereafter, the document-agent may interact with users, their local systems and some external services. If the next activity is intended for another user, the document is serialized again, packed and sent to the next user via a network protocol.

All workflow process data are brought to LWE by a document. So document-agents are the only means of communication between LWEs in a distributed system. Since LWEs can be implemented in different technologies, they can be adapted to various hardware.

5 DOCUMENT-FLOW PATTERNS

To run the appropriate activity in a workflow process locally, a MIND document-agent must contain not only a process definition, but also its current execution state. This state contains an ID of a *current activity* assigned to a given user, stored in an external XPDL attribute. But it is not enough in many situations in a process. In order to define all attributes that form a process state, a set of *document-flow patterns* has been defined. These attributes, namely *current activity*, *counter*, *sentinel*, *finish* and *semaphore* are implemented as internal variables of the MIND document and operated by a handling LWE as described below.

Based on the work of van der Aaalst [6] and characteristics of a distributed system three categories of the document-flow patters have been identified: distributed state patterns, coupled state patterns, and embedded state patterns [4].

5.1 Distributed State Patterns

These patterns describe situations in which a next activity or activities can be determined solely on the state of the *current activity.* Four patterns of this type have been distinguished:

Document sequencer involves a user transferring a document to another user. The document may be transferred in its entirety in one package or if it is too large for a transport layer protocols, it may be partitioned into several packages. If the document is partitioned into smaller pieces, the numbering of packages is entered and the last package is marked with the *sentinel* attribute. Thanks to that, a recipient knows if all packages of the document have been already received, even if they come in a different order.

Document splitter creates identical copies of a document or partitions it into separate fragments. Whether the document would be copied or decomposed depends on the functionality of the document provided by services. The resulting documents get new document IDs and they are transferred to respective users specified in a migration policy. They are modified parallelly while executing different activities. For each document, ID of an activity that would be executed on it in the next step is set as a *current activity* attribute value.

Depending on conditions defined in the workflow process, the splitter produces a different number of copied/fragmented documents. This number is stored in the *counter* attribute. If the splitter has n outgoing branches, the *counter* assumes a value from 1 to n: 1, if only one branch was chosen and n, if all branches were chosen.

Document merger complements the document splitter pattern and merges all received documents in one. Of course, it may involve various document functionality, depending on whether the preceding splitter has been cloning or decomposing. But before merging, all expected documents must be delivered. The LWE client on the basis of path component of the first received document (the value of the *counter* attribute) determines a number of expected documents that have to be merged.

Document iterator enables repeated execution of some sequence of activities controlled by a condition specified in a respective document migration policy. Many workflow languages allow for creating unstructured loop with more than one entry or exit point, that do not need any specific looping operators [6]. In this case, the value of the one boolean *finish* attribute brought by document to logic gateway can decide, whether a workflow should continue a loop or exit from it.

5.2 Coupled State Patterns

Sometimes completion of an activity performed by one user may require a notification on a state of some activity performed by another user somewhere in an organization. That involves a notion of asynchronous signals, sent between different parts of a workflow process. Three document-flow patterns of this kind have been distinguished: deferred choice, milestone and cancel activity.

Milestone and deferred choice are used to deal with situations when the current activity of one user has to be blocked until a signal notifying on some external event has been received from another worker. Both patterns require the *semaphore* attribute and embedded functionality to handle it. Initial value of the *semaphore* is closed, so if a signal from another worker has not been received, the current activity is blocked. Upon receiving a signal, a waiting activity is resumed. Milestone just blocks some activity of one user by another. Thus, a signal does not have to have any specific value. Deferred choice is used when sending a given document has to be postponed until the user gets information to whom it should be sent. Thus, a signal should have a value identifying a next user, e.g. the user's ID.

Cancelling pattern. Implementation of this pattern depends on which process activities should be cancelled. If a particular activity should be cancelled, a cancellation signal is sent only to LWE responsible for its execution. The decision on cancelling the activity is immediate for a receiving device or does not make sense any more if a document has been sent to another user. More problematic situation is to cancel a document (one part of the MIND document), because it requires a designation of its location. It is possible to search for a document in all places indicated by the workflow, but this solution is expensive and can be unreliable. Another solution is to chase a document that can leave a trail in each visited LWE. It is worth mentioning that a document flow takes hours, even days, rather than seconds, so it would be a reasonable solution. For example, the Intel's Email Service Level Agreement defines the acceptable time frame for replying to emails in 24 hours [29]. Using an external "ground control" service [5], which introduces the ability to track document-agents globally, allows for simplification of a document cancellation.

The cancelling pattern does not need any additional attributes to enable cancellation. In principle, every activity or every document can be cancelled. The *cancel* attribute is added after the cancellation to indicate this fact, which may be needed for further analysis of a process.

5.3 Embedded State Patterns

Performing an activity by some user may require a subprocess delegated to someone else with activities not specified originally in a migration policy of an arriving document. States of such a subprocess are embedded in a state of a current activity referring to that subprocess. Two types of subprocess can be distinguished: internal and external.

Internal subprocess. If a current user is authorized to extend an original migration policy of a document with new activities, they constitute an internal subprocess. Neither a structure of the internal subflow nor identity of added users have to be known earlier to a workflow originator (designer).

This is a key pattern in the MIND architecture. Internal subprocesses can be added during the workflow execution. It allows for defining a relatively small initial workflow and its dynamic expansion during execution. The activity assigned to the user is converted into a subprocess activity. The main advantage of editing a process during its execution is that it can be built ad hoc of small pieces. Each worker can define a fragment (subprocess) of a workflow, i.e. each user can be a document originator.

External subprocess. A performed activity may call some external subprocess whose structure is unknown for both, a workflow designer and a performer of a current activity. The external subprocess is often performed outside of an organization.

6 A PERSONAL DOCUMENT-AGENT

A special case of an iterator is recursion, which means the ability of some activity to invoke itself during its execution [6]. For documents, it is a situation in which a user performs the same activity several times on different devices. From the perspective of an entire process, this is still one activity, but from the perspective of a document-agent, executing conditions on different devices can vary significantly. This situation requires creating a subprocess with one input and one output, in which a document will be transferred between user's devices and edited until the end of work.

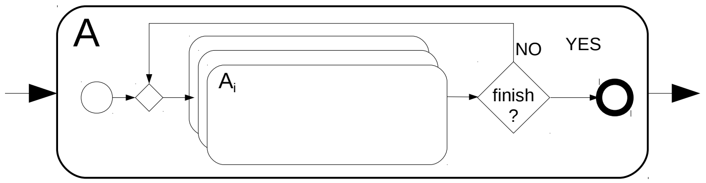


Figure 4. A recursive document-flow pattern

Figure 4 presents a recursive pattern as a subprocess added to the activity **A**, being a certain task that a user has to perform on a document part. In many cases, a user does not perform this task at once. Thus, the document has some intermediary states between receiving and sending. It can be opened, edited, saved and closed many times during one activity. LWE can put to sleep and next wake up a document-agent as many times as needed, including the restoration of necessary services. This is a typical behavior of the MIND document and does not require any subprocess. A subprocess would be needed if one activity would have to be performed on different devices of the same user. It is a common situation, as many people have several devices at their disposal. Each activity $\mathbf{A}_i$ is an i^{th} copy of the activity **A** started before adding the subprocess and each one expresses the same task **A**. After performing a certain stage of the task represented by $\mathbf{A}_i$, the user decides, if a work is completed. If not, it may happen that it would be continued on another device as $\mathbf{A}_{i+1}$.

A document-agent, when interacting with LWE, can recognize on which device it currently resides. Thus, it can dynamically adjust to various execution contexts provided by devices. It may have a certain performance strategy for a given device, which it can consult with users by negotiating with their device [30].

The recursive pattern proposed in Figure 4 can been exploited by the concept of a Personal Document-Agent presented in Figure 5.

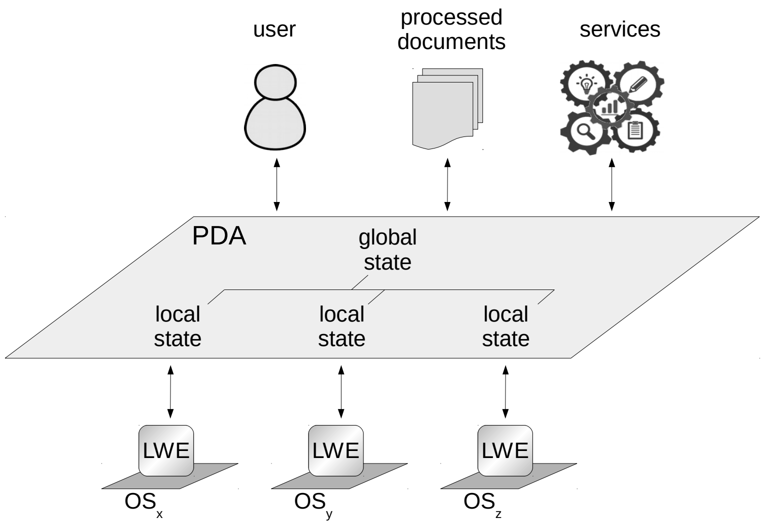


Figure 5. PDA as a middleware between user and his OSs

Assume that everything users do on their devices is the implementation of their *personal process*, and LWE has been installed on each of them. The special MIND document does not implement a path assigned in advance, but follows users to support their work. This is a process that is being built ad hoc. In a personal process, a state of work is often more important than the sequence of activities performed. A user interrupts it at a certain point and after some time wants to resume it in the same point where it was interrupted. This is not a problem of just one device. Most operating systems allow for putting them into a sleeping or hibernating mode. Incidentally, the system is not able to wake up and then the state of work would be lost. It would be much more useful to provide a prospect for continuing work from the sleeping/hibernating point, but on another device.

PDA collects the data about tasks performed by the user, as snapshots of a state of work. Next, PDA distributes the obtained data into two sets: data important for the work on local operating system (a local state) and data that express the state of performing a certain task (a global state). This distribution is supported by the MIND service component (see Section 3). The local state contains information about local services that users used in their latest work, while the global state registers the use of external and embedded services. But not only, the global state also includes mappings between corresponding applications on different devices, for example, Adobe Acrobat Reader on PC with the Windows OS would correspond to Google PDF Viewer on a tablet with the Android OS. It also can indicate discrepancies, for example an AutoCAD application would be only available on a computer at work.

PDA also uses the part component to enable users interaction with the documents they have recently worked on. It keeps information not only about open documents, but also about specific places in these documents, if it is possible for a given format. Thanks to this, users changing the device, are redirected to the exact place in document.

Interaction between PDA and a user could evolve in time. PDA can use many services to cooperate with a user, including voice communication. However, turning on too many services at the beginning, would overload the system and a user might not be able to use so many of them. Instead, PDA should be kept as small as possible and should be able to collect data about the user's habits to better support the latter. For this purpose, various solutions are currently being adopted by the Author, such as machine learning to automatically built document migration, strategies or emotion recognition of its local user [31]. For example, let imagine the situation, that PDA "knows" a user named Bob. PDA knows, that Bob never works on Friday evenings. If he turns on the computer, he usually watches movies or plays video games. He gets angry when anything reminds him of work. Therefore, PDA would ask Bob what to turn on for him: his recently acquired video game or a Netflix service, and only one exception would be allowed – when the tight deadline to complete task is about to expire. Then Bob would accept the suggestion to continue his work instead. In such a case, PDA would readily recreate the state of his recent work.

PDA can collect some other data, not only a status of work. It is practical to collect data about peripheral devices, so as not to have to configure them for each device separately and to communicate with them faster and easier.

7 CASE STUDY AND VALIDATION

The MIND architecture was created to facilitate knowledge management in complex knowledge processes, in which a flow of electronic documents and extracting knowledge from them are crucial. Coordination of document workflows may be often enforced by law, especially when related procedures are implemented manually by workers – as it takes place in court trials, crash investigations or complex medical cases.

The first case study was a judicial proceeding system. A real judicial case in the form of complete files could reach an enormous size. The purpose of this exercise was to consult with court workers (judges, attorneys, counsellors, judicial officers, etc.) to verify the usefulness of the proposed document-flow patterns and their required functionality. In court trials, there are many constituent documents that have specified structure and workflows are precisely defined by legal procedures. So, using MIND-like documents would be essential to redirect attention of all stakeholders involved to the content of a court trial, rather then concentrating on complying with the legal rules governing it. A feasibility study of the MIND architecture was carried out in cooperation with lawyers and a company providing software for courts. This allowed for developing the MIND document model and defining the document-flow patterns.

The second case study involved the issue of evaluation of students in a typical university grading process. It allowed to test the validation of implementing the MIND architecture in a real environment, using the document-patterns.

A worker of Registrar's Office forms a grade roster hub document, and transfers it to a Course Leader. A Course Leader runs his own subprocess of collecting credits from instructors during the entire semester; structure and implementation of that subprocess is irrelevant to the Registrar's Office. While the Registrar's Office may use an online grade system for one-time roster submission and approval, a Course Leader is responsible for all subprocesses of collecting credits and has modification, control and cancellation permissions. Instructors receive only a class roster of their student's groups, which can be filled out at any time, before a specified deadline.

Several prototype applications were implemented to validate the MIND architecture and demonstrate its implementability in the context of the above mentioned case study. The main task of the implementation was to create an environment for document-agents: for their actions on users' devices and for transferring between devices. It was common for all prototypes to use XML [32] for MIND document implementation. The path component that describes the document workflow has been specified in XPDL. XML files can be easily transformed to other formats, tailored to the specific technology.

The first implemented prototype [3] used the JADE. Document-agents extended the JADE MobileAgent class and LWE was a component of a JADE container. The transport layer was built on top of the JADE IMTP protocol. However, users were hesitant to use this prototype as difficult to configure and maintain, subscribing too much to the specific technology, and requiring troublesome inclusion of additional ports, often blocked by intermediary firewalls.

Further prototypes used email, and standard email's protocols, as a transport layer, with LWE implemented simply as a lightweight email client. Email is the most popular computer mediated communication in the workplace, as a simple textual form combined with a possibility to disseminate attachments in any format. This solution did not require additional skills from users and could support asynchronous work. LWE was implemented in several leading technologies: as a Java desktop application for PCs and laptops, and for mobile platforms: iOS, Windows Phone and Android [33]. These implementations used, however, different available libraries for email messaging, serialization and compression of documents, and XML data binding. LWE prototypes were tested simultaneously, while performing one process, so they formed a really heterogeneous system.

Installation of LWE on each device is recommended to take full advantage of a distributed system. The implementability of LWE was proven by prototypes implemented for various platforms. They were lightweight standalone applications and there was no need to configure any servers or databases. With email protocols used as the underlying transport layer, the configuration proceeded in the same way as the configuration of a typical email client. Alternatively, LWE could be provided as a Web service – especially in cases, when a user has a device for which LWE has not been implemented yet. Also, a user may play a marginal role in the process or just want to refuse installation of LWE.

This system based on email worked satisfactorily and has been considered by users as friendly. The Course Leader was free to implement his evaluation process in any way and course instructors could perform their activities using their personal mobile devices in any time, even if they were out of their campus network. The grading process involved both scheduled and unpredictable events, such as project assessment or homework collection for the former, and grade correction or disciplinary actions in a case of academic misconduct. These events were effectively handled with document-flow patters outlined in Section 5. The users' satisfaction was also influenced by: a simplicity of use, configuration, and easiness of the LWE application, as well as the ability to work without a permanent internet connection.

8 CONCLUSIONS AND FUTURE WORK

One of the main objectives for the presented MIND architecture has been its openness to new policies, services and diverse applications. Some of them have already been implemented, while others are still in the development phase.

Executability and *mobility* constitute the enabling services for MIND document-agents (see Section 7). The former involves unpacking, assembling and activating arriving document components to enable execution of the current activity, and after that packing them back before their departure, while the latter involves transporting them between personal devices of users to proceed with subsequent activities.

Next, *reliability* of the MIND agent system has been provided by a "ground control" external service [5] to make the distributed workflow enactment system more useful and trustworthy. It allowed for estimation of the global state of a distributed loosely coupled system, taking into account transport layer errors, unforeseen actions of users and process modifications. The "ground control" service together with LWE enabled communication between a persons responsible for executing a process and performers of a related activity. Additional permissions and rules allowed to determine which participant could control the process execution and make decisions in unforeseen or conflict situations. That also allowed for introducing the *choreography policy* [5] into the process enactment.

There is also a *security* issue, which answers the question: what to do if a document gets to an unauthorized person? LWE may require authentication of a user before unpacking and activating document components – to verify if the performer assigned to the current activity is the same person as the recipient of the document. An interesting solution for that has been proposed in [34]; it introduced a biometric face recognition mechanism built in MIND document-agents.

Finally, a *negotiation* capability was added to MIND documents to resolve possible conflicts between document-agents and user's devices they visit to execute a particular activity at their workflow [30].

Personal Document-Agent (PDA) presented in Section 6 is the next concept building on the MIND architecture. It explores executability and mobility of the MIND document-agents to improve the work of a specific user with his devices and peripherals. Thanks to this, a user may have an impression of continuing work from the point where it was interrupted, despite of changing the device and its location.

The MIND architecture enables dynamic modification of the workflow process. After the workflow process is completed, the document is archived (see Figure 1), and the data collected during it are a great base for analysis. Retracing already completed processes allows for their optimization in accordance with the real behavior of users. Machine learning approaches can be used to choose the best path or services in the process. During the process, users can assess the accuracy of the activity, that they performed. For example, if a document got to someone's device unnecessarily, it could learn not to follow such a path in the future.

Acknowledgment

This work was supported in part by the National Science Centre, Poland, under Grant No. DEC1-2011/01/B/ST6/06500.

REFERENCES

[1] Bhatt, G. D.: Organizing Knowledge in the Knowledge Development Cycle. Journal of Knowledge Management, Vol. 4, 2000, No. 1, pp. 15–26, doi: 10.1108/13673270010315371.

[2] Spira, J. B.: Overload!: How Too Much Information Is Hazardous to Your Organization. John Wiley and Sons, 2011.

[3] Godlewska, M.: Agent System for Managing Distributed Mobile Interactive Documents in Knowledge-Based Organizations. In: Nguyen, N. T. (Ed.): Transactions on Computational Collective Intelligence VI. Springer-Verlag, Berlin, Lecture Notes in Computer Science, Vol. 7190, 2012, pp. 121–145.

[4] Godlewska, M.—Wiszniewski, B.: Smart Email: Almost an Agent Platform. In: Sobh, T., Elleithy, K. (Eds.): Innovations and Advances in Computing, Informatics, Systems Sciences, Networking and Engineering. Springer-Verlag, Berlin, Lecture Notes in Electrical Engineering, Vol. 313, 2015, pp. 581–589.

[5] Godlewska, M.: Reliable Document-Centric Processing and Choreography Policy in a Loosely Coupled Email-Based System. International Journal on Advances in Intelligent Systems, Vol. 9, 2016, No. 1-2, pp. 1–13.

[6] Russell, N.—ter Hofstede, A. H. M.—van der Aalst, W. M. P.—Mulyar, N.: Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, 2006.

[7] MeNaID. National Science Center, Poland, Grant DEC1-2011/01/B/ST6/06500, 2012-2014. Available on: `http://menaid.org.pl`, 2017.

[8] Phelps, T. A.—Wilensky, R.: Toward Active, Extensible, Networked Documents: Multivalent Architecture and Applications. Digital Libraries (DL '96), 1996, pp. 100–108, doi: 10.1145/226931.226951.

[9] Dourish, P.—Edwards, W. K.—LaMarca, A.—Lamping, J.—Petersen, K.—Salisbury, M.—Terry, D. B.—Thornton, J.: Extending Document Management Systems with User-Specific Active Properties. ACM Transactions on Information Systems (TOIS), Vol. 18, 2000, No. 2, pp. 140–170, doi: 10.1145/348751.348758.

[10] Satoh, I.: Mobile Agent-Based Compound Documents. Proceedings of the 2001 ACM Symposium on Document Engineering (DocEng '01), ACM, 2001, pp. 76–84.

[11] Telecom Italia. Workflows and Agents Development Environment. Available on: `http://jade.tilab.com/wade`, 2017.

[12] Telecom Italia. Java Agent Development Framework. Available on: `http://jade.tilab.com`, 2017.

[13] WfMC. Workflow Management Coalition: Process Definition Interface – XML Process Definition Language (Version 2.2). Technical Report WFMC-TC-1025, 2012.

[14] AMODIT Web Site. Available on: `http://amodit.com/`, 2017.

[15] Enembreck, F.—Barthès, J. P.: Agents for Collaborative Filtering. Cooperative Information Agents VII, 7th International Workshop Proceedings (CIA 2003), Helsinki, Finland, August 2003, pp. 184–191, doi: 10.1007/978-3-540-45217-1_14.

[16] Apple Web Site. Automatically Re-Open Windows, Apps, and Documents on Your Mac. Available on: `https://support.apple.com/en-us/HT204005`, 2018.

[17] Cache My Work Web Site. Available on: `http://cachemywork.codeplex.com/`, 2017.

[18] SmartClose Web Site. Available on: `http://bmproductions.fixnum.org/smartclose/index.htm`, 2017.

[19] Google Drive Web Site. Available on: `https://www.google.com/drive/`, 2017.

[20] One Drive Web Site. Available on: `https://onedrive.live.com/about/pl-pl/`, 2017.

[21] Apple Siri Web Site. Available on: `https://www.apple.com/ios/siri/`, 2017.

[22] Google Assistant Web Site. Available on: `https://assistant.google.com/`, 2017.

[23] Amazon Alexa Web Site. Available on: `https://developer.amazon.com/alexa/`, 2017.

[24] Rao, A. S.—Georgeff, M. P.: BDI Agents: From Theory to Practice. Proceedings of the First International Conference on Multiagent Systems (ICMAS '95), 1995, pp. 312–319.

[25] Bratman, M. E.: Intention, Plans, and Practical Reason. Harvard University Press, Cambridge, MA, 1987.

[26] Kaczorek, J.—Wiszniewski, B.: Document Agents with the Intelligent Negotiation Capability. Knowledge and Cognitive Science and Technologies (KCST 2015), Proceedings of the 19th World Multiconference on Systemics, Cybernetics and Informatics (WMSCI 2015), Orlando, FL, USA, July 12–15, 2015, pp. 353–358.

[27] WfMC Workflow Management Coalition: Terminology and Glossary, WfMC, Winchester, UK, Technical Report WFMC-TC-1011, Issue 3.0, 1999.

[28] Foundation for Intelligent Physical Agents: FIPA Agent Message Transport Service Specification, Geneva, Switzerland, 2000.

[29] Spira, J. B.—Burke, C.: Intel's War on Information Overload: A Case Study. Basex, Inc., 2009.

[30] Kaczorek, J.—Wiszniewski, B.: Augmenting Digital Documents with Negotiation Capability. 13th ACM Symposium on Document Engineering (DocEng 2013), Florence, Italy, 2013, pp. 95–98, doi: 10.1145/2494266.2494305.

[31] Landowska, A.—Szwoch, M.—Szwoch, W.: Methodology of Affective Intervention Design for Intelligent Systems. Interacting with Computers, Vol. 28, 2016, No. 6, pp. 737–759, doi: 10.1093/iwc/iwv047.

[32] Bray, T.—Paoli, J.—Sperberg-McQueen, C. M.—Maler, E.—Yergeau, F.: Extensible Markup Language (XML) 1.0 (Fifth Edition). World Wide Web Consortium, Recommendation REC-Xml-20081126, 2008.

[33] Wiszniewski B.: Interactive Documents for Network Organisations. Adjacent Digital Politics, Ltd., 2013.

[34] Siciarek, J.—Smiatacz, M.—Wiszniewski, B.: For Your Eyes Only – Biometric Protection of PDF Documents. 2013 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE '13), Las Vegas, USA, 2013, pp. 212–217.

Magdalena Godlewska received her M.Sc. in computer science from University of Gdansk. She received her Ph.D. degree from Gdansk University of Technology also in computer science. Area of her interest, in general, is document engineering. She got two scholarships for Ph.D. students: the first one awarded by the Poland Pomeranian Special Economic Zone and the second one co-financed by the European Union. She participated in Grant "Methods and Tools of Future Document Engineering – MENAID" – financed by the Poland National Science Centre. She is currently working at the Institute of Informatics at the University of Gdansk.

Computing and Informatics, Vol. 37, 2018, 693–716, doi: 10.4149/cai_2018_3_693

STREAMED DATA ANALYSIS USING ADAPTABLE BLOOM FILTER

Amritpal Singh, Shalini Batra

Department of Computer Science and Engineering
Thapar university
Patiala, Punjab, India
e-mail: amritpal.singh203@gmail.com, sbatra@thapar.edu

Abstract. With the coming up of plethora of web applications and technologies like sensors, IoT, cloud computing, etc., the data generation resources have increased exponentially. Stream processing requires real time analytics of data in motion and that too in a single pass. This paper proposes a framework for hourly analysis of streamed data using Bloom filter, a probabilistic data structure where hashing is done by using a combination of double hashing and partition hashing; leading to less inter-hash function collision and decreased computational overhead. When size of incoming data is not known, use of Static Bloom filter leads to high collision rate if data flow is too much, and wastage of storage space if data is less. In such cases it is difficult to determine the optimal Bloom filter parameters (m, k) in advance, thus a target threshold for false positives (f_p) cannot be guaranteed. To accommodate the growing data size, one of the major requirements in Bloom filter is that filter size m should grow dynamically. For predicting the array size of Bloom filter Kalman filter has been used. It has been experimentally proved that proposed Adaptable Bloom Filter (ATBF) efficiently performs peak hour analysis, server utilization and reduces the time and space required for querying dynamic datasets.

Keywords: Bloom filter, partition hashing, double hashing, Kalmann filter

1 INTRODUCTION

In today's world, data is considered as one of the most valuable assets. It has been acknowledged by data scientists that timely and accurate analysis of available data helps in creating more opportunities by taking right decision at right time in ever changing business environments [1]. With the coming up of plethora of web applications and technologies like IoT [2], cloud computing [3], etc., the data generation resources are increasing exponentially. This change is leading to a shift in paradigm from existing relational data base based systems to systems which can efficiently accommodate Big data. Initially Big data [4] refereed to the collection of huge amount of unstructured data (volume and variety) only. But, with the rise in continuous data generation resources like traffic data, climate data, stock market data, etc., term velocity was introduced in Big data. Streamed data arriving from various resources requires fast processing and storage framework for handling huge amount of data. Storing entire data requires lot of memory and usage of fixed size data structures will require a lot of time for analysis [5]. Further, in case of streams, continuous analysis of data is required before storing it [6, 7].

Streaming data analytics [8] focuses on reducing time and space complexity of incoming data before storing it on disk. The important issue in stream processing is that data diminishes with time so data must be processed in a particular time window in single pass.

In many applications, fast and real time processing is required to make timely decisions accurately [9] for example in finance sector, the analysis of stock market streaming data is an essential tool for predicting stock price of the companies and real time fraud detection in a short time span. In dynamic recommender applications, processing of streamed data is necessary for referral of products according to interest of user and promotion of new products in the market [10]. In network applications, managing data streams for system monitoring can be time-varying, volatile and unpredictable since tasks to be managed include accessing the server's utilizations in particular time frame, tracking the number of unique visitors on a network in a particular time, identifying common users between two time slots, or calculating maximum number of hits on network in particular span of time. Results of network analysis can help to predict the resource usage over network, identify rush hours in network, management of network resources on the time slot basis and detect attacks like DoS and DDoS.

Fast matching of arbitrary identifiers to the values of incoming data and real time response are the basic requirements for majority of streaming data applications. Given millions or even billions of data elements, developing efficient solutions for storing, updating, and querying them becomes increasingly important especially when data is available for a short span.

Using traditional data base approaches which include performing filtering and analysis after storing the data is not efficient for the real time processing of streamed data. Since the size of incoming data is unpredictable, data structures used for the storage of data should be dynamically adjustable, but changing size in each iteration may lead to the extra computational overhead. Thus, some adaptive storage mechanism is required which performs predictive analysis to determine size of data structures being used. Provisions should also be available for adjustments on the basis of previous incoming data or on the basis of real time data flow.

Above mentioned issues clearly indicate that efficient storage and searching techniques are required for processing streaming data. Various solutions proposed by researchers in this domain utilize probabilistic techniques to reduce information processing and analytics cost. This paper proposes the use of Bloom filter [11], a probabilistic data structure [12] which can store the elements of a set in a space-efficient manner by using hashing principles with a small error in querying process. Presently Bloom filter is widely used in many networking and security algorithms like authentication, tracebacking, IP tracebacking, string matching, reply protection, etc. It is also used in fields as diverse as accounting, monitoring, load balancing, policy enforcement, routing, clustering, security and even in many database applications [13]. There are number of variants of Bloom filter which have been successfully used in different application domain [14].

The prime focus of the proposed framework is to efficiently query the incoming data within the limited time domain. To deal with instream data and store the information for a short time the proposed framework uses a Scalable Bloom filter [15] with Ageing Bloom filter properties, i.e. evicting data after fixed time interval. In the proposed framework Kalman filter [16] is used to make scalable Bloom filters adaptive in terms of size and reduce the computational overhead of adding extra filters at run time. Further, query complexity of dynamic data has also been reduced. The proposed Bloom filter is named as Adaptable Bloom Filter (ATBF) and it has been experimentally proved that the proposed filter outperforms the existing Scalable Bloom filter when dealing with in-stream data.

The plan of this paper is as follows: Section 2 provides the literature survey of the Bloom filter and its variants. In Section 3, proposed approach is discussed in detail. Section 4 provides experimental results and compares existing approach with the proposed approach. Finally, Section 5 concludes the paper.

2 RELATED WORK

2.1 Standard Bloom Filter

The Bloom Filter (BF) [11], a space efficient probabilistic data structure, is used to represent a set S of n elements. It consists of an array of m bits, denoted by $BF[1, 2, \ldots, m]$, initially all set to 0. To describe the elements in the set, the filter uses k independent hash functions $h_1, h_2, \ldots, h_k$ with their value ranging between 1 to m assuming that these hash functions independently map each element in the universe to a random number uniformly over the range. For each element $x \in S$; the bits $BF[h_i(x)]$ are set to 1 for $1 < i < k$. Given an item y, its membership is checked by examining the BF whether the bits at positions $h_1(y)$; $h_2(y)$; $\ldots$; $h_k(y)$ are set to 1. If all $h_i(y)$ $(1 < i < k)$ are set to 1, then y is considered to be part of S. If not, then y is definitely not a member of S. The accuracy of a Bloom filter depends on the filter size m, the number of hash functions k, and the number of elements n.

User can predefine false positives (f_p) according to application's requirement.

$$f_p = \left(1 - e^{-kn/m}\right)^k. \tag{1}$$

2.2 Scalable Bloom Filter

In some applications like in-stream data coming from sensors, network traffic, etc., the data is generated dynamically and size of the data set being generated cannot be determined a priori. When size of incoming data is not known, use of static Bloom filter will either lead to high collision rate or result in wastage of storage space. In such cases it is difficult to determine the optimal Bloom filter parameters (m, k) in advance, so a target false positives threshold cannot be guaranteed. In order to accommodate the growing data size, one of the major requirements in Bloom filter is that filter size m should grow dynamically.

Dynamic and scalable Bloom filters deal with the scalability problem by adding bit arrays of varying sizes as the incoming data increases. In Dynamic Bloom Filter (DBF), an array of size same as that of initial array, i.e. an array of m bits, is added repeatedly to accommodate the ever rising data, once the threshold or the fill capacity of the existing DBF exceeds. But this addition in DBF causes the significant increase in error rate. In scalable Bloom filter, a variable size array is added whenever the defined threshold is crossed, with an extra parameter ρ to maintain the error rate in defined bound.

Scalable Bloom Filter (SBF) [15, 17] is a BF variant that can adapt dynamically to the number of incoming elements, with an assured maximum false positives f_p^0. In addition to the initial array of size m_0, SBF includes two additional parameters: expected growth rate (s) and the error probability tightening ratio (ρ) ($0 < \rho < 1$); insert operation in SBF for an element x using k hash functions and parameters s and ρ is given by $Insert(SBF[.], x, {}_{i=1}^{k} h_i(x), f_p^0, s, \rho)$. When $m_0 = \log_2(f_p^0)^{-1}$ exceeds the defined threshold, a new array $m_1 = m_0 + \log_2 \rho^{-1}$ is added and error probability for new filter $f_p^1 = f_p^0 \rho$. Size of additional i^{th} array m_i is:

$$m_i = \log_2(f_p^i)^{-1} = m_0 + i \times \log_2 \rho^{-1}. \tag{2}$$

For flexible growth in SBF size, exponential growth factor s is added, generating i individual filters of size $m_0, m_0 s, m_0 s^2, \ldots, m_0 s^{i-1}$. When the fill ratio t_h for one filter exceeds the defined threshold, another filter is added to it with a well defined growth parameter s. Elements stored in i^{th} filter are approximately:

$$N_i \approx m_0 s^i (\ln(t_h)). \tag{3}$$

At a given time, error probabilities of all i individual filters ($0 < i < (i-1)$) is $f_p^0, f_p^0 \rho, f_p^0 \rho^2, \ldots, f_p^0 \rho^{i-1}$. The compounded error probability for the SBF is:

$$f_p^{SBF} = 1 - \prod_{x=1}^{i} \left(1 - f_p^0 \rho^{x-1}\right). \tag{4}$$

Query process in SBF is accomplished by testing the presence of query element in each filter, starting from active filter to oldest filter. At the time of query, if N be the total incoming elements and N_0 be the elements in m_0, total number of arrays added in Bloom filter. Search complexity for the worst case analysis is:

$$O(k(\lfloor \log(N/N_0 + 1) \rfloor) + 1). \tag{5}$$

It has been experimentally verified that computational overhead of SBF surges as the size of SBF grows with the increase in incoming data set.

2.3 Ageing Bloom Filter

Some network applications require high-speed processing of packets. For this purpose, Bloom filters array should reside in a fast and small memory. In such cases, due to the limited memory size, the stale data in the Bloom filter should be deleted to make space for the new data.

To accommodate such type of issues, number of solutions are proposed by domain experts, one of these is using only one buffer [18], i.e. allocating a buffer for insertion of elements coming from a particular network stream. For each new element, the buffer can be checked, and the element may be identified as distinct if it is not found in the buffer, and duplicate otherwise. When the buffer reaches its fill ratio, whole data is evicted from the buffer, i.e. buffer is reset to original value. Search time complexity and false positive rate in this case is determined for a particular interval, same as that of the Bloom filter. Another solution proposed for aging scheme using similar concept is double buffering [19]. In this approach, concept of buffering is used but with two filters. Initially data is filled in the first filter and once the threshold exceeded, data is filled in the next filter but as soon as the threshold of the second Bloom filter is crossed, data is evicted from the first filter and this process continues. Advantage of this approach is that we can store data for more time by using double memory than by simple buffering approach. Example of aging Bloom filter includes techniques like A^2 buffering where one buffer is divided into two parts and then double buffering is performed. One of the short comings of this approach is that size of the filter used is static, and rough prediction of size of the filter required may affect the accuracy of membership query.

2.4 Partition Hashing

Partitioning hashing is a technique where small portion of large table is uniquely allocated to each hash function such that hash key l_i generated by hash function ℓ_i, is randomly distributed over a small part of the array, i.e., each hash function is allocated to a sub part of an array [20]. In Bloom filter, an array of m bits is partitioned into k disjoint arrays of size $\vartheta = \frac{m}{k}$ bits and k hash functions are used corresponding to each part. For an element $u_i \in U$, hash functions are calculated as:

$$\kappa_i(u_i) = \ell_1(u_i) + i \times \ell_2(u_i) \bmod \vartheta. \tag{6}$$

Each hash function $\kappa_i(.)$ changes bit in i^{th} array where $i|1 < i < k$. Two independent hash functions $\ell_1(x)$ and $\ell_2(x)$ are used to generate k hash functions such that $\forall i|i < k$. To get best results of this schema, ϑ should be prime; so size of array and number of buckets should be choosen in such a way that $\frac{m}{k}$ returns a prime number. This technique leads to less inter-hash function collision, further usage of only two hash functions to generate all k hash functions decreases the computational overhead [20].

2.5 Approximate Counting in Streaming Data

Besides the batch processing infrastructure of map/reduce, Big data analytics require techniques where streamed data is processed in near real time in single pass for some specific applications.

Approximate counting problem and solution for large data sets was defined in 1981 by J. S. Moore in the Journal of Algorithms [21], and later many solutions were proposed using approximate counting in massive data sets. Two solutions are considered: Counter-based algorithms which include Frequent majority [22], LossyCounting [23], SpaceSaving [24] and Sketch based algorithms like Count-Min Sketch [25], Count Sketch [26], etc.

Manku et al. [23] proposed lossy counting algorithm, which divides large data into B_i buckets and calculates the frequency of different type of elements. Count is maintained in bucket counters C_{Bi} for only those elements which cross a defined threshold. For adding a new bucket B_n, counter of previous bucket, i.e. $C_{B_{n-1}}$, is used as base. Random decrement of all counters on the extreme sides is done after the calculations are performed on each bucket.

To answer frequency queries and reduce computational complexity, a sketch data structure named CountMin Sketch (CMS) was proposed by Muthukrishnan and Cormode in 2003 and later improved in 2005 [25]. This data structure is based on probabilistic techniques which are used to answer various types of queries on streaming data. It is a histogram which stores elements and their associated counts. Major difference between Bloom filter and CMS is that Bloom filter effectively represents sets, whereas the CMS considers multisets instead of storing a single bit to answer a query, the count min sketch maintains a count of all object. It is called a 'sketch' because it is a smaller summarization of a larger data set. The probabilistic component of CMS provides more accurate results compared to proposed sketching algorithm solutions as it has less space complexity and decreased computational cost.

2.6 Kalman Filter

Kalman filter (KF) is a linear system model derived from stochastic process, making it ideal for systems which are continuously changing. In KF recursive approach is used where a common model is formulated and all future calculations are performed on the same equations without any modification. It is easy to implement and requires less memory since it does not keep record of old data except the previous state. Further, less computational cost makes it suitable for real time problems.

Notaions	Description
$\hat{x}_k$	Posteriori state estimate
$\hat{x}_k^-$	Priori state estimate
$\hat{P}_k^-$	Priori estimated error
$\hat{P}_k$	Posteriori state estimate
K	Kalman gain
v_k	Measurement noise
R	Co-variance for measurement noise
w_k	Process noise
u_k	Control signal
Q	Co-variance for process noise
z_k	Measured value
A, B, H	Constants according to process

Table 1. Nomenclature for Kalman filter

Kalman filter is a powerfull mathematical tool mainly used for stochastic estimation from noisy sensor data or data streams occurring at regular intervals. The basic assumption of Kalman filter is that system should be continuous and can be modeled as a normally distributed random process X, with mean μ and variance σ (the error covariance), i.e. $X \sim N(\mu, \sigma)$ [16]. Kalman filter addresses the problem of estimation of state x_k of a discrete-time controlled process on the basic of previous state x_{k-1} using following equation:

$$x_k = A.x_{k-1} + B.u_k + w_{k-1} \tag{7}$$

with a measured value z_k for k^{th} state given by:

$$z_k = H.x_k + v_k \tag{8}$$

where $Pr(w) \sim N(0, Q)$ and $Pr(v) \sim N(0, R)$.

2.6.1 Discrete Kalman Filter

Kalman filter is a set of mathematical equations that build a predictor-corrector type estimator model to optimally minimize the estimated error covariance. It provides an estimate of a process for k^{th} state by using a feedback control model. In this, filter first estimates the value for k^{th} state based on the current information of the process and then obtains feedback from some measured value, i.e. noisy input. Based on the error in estimated value, Kalman gain is calculated which helps in minimizing error in further iterations. The algorithm converges to the near optimal result after few iterations.

Kalman filter is divided in two groups: time update equations and measurement update equations. The time update equations help in projecting the priori current state value ($\hat{x}_k^-$) and priori error covariance estimates ($\hat{P}_k^-$) for the next step. The time update equations act as predictor equations for estimation model [27].

$$\hat{x}_k^- = A.\hat{x}_k + B.u_k, \tag{9}$$

$$\hat{P}_k^- = A.\hat{P}_k.A^T + Q. \tag{10}$$

The measurement update equations provide feedback to the time update equations for incorporating new measurement in priori estimate to obtain an improved posteriori estimate. The measurement update equations are also known as corrector equations.

$$\hat{x}_k = \hat{x}_k^- + K_k.(z_k - H.\hat{x}_k^-), \tag{11}$$

$$K_k = \frac{\hat{P}_k^-.H^T}{H.\hat{P}_k^-.H^T + R}, \tag{12}$$

$$\hat{P}_k = (1 - K_k.H)\hat{P}_k^-. \tag{13}$$

Wiener filter deals with static data only; Kalman filter, a generalization of Wiener Filter [28] allows dynamic data with noisy parameters as input. Predictor model based on polynomial regression [29] uses combination of number of linear regression models which increases the computational complexity of calculations for each prediction manifolds. Extended Kalman Filter (EKF) [30] is an extension of Kalman filter, where at each step non-linear system is transferred to linear system by calculating first and second order derivative. Generally, EKF are considered for multi-class problems.

Simplicity of Kalman filter in implementation, less memory requirement and support for dynamic environment makes it a wonderful candidate for predicting the size of Bloom filter in streaming data.

3 ADAPTABLE BLOOM FILTER (ATBF)

To perform timely analysis on streaming data, an adaptive data structure is required which performs analysis in one pass with minimum computational complexity and less storage overhead. For a stream of network data $S : (x_1, x_2, \dots, x_n)$ over a time based window of h time slots i.e. $T : (t_1, t_2, \dots, t_h)$, this paper addresses the following points:

- Analysis of network traffic for a particular time slot.
- Predicting amount of in-coming data in the next slot.

- Allocation of memory for the next time slot based on prediction in the present time slot.

Proposed model is hybrid of two types of Bloom filters: scalable Bloom filter (for dynamic data input) and ageing Bloom filter (store data for particular time interval only). In the proposed framework, an efficient learning model is propounded for a time slot based analysis of network traffic using a novel technique called Adaptable Bloom Filter (ATBF), a variant of scalable Bloom filter. Figure 1 provides the basic framework and coming section elaborate the proposed framework along with its phases.

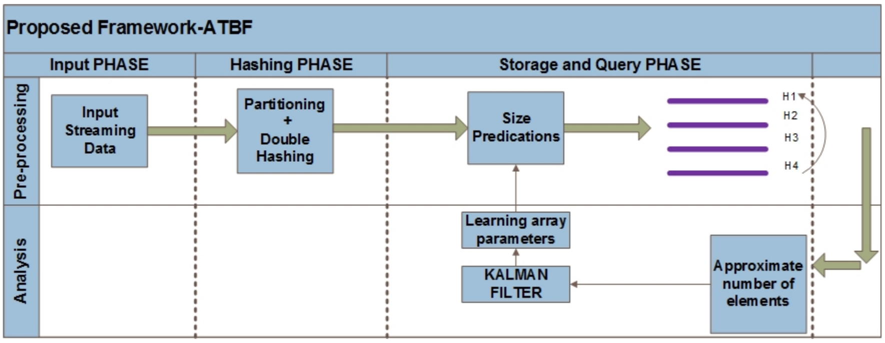


Figure 1. Proposed framework

3.1 Input and Hashing Phase

A stream of data $S = (x_1, x_2, \dots, x_n)$ coming from any resource like sensor, social networking websites, network data and mobile data, etc., is assumed to be the input for the proposed framework. It is assumed that data is available only for limited time and hence it has to be processed in the single pass in the defined time frame. Data may be in varied formats like IP address for network data, website names, email address, etc.

In the proposed scheme, the format of the incoming data is not an issue as all the inputs irrespective of the format (numeric, alphanumeric, text) are hashed using a combination of double hashing and partition hashing. Two independent hash functions $h_1(x)$ and $h_2(x)$ are used to generate k hash functions such that each hash function has a disjoint range of $p = m/k$ (p must be prime for efficient hashing) consecutive bit locations (bucket) instead of having one shared array of m bits, i.e., partitioning hashing is used, where m is size of array, k is number of hash functions and p is prime number denoting the buckets in an array. $\forall i | i < k$

$$g_i(x) = \{h_1(x) + i \times h_2(x)\} \bmod p. \tag{14}$$

To achieve uniformity in maintaining bucket for each hash function at runtime, a parameter σ^i has been introduced, with initial value $\sigma^0 = p$. For each element $x \in S$ $(\forall i | i < k)$

$$H_i^j(x) = (h_1(x) + (i-1)h_2(x)) \bmod \sigma^j. \tag{15}$$

Corresponding to j^{th} slice added in i^{th} slot of ATBF, new σ^j is defined to synchronize bucket size for each hash function. $\Phi(x)$ function returns an optimal number p s.t. $p \leftarrow \Phi(p \geq x$ and p is prime).

3.2 Storage

After hashing is done for each incoming element, i.e. $\forall x_i \in S$, next task is to store the data in the array for a defined time slot say one hour or two hours. For each time slot, i.e. $t_i \in T$, a Bloom filter ($ATBF_i[]$) is maintained to store the elements for that particular time slot.

Selection of initial size of the Bloom filter for each slot in every iteration is critical task because it affects time and query complexity. For the very first iteration, an array of size m_0 is allocated and for further time slots size of Bloom filter is decided based on data received in the previous slot. Initial array size for each time slot t_h is decided on the basis of number of elements accommodated in previous slot t_{h-1}, using an array called Learning Array (LA) which keep the track of size of Bloom filter in each slot. The intent of providing an additional counting array is to reduce the slice addition overhead at the run time. The size of the array required for the next time slot is predicted through Kalman filter and each slot is provided the required slices at the beginning in the form of a single array instead of multiple chunks called slices. This process helps in adjusting the size of $ATBF_i[]$ to accommodate the dynamic input and reduces search time since query is done on single array instead of slices where we traverse from latest to oldest slice one by one. To maintain the uniformity in the partition hashing, size of slice is decided on the basis of $\phi()$ function. Insertion is performed by setting all hash indexed values one in the active slice of filter.

After t time slots when maximum number of time slot for which data records are maintained is reached, insertion is performed in first slot, i.e. $ATBF_1[]$, by evicting its old data. The proposed model works in round robin manner, i.e., slots after h hours perform insertion on same array during next iteration, i.e., $(t_i + o \times h) \leftrightarrow t_i$ where $o \in Z$. After completion of insertion in each time slot $InsertLA()$ function is invoked to update the values for performing size estimation for next time slot (Algorithm 1).

One of the major issues in SBF is how to measure the defined threshold for addition of the new slice. Number of solutions have been proposed for this issue which include 50 % percent rule, i.e., threshold is reached, when the maximum number of one's which a Bloom filter can accommodate reaches 50 % of its original capacity; but how to find that a filter is 50 % occupied is again a tedious task.

One of the options is to maintain a counter which increment every time an element is added or the number of one's in filter have to be counted after regular intervals. Another method is to keep the track of false positives after every insertion to check whether the results are within the desired false positive rate or not, but these solutions lead to extra computational overhead as one needs to continuously check when a filter gets saturated and such operations will definitely dilute the very purpose of using Bloom filter.

Proposed scheme addresses the issue of finding threshold for addition of new filter by the usage of buckets generated through partition hashing. Instead of calculating the threshold of the entire array, a function named *CheckFp*(), which uses standard threshold calculation technique, is used to find the threshold value of the randomly chosen bucket. Such technique limits the threshold calculation to a single bucket instead of entire array, reducing the overall computation time. To avoid calling *CheckFp*() after every iteration, a function *Random*() has been defined which returns a random value through which *CheckFp*() function is called, leading to further optimization of the entire process.

For experimental analysis, data is considered for varying time slots, e.g., one time slot is equal to four or six hours, i.e., all the hashed data of first time slot is added to the array $ATBF_{t1}$, data of second time slot moves to array $ATBF_{t2}$ and size of $ATBF_{t2}$ is determined by LA, based on the traffic in t_1 time slot. The proposed approach is flexible enough to accommodate n time slots, with each time slot represented by one array. Based on data stored in these Bloom filters, i.e. $ATBF_{1\ldots t_n}$, further analysis like peak hour analysis, detecting approximate number of users in each time slot and server utilization are performed.

3.3 Query Process in ATBF

To query the occurrence of a particular element in a time window, *Query*() function is used.

$Query(LA[], p, Q, T, _{i=1}^{i=k} H_i)$ in ATBF checks each Bloom filter, i.e. $ATBF_{t_i} | \forall t_i \in T$ from latest array to oldest array, and in each Bloom filter all slices (if added), i.e. from r to 1, are checked corresponding to the queried element. Query process is made fast by calculating hash functions at the run time, i.e. for a particular query, all hash functions are not computed in advance, each hash function is calculated and comparison is performed in defined bucket of hash function. If bit at hash index is one then next hash function is computed and comparison is performed otherwise query process terminates. Query process terminates as soon as first zero is encountered in a bucket and thus time is saved as remaining hash functions for other buckets are not calculated. The query process is terminated successfully if element is found, i.e. all ones are returned (Algorithm 2).

Algorithm 1 Insertion procedure in ATBF

```
1: procedure INSERT($ATBF[], p, S, T,_{i=1}^{i=k} H_i$) ▷ Insert $x_i \in S$ for $t_i \in T$ in $ATBF_i$ array
2:     for $\forall i | i \leq T$ do
3:         $LA[i][][] \leftarrow InsertLA()$.
4:         $r_i \leftarrow 1$
5:         $\sigma^{r_i} \leftarrow \phi((LA[i] \times p)/k)$    ▷ Return optimal prime number according to variable size of filter
6:         $ATBF_i[r] \leftarrow SizeOf(\sigma^{r_i} \times k)$    ▷ Assign initial size to $i^{\text{th}}$ filter
7:         $C_{Slice} \leftarrow 1$
8:     end for
9:     for $\forall x_j \in S$ do
10:        while $t_c == t_i$ do                                   ▷
11:            if $t_h ATBF_i[r] > thresVal$ then
12:                $\sigma^{r_i} \leftarrow \phi((s^{r-1} \times p)/k)$
13:                $r \leftarrow r + 1$
14:                $SizeOf(ATBF_i[r] \leftarrow \sigma^{r_i} \times k)$
15:                $C_{Slice} + +$
16:            else
17:                for $\forall z | z \leq k$ do
18:                    $h_z(x_j) \leftarrow H_z(x_j)$
19:                    $ATBF_i[r](h_z(x_j)) \leftarrow HIGH$
20:                end for
21:            end if
22:            if $Random() == TRUE$ then
23:                $CheckFp(ATBF[r])$
24:            end if
25:        end while
26:        $InsertLA(FR_{LA[i]}, C_{slice})$
27:    end for
28: end procedure
```

Lemma 1. The worst case query time complexity in proposed model for filter with h time slots, assuming r slices in each slot with k hash functions is always less then $O(rhk)$.

Proof. Searching starts with hashing of the query element y, i.e. $\forall i | y \in Q, h_{i=1}^{k}(y) \leftarrow H_{i=1}^{k}(y)$ and corresponding hash indexes are checked for value zero. Query process begins from the latest time slot to the oldest one, i.e. $t_{h\text{to}1}$ and same is followed in search from slices $s_{r\text{to}1}$ in Bloom filter. During search operation when hash indexed value 0 is encountered, searching for that particular array is terminated and previous slice is not searched. In such case, number of evaluated for unsuccessful query, i.e. not finding the element queried, is always less than k hash functions. For a Bloom

Algorithm 2 Querying in proposed framework

```
1: procedure QUERY($LA[], p, Q, T, {}_{i=1}^{i=k} H_i$)
2:     for ∀ Query elments($y$)$|y \in Q$ do
3:         for ∀ Time slots($t$)$|t \in T$ do
4:             for $l = (ATBF_t[.] \ldots 1)$ do
5:                 if $(ATBF_t[l]({}_{i=1}^{i \le k} h_i(y)) == 1)$ then
6:                     ELEMENT FOUND
7:                 end if
8:             end for
9:         end for
10:        ELEMENT NOT FOUND
11:    end for
12: end procedure
```

filter with r slices, it will be always less than $O(rk)$. Thus, for h time slots from $t_{1 \ldots h}$ having r slices each, the worst case query complexity is always less than $O(rhk)$. $\square$

3.4 Learning Array (LA)

Since the amount of incoming data will keep on varying in every time slot, the size of array will change. Calculating the threshold after every addition and providing new slice accordingly in every time slot at run time requires lot of computation which can be saved if record of size of array, i.e. a counter C_{slice}, is maintained which keeps the count of number of slices added in a particular $ATBF_{ti}$ in a particular time slot. Initially a constant size Bloom filter m_0 is allocated for the first time slot and if the incoming data increases, more slices are added and counter c_{slice} is incremented. To make proposed framework adaptive, a Learning Array $LA[value][c]$ is initially added. The main role of LA is to record the array size of $ATBF_{ti}$ after filling of data in each time slot. This helps in predicting the array size required in the next time slot.

With the help of LA an optimal size of $ATBF_i[]$ required for successive time slots is decided. If for a time slots no slices are added, indicating unused Bloom filter bits, then value of LA is decremented for next time slot (Algorithm 3).

To make the functioning of LA more efficient, Kalman filter is used for predicting array size. The approximate number of elements are estimated through Algorithm 3 and the number of slices 'x' added to the initial filter in a particular time slot serves as input parameter to Kalman filter. After observing incoming data patterns for particular t_i, proposed model decides the optimal size required for next time slot, i.e. t_{i+1}, reducing the overhead of slice addition at run time for each time slot, thus improving the search time complexity of $ATBF_i[]$.

Number of slices (s_n) added to a particular time slot is recorded in LA, from this we can compute the total size of filter required for the particular time slot, i.e. $\hat{S}_s$.

The number of elements n_a accommodated by ATBF is given by:

$$n_a \approx m_0 s^i(\ln(t_h)). \tag{16}$$

From the approximate number of elements accommodated, the size of filter, i.e. $\hat{S}_e$, is calculated as:

$$\hat{S}_e = n_a \times k. \tag{17}$$

These two estimates for the size of Bloom filter act as input for Kalman filter and help the framework to predict the approximate size for coming time slots in further iterations.

Since the incoming data is one dimensional, Kalman filter parameters A, B, H, Q and R in Equations (7), (8), (9), (10), (11), (12), (13) have constant values in the proposed model. u_l is assumed to be zero because no control signal is used in the model. $\hat{S}_l$ denotes posterior estimated size and $\hat{S}_l^-$ denotes priori estimated size for i^{th} time slot and for l^{th} iteration. Thus

Time update:

$$\begin{aligned}\hat{S}_l^- &= \hat{S}_{l-1},\\ \hat{P}_l^- &= \hat{P}_{l-1}.\end{aligned}$$

Measurement update:

$$\begin{aligned}K_l &= \frac{\hat{P}_{l-1}^-}{\hat{P}_{l-1}^- + R},\\ \hat{S}_l &= \hat{S}_{l-1}^- + K_l(z_l + \hat{S}_{l-1}^-),\\ \hat{P}_l &= (1 - K_l)\hat{P}_{l-1}^-\end{aligned}$$

where

$$z_l = .5(\hat{S}_s + \hat{S}_e).$$

Lemma 2. Use of Kalman Filter based *LA* in proposed model reduces the query complexity of *ATBF* in handling in-stream data compared to *SBF* by approximate $O(\frac{1}{r})$ i.e. $\approx< O(k)$, where r is number of slices added and k is number of hash functions considered.

Proof. In case of SBF, when an array crosses defined threshold a new slice is added and insertion is performed. Assuming N_s is elements in stream, let us assume SBF needs r slices to accommodate the incoming data. Query process in SBF is accomplished by testing the presence of query element in each filter, starting from active filter to oldest filter. Search complexity for the worst case analysis is $O(k \times r))$.

In *ATBF* first time slot is functionally similar to SBF, but size for next time slot can be predicted using *LA* and Kalman filter. Predicting size for next time

Algorithm 3 Learning array algorithm

```
 1: procedure InsertLA($LA[i], j$)                                  ▷
 2:     if ($j > 1$) then
 3:         if $LA[i] < j$ then
 4:             $LA[i][c] + +$
 5:         end if
 6:     end if
 7:     if ($j == 1$) then
 8:         if $FR_{LA[i]} < thres_{fill}$ then
 9:             $LA[i][] - = 1$
10:             EXIT
11:         end if
12:     end if
13:     $\hat{s}_n \leftarrow LA[i][c]$
14:     $\hat{S}_s \leftarrow m_0 \sum_{i=1}^{s_n}\{i \times \frac{m_0}{k}\}$
15:     $\hat{S}_e \leftarrow Count(ATBF_i[], r)$
16:     Set$\hat{S}_1^- = 0$
17:     Set$\hat{P}_1^- = 1$
18:     for ($l$ : 1 to $\ell$) do
19:         $z_l = .5(\hat{S}_s + \hat{S}_e)$
20:         Time update
21:         $\hat{S}_l^- = \hat{S}_{l-1}$
22:         $\hat{P}_l^- = \hat{P}_{l-1}$
23:         Measurement update
24:         $K_l = \frac{\hat{P}_{l-1}^-}{\hat{P}_{l-1}^- + R}$
25:         $\hat{S}_l = \hat{S}_{l-1}^- + K_l(z_l + \hat{S}_{l-1}^-)$
26:         $\hat{P}_l = (1 - K_l)\hat{P}_{l-1}^-$
27:     end for
28:     $LA[i][] \leftarrow \hat{S}_\ell$
29: end procedure
```

slot leads to decreased computational overhead as addition of new slices at run time is not required. Further, since the size of new array is combination of initial array and additional slices, inter-function collisions are reduced especially when partition hashing is used. From the second time slot onwards the query complexity is always less than $O(rk)$, because from the the second array onwards the number of new arrays added will always be less than r. In best case when no extra slice is added in future time slots, i.e., the input data arrival rate is constant, search complexity is equal to standard Bloom filter $\approx O(k)$. Thus, for the h time slots, search time complexity for $(h-1)$ slots is reduced drastically. □

3.5 Network Traffic Analysis for a Particular Time Slot

The standard algorithms for counting number of element in streams like CMS, probability based counter and DGIM are quite accurate but need lot of extra space and have computational overhead. Proposed model provides a rough estimate of number of elements using Kalman filter.

To calculate the approximate number of elements in a particular time slot t_i, $Count_i(.)$ is used with initial parameters like slices added in the array (r), threshold fill ratio (f_r), number of hash function (k), initial size of filter (m_0) and prime number used in first filter (p). Two methods have been used to calculate the number of elements in a particular time slot and results are verified by both methods (Algorithm 4). In the first method, growth parameter (s) are considered as $s = 2$ for slow growing data and $s = 4$ for fast growing data with optimal threshold t_h value as 50 % same as that considered in SBF [15]. Total number of elements accommodated by Bloom filter (N_i) is given by:

$$N_i \approx m_0 2^i * (.693). \tag{18}$$

The second method is to calculate the total size of the Bloom filter used and then predict the number of elements accommodated by it. Since σ^g is optimal prime for g^{th} slice, i.e., size of bucket and number of buckets are equal to number of hash functions (k), total size of an array with r slice of σ bits, is given by:

$$\text{Number of Slices}(r) \times \text{Size of Slice}(\sigma).$$

Thus total size (t_s) of $ATBF_i$ with r slices is given by:

$$t_s \leftarrow \sum_{g=1}^{r} (\sigma^g \times k). \tag{19}$$

Bits available for insertion in ATBF are determined by threshold fill ratio (f_r). Total available bits t_a are:

$$t_a \leftarrow t_s \times f_r. \tag{20}$$

Thus, maximum number of elements (E_a) accommodated by $ATBF_i$ are:

$$E_a \leftarrow \frac{t_a}{k}. \tag{21}$$

4 OBSERVATIONS AND ANALYSIS

All the experiments have been performed on i7-3612QM CPU @ 2.10 GHz with 8 GB of RAM. To maintain the uniformity in the results *CityHash* 64 bit library is used to compute two hash functions in double hashing. In all experiments five hash functions have been used with initial size of the filter m_0 as 1 285 bits, slice size σ^0 for all the iterations is considered as 275 ($s = \frac{1\,285}{5}$) for first array in all iterations.

Algorithm 4 Approximate number of elements in $ATBF_i[]$

```
1: procedure $Count_i(ATBF_i[], r)$    ▷
2:     Method 1
3:     $N_i \leftarrow \ln(f_r) \times m_0.s^r$
4:     Method 2
5:     for g:1 to r do
6:         $\sigma^g \leftarrow \phi((g \times p)/k)$
7:     end for
8:     $t_s \leftarrow \sum_{g=1}^{r}(\sigma^g \times k)$
9:     $E_a \leftarrow \frac{t_s.f_r}{k}$
10: end procedure
```

4.1 Performance Evaluation of SBF and ATBF

The performance of SBF and ATBF is compared on the basis of computational time taken for hashing, querying and extra slice addition as the incoming data increases. Figure 2 provides a comparative analysis on the basis of hashing complexity of SBF and *ATBF*. In *SBF*, for every input, hash value is computed for all hash functions (k) while in *ATBF* only two hash functions have been used to generate k hash functions, leading to a major decrease in computational overhead.

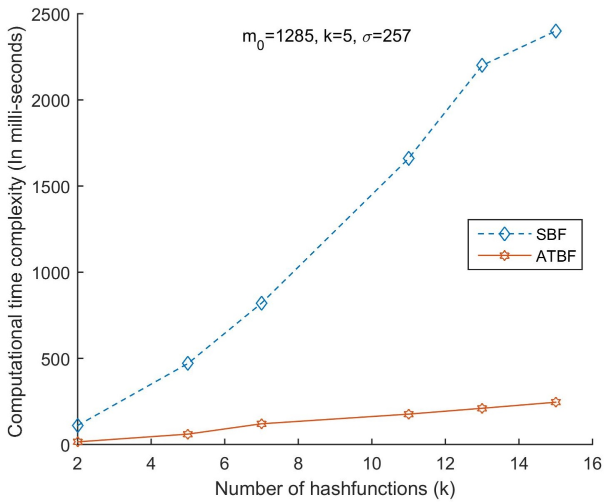


Figure 2. Computational time complexity vs. number of hash functions

Query complexity and slice addition overhead for both the filters is checked on dynamically growing environment. Both filters, i.e. SBF and ATBF, start with the size $m_0 = 1\,285$ and $5\,000$ elements have been considered for the first iteration and each iteration adds $1\,000$ element to previous value.

Figure 3 depicts the analysis performed on the basis of number of slices needed to accommodate the dynamically growing data. In SBF, filter starts with size m_0 and as the number of incoming elements increases more slices are added in each iteration. In case of ATBF, as the incoming data increases the size of t_{n+1}^{th} iteration is predicted in advance, based on the elements accommodated per iteration in t_n^{th} using Kalman filter.

Based upon the data considered for experiments, i.e. $1\,000$ elements increase from previous value per iteration, in t_{n+1}^{th} iteration only one slot is added to accommodate additional elements in ATBF. The graph of ATBF becomes constant after the first iteration since one slice is added in every successive iteration and no overflow of data is registered (Figure 3). Hence overhead of adding new slices at the run time is reduced to a large extent in the proposed scheme.

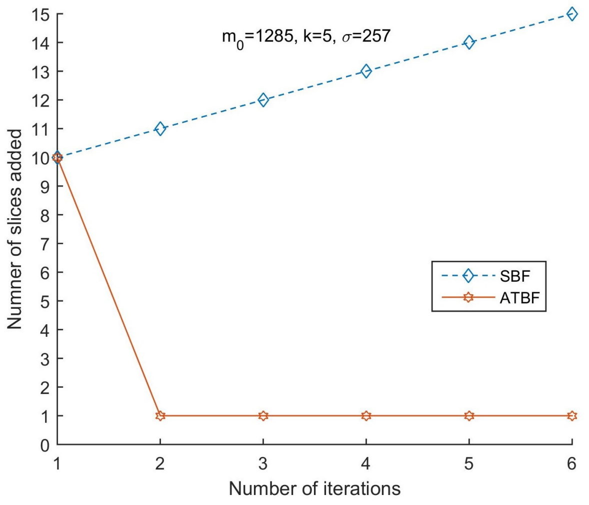


Figure 3. Number of slices required vs. number of iterations for dynamically growing dataset

Figure 4 depicts the comparative analysis of the worst case query complexity for an element (when the element is not present in the set), i.e., scenario where all slices need to be scanned. As the size of data grows in each iteration in SBF, more slices are added to accommodate the data elements. In SBF, all slices need to be scanned in query process which increases the query complexity many folds. ATBF has the advantage of size adaptation from second iteration onwards. For the first iteration the process is similar to SBF, but from the i^{th} iteration (where $i \neq 1$), the size of Bloom filter is predicted on the basis of previous $(i-1)^{\text{th}}$ iteration. The predicted size of Bloom filter is added as a single Bloom filter. So, in querying process only one Bloom filter needs to be scanned, thus the total cost is $O(k)$. As the data grows, the number of slices added are always less than SBF for same number of elements thus search time complexity of ATBF shows a significant improvement.

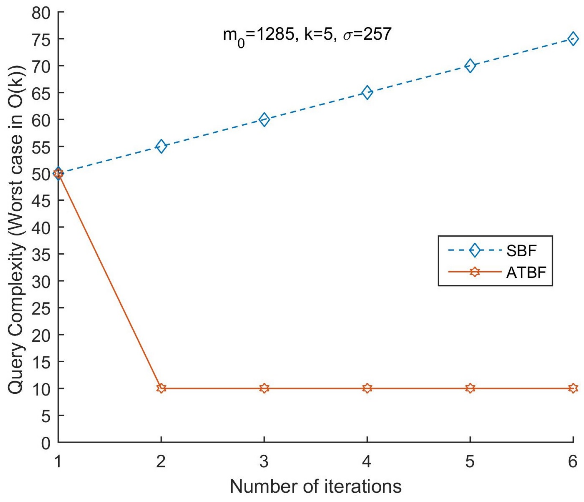


Figure 4. The worst case query complexity vs. number of iterations

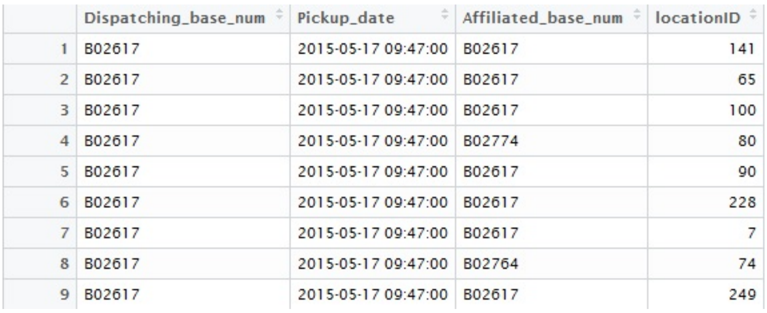
4.2 Experimental Evaluations

Two data sets from different application domains have been considered for evaluating the performance of proposed model, one is data of pickup calls of Uber cabs [31] and other is incoming data generated for network server. Results are represented for first few iterations only, which can be extended to n number of iterations according to application's requirements.

Tables 2 and 3 provide the count of actual number of users and number of users identified using Kalman filter. $\hat{S}^-$ represents the size of Bloom filter in the current iteration by considering the previous one, initially size of Bloom filter is set to m_0. Number of slices added in $ATBF_i$ is maintained by c_{slice} counter. $\hat{S}$ is the array size predicted by Kalman filter for the next iteration. Peak hours analysis is performed by "peak hour ranking" with 1 indicating maximum and 5 as minimum value. Peak hour rank helps in identifying changing patterns of data in current iteration in relation to the previous iteration. Initially for all iteration, the peak hour rank is set to a default value of -1. This ranking system helps in allocating resources in accordance with the frequency of incoming data.

4.2.1 Experiment 1: Uber Pickups Data Sets

Data of 14 270 479 trips of Uber pickups in New York City from January 2015 to June 2015 for around 265 different locations is considered for 12 hours a day as input. The data set is time series based having attributes like date, time, location id and base number. A snapshot of an instance of data is shown in Figure 5. In proposed model "location id" is used as insertion element in Bloom filter and attributes "Pickup_date" and "Time" are used to select the size of a time slot.



	Dispatching_base_num	Pickup_date	Affiliated_base_num	locationID
1	B02617	2015-05-17 09:47:00	B02617	141
2	B02617	2015-05-17 09:47:00	B02617	65
3	B02617	2015-05-17 09:47:00	B02617	100
4	B02617	2015-05-17 09:47:00	B02774	80
5	B02617	2015-05-17 09:47:00	B02617	90
6	B02617	2015-05-17 09:47:00	B02617	228
7	B02617	2015-05-17 09:47:00	B02617	7
8	B02617	2015-05-17 09:47:00	B02764	74
9	B02617	2015-05-17 09:47:00	B02617	249

Figure 5. An instance from data set of Uber pickups

Iteration	**No. of actual users**	**Initial array size** ($\hat{S}^{-}$)	**No. of slots added** (C_{slice})	**No. of users predicted by ATBF**	**Error (In %)**	**Size of Bloom filter predicted for next time slot (in bits)** ($\hat{S}$)	**Previous Peak hour ranking**	**Current Peak hour ranking**
Time Slot = 4 hours								
1/1/2015								
Time Slot 1 (1 to 4) hrs.	5 864	1 285	11	5 746	2.01	28 160	−1	1
Time Slot 2 (5 to 8) hrs.	2 389	28 160	0	2 358	1.3	24 320	−1	3
Time Slot 3 (9 to 12) hrs.	2 922	24 320	0	2 935	−0.4	20 736	−1	2
2/1/2015								
Time Slot 1 (1 to 4) hrs.	1 765	1 285	4	1 732	1.9	8 960	−1	3
Time Slot 2 (5 to 8) hrs.	2 437	8 960	2	2 387	2.1	14 336	−1	2
Time Slot 3 (9 to 12) hrs.	2 534	14 336	0	2 456	−0.9	20 736	−1	1
Time Slot = 6 hours								
1/1/2015								
Time Slot 1 (1 to 6) hrs.	7 314	1 285	12	7 287	0.4	36 660	−1	1
Time Slot 2 (7 to 12) hrs.	2 326	36 660	0	2 342	−0.6	32 256	−1	2
2/1/2015								
Time Slot 1 (1 to 6) hrs.	2 915	1 285	7	2 867	1.7	17 408	−1	1
Time Slot 2 (7 to 12) hrs.	3 312	17 408	0	3 264	1.5	17 408	−1	2

Table 2. Bloom filter size prediction and Peak hour analysis for Uber pickup call for 1st January 2015 and 2nd January 2015

Table 2 shows the result of two days for peak time slot in Uber pickups, for date 1st January 2015 and 2nd January 2015 using two time slot ranges: four hours as a single time slot and six hours as a single time slot, respectively.

4.2.2 Experiment 2: Incoming Data on a Network Server

Table 3 provides the results for server utilization and peak hour analysis. Experiment is done for six time slots of one hour each. The results are simulated on network traffic with maximum per hour capacity of server as 15 000 users. Server utilization is given by ($\frac{n}{N} \times 100$), where n is approximate number of users detected and N is server capacity. The network data has IP address, date and time as its attributes. IP address is used as primary element for insertion in proposed model.

Iteration1	**No. of actual users**	**Initial array size ($\hat{S}^{-}$)**	**No. of slots added**	**No. of users predicted by ATBF**	**Error (In %)**	**Size of Bloom filter predicted for next time slot (in bits) ($\hat{S}$)**	**Server utilization (%)**	**Previous Peak hour ranking**	**Current Peak hour ranking**
Time slot 1	10 000	1 285	15	9 975	0.25	51 200	68.53	-1	2
Time slot 2	12 000	51 200	2	12 145	-1	62 210	82.97	-1	1
Time slot 3	9 000	62 210	0	9 216	-2	46 080	61.44	-1	3
Time slot 4	6 000	46 080	0	6 052	-0.8	32 256	43.01	-1	5
Time slot 5	8 000	32 256	2	7 952	0.6	41 216	54.97	-1	4
Time slot 6	4 000	41 216	0	3 924	1.9	20 736	27.65	-1	6
Iteration 2									
Time slot 1	9 000	1 285	14	8 982	0.2	46 080	60.84	2	3
Time slot 2	14 000	46 080	5	14 248	-1.7	74 240	98.99	1	1
Time slot 3	13 000	74 240	0	13 184	-1	70 400	92.17	3	2
Time slot 4	7 000	70 400	0	7 013	-1	36 352	48.09	5	4
Time slot 5	3 000	36 352	0	2 989	0.4	21 365	23.21	4	6
Time slot 6	4 000	21 365	0	3 968	0.8	20 736	27.65	6	5

Table 3. Hourly analysis of server utilization, the peak hour and Bloom filter size prediction for next time slot for incoming data on a network

5 CONCLUSION

In-stream data analytics works by processing data in a defined time windows. To accommodate dynamic data and query the hourly information, the proposed framework uses Bloom filter with Ageing Bloom filter properties, i.e., evicting data after fixed time interval. Partition hashing has been used which leads to less inter-hash function collision. Further usage of double hashing where only two hash functions are used to generate all k hash functions decreases the computational overhead.

A learning array has been introduced which stores the size of Bloom filter required in the next iteration and Kalman filter has been used to predict the size of Bloom filter required for the next iteration. Results achieved clearly indicate that the proposed framework performs efficiently for the peak hour analysis and server utilization analysis.

Acknowledgment

The first author would like to acknowledge the financial support given to him by the Department of Computer Science and Technology under the Department of Electronics and Information Technology (DeitY) to complete doctoral studies.

REFERENCES

[1] Mayer-Schönberger, V.—Cukier, K.: Big Data: A Revolution That Will Transform How We Live, Work, and Think. John Murray Publishers, UK, 2013.

[2] Gubbi, J.—Buyya, R.—Marusic, S.—Palaniswami, M.: Internet of Things (IoT): A Vision, Architectural Elements, and Future Directions. Future Generation Computer Systems, Vol. 29, 2013, No. 7, pp. 1645–1660.

[3] Amazon. What is Cloud Computing? `http://aws.amazon.com/what-is-cloud-computing/`, 2013.

[4] Krishnan, K.: Data Warehousing in the Age of Big Data. 1st edition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2013.

[5] Labrinidis, A.—Jagadish, H. V.: Challenges and Opportunities with Big Data. Proceedings of VLDB Endowment, Vol. 5, 2012, No. 12, pp. 2032–2033.

[6] Babcock, B.—Babu, S.—Datar, M.—Motwani, R.—Widom, J.: Models and Issues in Data Stream Systems. Proceedings of the Twenty-First ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '02), ACM, New York, NY, USA, 2002, pp. 1–16, doi: 10.1145/543613.543615.

[7] UN Global Pulse: Big Data for Development: Challenges and Opportunities. `http://www.unglobalpulse.org/projects/BigDataforDevelopment/`, 2012.

[8] Liberty, E.—Nelson, J.: Streaming Data Mining. Presented at Princeton University by Yahoo Research Group.

[9] Ylijoki, O.—Porras, J.: Conceptualizing Big Data: Analysis of Case Studies. Intelligent Systems in Accounting, Finance and Management, Vol. 23, 2016, No. 4, pp. 294–310, doi: 10.1002/isaf.1393.

[10] Bobadilla, J.—Ortega, F.—Hernando, A.—Gutiérrez, A.: Recommender Systems Survey. Knowledge-Based Systems, Vol. 46, 2013, pp. 109–132, doi: 10.1016/j.knosys.2013.03.012.

[11] Bloom, B. H.: Space/Time Trade-Offs in Hash Coding with Allowable Errors. Communications of the ACM, Vol. 13, 1970, No. 7, pp. 422–426, doi: 10.1145/362686.362692.

[12] Katsov, I.: Probabilistic Data Structures for Web Analytics and Data Mining. `http://highlyscalable.wordpress.com/2012/05/01/probabilistic-structures-web-analytics-data-mining/`, 2012.

[13] Geravand, S.—Ahmadi, M.: Survey Bloom Filter Applications in Network Security: A State-of-the-Art Survey. Computer Networks, Vol. 57, 2013, No. 18, pp. 4047–4064, doi: 10.1016/j.comnet.2013.09.003.

[14] Tarkoma, S.—Rothenberg, C. E.—Lagerspetz, E.: Theory and Practice of Bloom Filters for Distributed Systems. IEEE Communications Surveys and Tutorials, Vol. 14, 2012, No. 1, pp. 131–155, doi: 10.1109/SURV.2011.031611.00024.

[15] Almeida, P. S.—Baquero, C.—Preguiça, N.—Hutchison, D.: Scalable Bloom Filters. Information Processing Letters, Vol. 101, 2007, No. 6, pp. 255–261, doi: 10.1016/j.ipl.2006.10.007.

[16] Kalman, R. E.: A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME, Journal of Basic Engineering, Vol. 82, 1960, Series D, pp. 35–45, doi: 10.1115/1.3662552.

[17] Xie, K.—Min, Y.—Zhang, D.—Wen, J.—Xie, G.: A Scalable Bloom Filter for Membership Queries. IEEE Global Telecommunications Conference (GLOBECOM '07), 2007, pp. 543–547, doi: 10.1109/GLOCOM.2007.107.

[18] Chang, F.—Feng, W.-C.—Li, K.: Approximate Caches for Packet Classification. Twenty-Third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), 2004, Vol. 4, pp. 2196–2207, doi: 10.1109/INFCOM.2004.1354643.

[19] Yoon, M.: Aging Bloom Filter with Two Active Buffers for Dynamic Sets. IEEE Transactions on Knowledge and Data Engineering, Vol. 22, 2010, No. 1, pp. 134–138.

[20] Kirsch, A.—Mitzenmacher, M.: Less Hashing, Same Performance: Building a Better Bloom Filter. Random Structures and Algorithms, Vol. 33, 2008, No. 2, pp. 187–218, doi: 10.1002/rsa.20208.

[21] Moore, J. S.: A Fast Majority Vote Algorithm. Technical Report ICSCA-CMP-32, Institute for Computer Science, University of Texas, 1981.

[22] Boyer, R. S.—Moore, J. S.: MJRTY – A Fast Majority Vote Algorithm. Automated Reasoning, Springer, 1991, pp. 105–117, doi: 10.1007/978-94-011-3488-0_5.

[23] Manku, G. S.—Motwani, R.: Approximate Frequency Counts over Data Streams. Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02), 2002, pp. 346–357, doi: 10.1016/B978-155860869-6/50038-X.

[24] Metwally, A.—Agrawal, D.—El Abbadi, A.: Efficient Computation of Frequent and Top-k Elements in Data Streams. In: Eiter, T., Libkin, L. (Eds.): Database Theory (ICDT 2005). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3363, 2004, pp. 398–412.

[25] Cormode, G.—Muthukrishnan, S.: An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. Journal of Algorithms, Vol. 55, 2005, No. 1, pp. 58–75, doi: 10.1016/j.jalgor.2003.12.001.

[26] Charikar, M.—Chen, K.—Farach-Colton, M.: Finding Frequent Items in Data Streams. In: Widmayer, P., Eidenbenz, S., Triguero, F., Morales, R., Conejo, R., Hennessy, M. (Eds.): Automata, Languages, and Programming (ICALP 2002).

Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 2380, 2002, pp. 693–703.

[27] Jain, A.—Chang, E. Y.—Wang, Y.-F.: Adaptive Stream Resource Management Using Kalman Filters. Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data (SIGMOD '04), ACM, New York, NY, USA, 2004, pp. 11–22, doi: 10.1145/1007568.1007573.

[28] Wiener, N.: Extrapolation, Interpolation, and Smoothing of Stationary Time Series. MIT Press Cambridge, MA, 1949.

[29] Shaw, P.—Greenstein, D.—Lerch, J.—Clasen, L.—Lenroot, R.—Gogtay, N.—Evans, A.—Rapoport, J.—Giedd, J.: Intellectual Ability and Cortical Development in Children and Adolescents. Nature, Vol. 440, 2006, No. 7084, pp. 676–679.

[30] Julier, S. J.—Uhlmann, J. K.: A New Extension of the Kalman Filter to Nonlinear Systems. Signal Processing, Sensor Fusion, and Target Recognition VI (AeroSense '97). Proceedings of the SPIE, Vol. 3068, 1997, pp. 182–193, doi: 10.1117/12.280797.

[31] FiveThirtyEight. Uber Pickups in New York City. `https://www.kaggle.com/fivethirtyeight/uber-pickups-in-new-york-city`, 2016.

Amritpal Singh received his M.Eng. degree from Thapar University, Punjab, India, with a minor in big data and advanced data structures, in 2013. He is working as Research Scholar with Computer Science Department at Thapar University, Punjab, India since January 2015. He served both industry and academia. His research interests include probabilistic data structures, machine learning and big data.

Shalini Batra received her Ph.D. degree in computer science and engineering from Thapar University, Patiala, India, in 2012. She is currently working as Associate Professor with the Department of Computer Science and Engineering, Thapar University, Patiala, India. She has guided many research scholars leading to Ph.D. and M.Eng./M.Tech. She has authored more than 60 research papers published in various conferences and journals. Her research interests include machine learning, web semantics, big data analytics and vehicular ad-hoc networks.

Computing and Informatics, Vol. 37, 2018, 717–736, doi: 10.4149/cai_2018_3_717

DEFORMABLE OBJECT TRACKING USING CLUSTERING AND PARTICLE FILTER

Muhammad Aasim Rafique, Moongu Jeon*

School of Information and Communications
Gwangju Institute of Science and Technology (GIST)
Gwangju, Republic of Korea
e-mail: {arafique, mgjeon}@gist.ac.kr

Malik Tahir Hassan

University of Management and Technology
Lahore, Pakistan
e-mail: tahir.hassan@umt.edu.pk

Abstract. Visual tracking of a deformable object is a challenging problem, as the target object frequently changes its attributes like shape, posture, color and so on. In this work, we propose a model-free tracker using clustering to track a target object which poses deformations and rotations. Clustering is applied to segment the tracked object into several independent components and the discriminative parts are tracked to locate the object. The proposed technique segments the target object into independent components using data clustering techniques and then tracks by finding corresponding clusters. Particle filters method is incorporated to improve the accuracy of the proposed technique. Experiments are carried out with several standard data sets, and results demonstrate comparable performance to the state-of-the-art visual tracking methods.

Keywords: Visual object tracking, data clustering, object segmentation, cluster correspondence

* Corresponding author

1 INTRODUCTION

Visual object tracking (VOT) has numerous applications in surveillance, intelligent transportation systems, sports broadcasting, robotics and so on. Single object tracking is a base case and, usually, extended to track multiple objects in a scene. Video analysis is affected by the video quality, scene environment attributes (illumination, noise, shadow and jitter), spatio-temporal attributes and behavioral change of intrinsic properties of a target object (such as shape, color and size). The uncertain behavior of the intrinsic properties over the length of a video sequence is deformation of the object.

Single object tracking has defined a work-flow for general VOT cases. The general strategy considers detection of the target region (target object), representation of the target object and activity of the target object. An effective VOT technique is efficient and wise combination of the aforementioned. A brief description of each of the stated component work-flow is worth mentioning. The target region is selected as a preliminary geometric shape such as a quadrilateral or ellipse; these shapes provide with a benefit of handling few parameters and a disadvantage of redundant information besides the target object. Specific contours are used to avoid the ineffective data for demarcation of the target object precisely, but it burdens with many parameters to track along. Adaptive selection of the target region can be an effective way to track deformed objects, but it might come with an associated computational cost.

Second challenge is representation of the target object. The simplest representation is the pixels of target regions as color values. The basic RGB color values are vulnerable to the underlying challenges of video analysis, thus they are not invariant representation of the target object. Histogram of colors is an effective representation for alternating color change in the target region. Expensive features which are invariant to color, rotations and motion are some other choices for the target object representation. Well known features are edges, HAAR-like features, SIFT and SURF features, HoG features, etc. Motion representation of the target object relates its motion within vicinity of the current locality. However, a super fast object can disturb the tracking results drastically. Alternatively, one can model the motion from the initial frames of the video which can, later, be used to predict the location of the target object. Probabilistic Gaussian motion model, Kalman filters and particle filters, optical flow trackers, etc., are commonly used motion models.

In the end, prediction of the target object is required to conclude one step of tracking. Prediction may be as simple as template matching and can extend to complex sophisticated discriminative classifiers. Tracking of deformable objects is a situation, where the object alternatively changes color, shape and scale with motion. These variations make it hard to track deformable objects optimally, as a general case. In this work, we propose a spatio-temporal representation of target object and an optimal method to model the activity of the target object. The spatial representation of the target object is segmented using data clustering techniques, and the temporal representation is given by solving the clustering correspondence problem. Moreover, particle filtering technique is used to model the activity of the target object.

This paper is organized as follows. Section 2 presents the relevant literature survey. In Section 3, we explain our proposed method. Section 4 articulates the evaluation setup used to cross examine and describe our experiments to test the performance of our proposed method, and presents the obtained results with discussion. Conclusion and possible future directions are briefed in Section 5.

2 RELATED WORK

The single object tracking problem is an active research area, and persuasive literature is available for study. Comprehensive evaluations and contemplative discussions, with summaries of most of the interesting techniques, are aggregated in literature for interested users [26, 28, 25]. Another recent review evaluating the single object tracking techniques on different video sequences will be helpful for survey [17]. In addition to the aforementioned references, it will be beneficial to discuss recent progress in the single object tracking domain. Structure-preserving object tracker (SPOT) [30, 29] uses online structured SVM to learn the spatial constraints of different parts of the objects, and it predicts from the candidate windows for object tracking. Lucas-Kanade algorithm [19] is extended as an optimization problem in [23] where the object's pixels and the background segmentation are optimized by applying likelihood of a Bayesian framework. Incremental subspace learning and Fisher discriminant analysis techniques are combined, and a graph based combination is proposed to effectively capture the dynamic appearance of the target object and differentiate it from the background [32]. Another graph inspired technique used graph cut method for object segmentation, and it improved the object tracking results, reported in [31].

Since there are plenty of techniques employing variety of strategies to approach the single object tracking problem a rational thought is to discuss the pertinent literature which follows henceforth. Mean shift is used to find best candidate windows for the target object from the next frame by matching histograms discrimination information from the Bhattacharya coefficients [4]. The target region is divided into static segments of 20×20 pixel values, and each segment is associated with a separate Kalman filter in [22]. Later, the object tracking is performed using template matching. A likely idea is to divide the target object in fragments of fixed size and use the color histogram of these fragments to compare the probable matches from candidate segments with Earth Movers Distance (EMD) [2] to track. A recent work in similar regards is the representation of the segmented target object by a superpixel per segment. A superpixel is defined by the center of mass and average HSV-values [24], and EMD is used for comparisons. The target object state is sampled using particle filter for the segments. Key-points are used with hierarchical clustering techniques for deformable object tracking in [21].

Deformable object tracking has been aimed by many researches from general to specific cases. As discussed in Section 1, the challenges put forth by change of shape, occlusion, motion activities, and so on, recognized the deformable object tracking as a standalone task. A nonlinear model with implicit representation of the target object by contours and defining generative dynamical model for the motion is presented in early literature [13]. The boundary element method is applied with a deformable template to model the displacements, and the template is registered to the image by energy minimization of the force field [8]. Later, the idea is extended with the use of canny edge detector for occlusion [9]. An optical flow equation applied on the whole image with constraints on the elastic deformation is discussed in [12]. Deformable objects are tracked using a sliding window particle filter, where the change in an object's shape is captured using a modified technique of principle component analysis [16].

Dynamic graphs are employed in tracking to represent the geometrical structure of the target and the candidate object as nodes, and their interaction is denoted by edges; Markov random field and spectral clustering is used to solve the target and the candidate graph matching [3]. A recent work used the weightless neural networks for tracking the deformable objects to a success [27]. [18] discussed a path based tracking which overcame the limitation of core reliance on the initialization by intelligently selecting the correct patches. [5] proposed use of hyper-graph for guessing correspondence in deformable object in successive multiple frames, which helped in long-term occlusions and intense deformations. Fusion of the data from multiple sensors used with a multiple Kalman filters tracking technique to improve visual tracking is presented in [15].

In comparison to existing techniques, we propose the use of clustering, an unsupervised technique, to segment the target object into parts, and use these parts wisely to track the object. We keep with us the discriminative parts of the reference (target) object, and estimate the location of matching parts in the vicinity of the object in the previous frame. Moreover, particle filtering is incorporated into the method to make it more robust to the tracking challenges. We shall discuss the formal details of our methodology in coming sections.

3 OUR METHODOLOGY

Formally defining the single object tracking problem: given a sequence of N images I_1, I_2, $\dots$, I_N, and an initializing bounding box ground truth region $b_g = b_1$ in I_1 containing the object to be tracked, we aim at predicting the bounding boxes $b_2, \dots,$ b_N that contain the target object in remaining frames of the sequence I_2, $\dots$, I_N, respectively. The detail of our clustering and particle filter based tracking method TUC (tracking using clustering) is provided in the remainder of this section. We call the target object to be tracked as *tracked object* or *reference object*, and the estimated object as the *predicted object* alternatively.

3.1 Clustering for Object Segmentation

Data clustering discovers groups of similar patterns in data and its application for image segmentation is quite intuitive. In our first step, we obtain k segments of b_g, the initially provided ground truth region in I_1, using k-means clustering method. K-means is chosen for its efficiency and simplicity. Note that although clustering is expensive for large data yet applying it to a usually small region like b_g is not computationally expensive. These k segments of b_g become the reference segments that will be compared with the segments of test regions in next frames to estimate the tracked object's location.

3.1.1 Number of Clusters

Number of clusters k is an input parameter for k-means. We tested different values for k and empirically fixed it to 15 being a good tradeoff between accuracy and efficiency. Figure 1 shows the segments of an object discovered using different values of k.

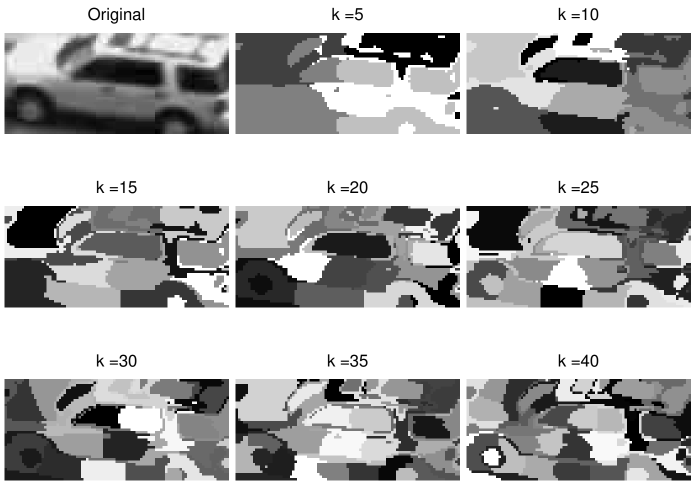


Figure 1. Segments of the object using different number of clusters, $k \in \{5, 10, \ldots, 40\}$

3.1.2 Feature Selection

Feature selection can be regarded as the most important part in any computer vision, machine learning and pattern recognition algorithm in general, and in a tracking method in particular. We segment the object using pixel location, gray intensity, and x- and y-directional gradient values. The separation of salient segments in Figure 1 justifies the suitability of using these features.

3.2 Selecting Discriminative Segments

In practice, the target object's neighborhood may contain textures that are similar to the target object itself and can hinder the tracker's accuracy. Considering the fact that the far regions has less to add to this obstruction, we select the segments of the reference object that are most discriminative from the immediate background. We take four neighboring regions up, down, left and right of the object having same size as the object, and segment each of these regions with same k value (Figure 2). The segments of the reference object b_g that have high similarity with the segments from neighboring regions are removed and not used as reference segments. Thus, we obtain the set of most discriminative segments of the reference object, S_g. We removed the top 25 % most similar segments to the background in our experimentation.

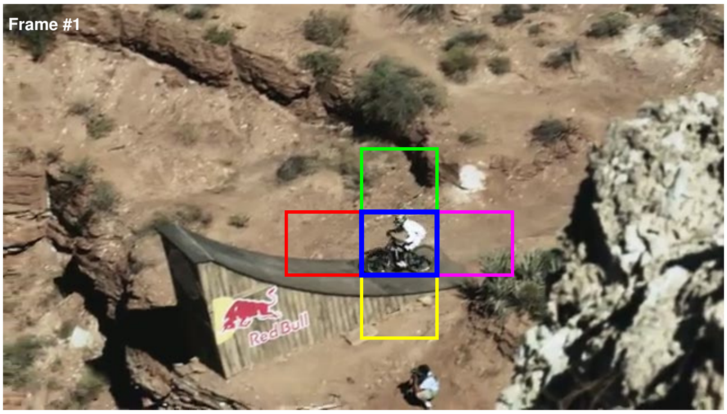


Figure 2. The four background boxes around the tracked object are shown that are used to calculate discriminative segments of the object

3.3 Object Tracking Using Segments

Once we have the discriminative segments of the reference object from the frame I_1, the next step is to locate and track the object in subsequent frames $I_2, \ldots, I_N$. For this, we pick the region b_n in frame I_n where $n \in \{2, 3, \ldots, N\}$ in sequence, using the immediate previous frame's region information, i.e., the location, width and height of the bounding box in frame I_{n-1}. A realistic assumption which will be relieved later is that the object is not moving too fast from I_{n-1} to I_n, and we get some part of the object in b_n to estimate the object's location in I_n. However, such fast motion situations are handled by incorporating particle filter in our method. Detail of using particle filter is presented in Section 3.5.

Thus clustering is applied on region b_n to obtain the set of k segments S_n, and these segments in S_n are then compared with the reference segments in S_g. This comparison, however, demands to solve the segments correspondence problem, which is discussed in detail in Section 3.4. The segments correspondence problem enables us to compute the amount of translation between two corresponding segments by using centroids of the segments. As different pairs of corresponding segments suggest different translation values, we take the median of these translation values and predict the translated location of the bounding box in I_n. Hence, the change in locations of the corresponding segments in S_n and S_g helps us estimate the distance the object has traveled.

3.4 Finding Corresponding Segments

Finding correct corresponding segments in the set of current segments S_n and the set of reference segments S_g is of key importance in our method, and we are able to solve this correspondence problem pretty accurately. Different regional properties of the segments are compared to calculate their similarity. These regional properties include area, eccentricity, Euler number, mean intensity and normalized intensity range of a segment. Area is the number of pixels in a region. Area is computed as actual number of pixels in a segment. Eccentricity specifies the eccentricity of the ellipse that has the same second-moments as the region, analogically it represents how circular the region is. Eccentricity is computed as a ratio of the distance between the foci of the ellipse and its major axis length. A line segment has 1 eccentricity and a circle has 0 eccentricity. Euler number specifies the number of objects in the region minus the number of holes in those objects. Mean intensity is the average intensity value of a region, and normalized intensity range of a region is defined as:

$$\frac{(MaxIntensity - MinIntensity)}{255}.$$

Euclidean distances between each pair of segments is calculated based on these regional properties.

$$\text{dist}(s_i, s_j) = \sqrt{\sum (u_i - v_j)^2}, \quad \forall s_i \in S_n, s_j \in S_g, \tag{1}$$

where u_i and v_j represent the vectors of the regional properties of segments s_i and s_j, respectively.

In addition to this distance calculation of regions, overlap of each pair of segments is also computed using Jaccard index as follows:

$$o(s_i, s_j) = \frac{|s_i \cap s_j|}{|s_i \cup s_j|}, \quad \forall s_i \in S_n, s_j \in S_g. \tag{2}$$

Finally, the similarity of two segments $s_i \in S_n$ and $s_j \in S_g$ is computed as:

$$\text{sim}(s_i, s_j) = \alpha \cdot o(s_i, s_j) + \beta \cdot \frac{1}{\text{dist}(s_i, s_j)}. \tag{3}$$

We fixed α and β values to be 0.25 and 0.75, respectively, based on empirical results. Section 4.5 shows the impact of various combinations of α and β value on the quantified results.

The correspondence solving method returns matching segments in S_n and S_g along with the confidence weights based on similarities of the corresponding segments. Since there exist low similarity pairs of segments, we pick the top 75 % of the segment matches based on these confidence weights, and use them for tracking.

3.5 Incorporating Particle Filter

Particle filtering is used to approximate the intractable distributions for sample generation techniques. It starts by generating a random set of particles and it estimates states and observations for the next time step. It overcomes the limitation of unnormalized and non-gaussian distributions and generate samples using the weighted previous observations. It is interesting to initialize the particles and weights updating strategy, what is a domain specific gimmick.

We incorporate particle filtering into our clustering based tracking method to improve its robustness and to behave well with less accurate clustering. P particles are sampled from a 2-d Gaussian distribution centered at the center of the target object in previous frame, with covariance matrix V. Initial weight to every particle is assigned based on two measures. First, the sum of distances of a particle p to all the centers of the reference object's segments c_i^g; call it w_p^d. Second, the correlation of the reference window b_g and the same sized window centered at the particle b_p; call it w_p^r.

$$w_p^d = \sum_{i=1}^{k} \text{dist}(p, c_i^g), \tag{4}$$

$$w_p^r = \text{corr}(b_p, b_g), \tag{5}$$

w_p^d and w_p^r are normalized by their total sum values and then combined to find initial weight of the particle p as:

$$w_p = \frac{1}{w_p^d} \cdot \exp(w_p^r), \tag{6}$$

w_p is normalized to sum to 1. The estimate for object's motion in current frame is computed using clustering as described in the previous steps of this section. Next step is to move every particle using this estimated amount of motion. Instead of using the single motion value, we sample P motion values from a 2-d Gaussian distribution centered at the estimated amount of motion, and having covariance V_d. Updated weights are calculated again using particles' distance from reference centers and correlation with the reference window. Finally, particle with the maximum weight is picked as center of the target object's new location. Figure 3 gives a small demo of our tracking method by showing the object, the estimated bounding box and the particles.

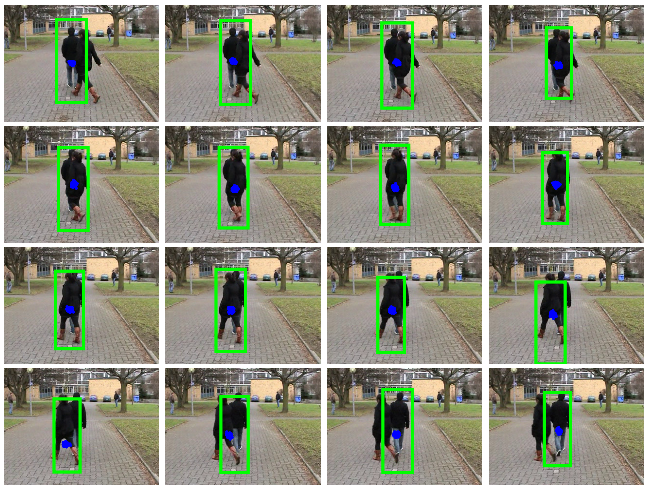


Figure 3. Tracked object (the person moving straight) and particles are shown in 16 consecutive frames from top-left to bottom-right (person crossing data set). Successful occlusion handling is also visible.

3.6 Scale Estimation

The estimation of change in scale of the tracked object is assisted by the nature of our clustering based procedure. Corresponding segments or clusters of the true object b_g and the predicted object b_n are identified, as described in Section 3.4, and the sizes of these corresponding segments in S_g and S_n are compared. The ratio of their sizes gives an estimate of the scale-change factor δ_{scale}. As different corresponding segments give different estimate values, δ_{scale} is set to be the median of these values.

$$\delta_{scale} = \text{median}\left(\frac{|s_i^c|}{|s_j^c|}\right), \quad \forall s_i^c \in S_n, s_j^c \in S_g. \tag{7}$$

The superscript c indicates that these are the corresponding segments of the current and ground truth segments, S_n and S_g, respectively. $|.|$ is the size of the segment calculated as count of pixels in the segment, also known as area. δ_{scale} is used to get the updated width w_n and height h_n of the predicted bounding box b_n.

$$[w_n, h_n] = \sqrt{\delta_{scale}} \cdot [w_g, h_g], \tag{8}$$

where w_g and h_g are the width and height of the ground truth bounding box b_g, respectively.

The steps of our methodology are summarized in Algorithm 1.

Algorithm 1 TUC – *T*racking *U*sing *C*lustering

```
Require: $I_1, \ldots, I_N$ {image sequence}, $b_1$ {bounding box in $I_1$}
1:  $k \leftarrow 15$ {initialize number of clusters}
2:  $P \leftarrow 200$ {initialize number of particles}
3:  $F_1 \leftarrow computeFeatures(b_1)$ {features of ground truth $b_g$}
4:  $S_g \leftarrow kmeans(F_1, k)$ {segments of the object $b_g$}
5:  $S_g^d \leftarrow findDiscriminativeSegments(S_g, I_1)$ {discriminative reference segments, Section 3.2}
6:  for n = 2 to N do
7:      $P_{xy} \leftarrow generateParticles(c_n^0, V, P)$ {Section 3.5 and Equation (11)}
8:      $w_p \leftarrow assignWeights(P_{xy})$
9:      $P_{xy} \leftarrow resample(P_{xy}, w_p)$
10:     $F_n \leftarrow computeFeatures(b_n^0)$ {$b_n^0$ is the box in current frame using previous frame's box information}
11:     $S_n \leftarrow kmeans(F_n, k)$
12:     $MATCHES \leftarrow findCorrespondingSegments(S_n, S_g^d)$ {Section 3.4}
13:     $t_{xy} \leftarrow estimateTranslation(MATCHES)$ {Section 3.3}
14:     $t' \leftarrow generateRandomSpeeds(t_{xy}, V_d, P)$ {Section 3.5 and Equation (12)}
15:     $P_{xy} \leftarrow P_{xy} + t_{xy} + t'$ {move the particles with estimated and random speeds}
16:     $w_p \leftarrow assignWeights(P_{xy})$
17:     $c_n \leftarrow \max(w_p, P_{xy})$ {estimated center of the object}
18:     $b_n^0 \leftarrow boundingBox(c_n)$
19:     $\delta_{scale} \leftarrow estimateScale(b_n^0, b_g)$ {Section 3.6}
20:     $b_n \leftarrow scale(b_n^0, \delta_{scale})$
21:     return $b_n$ {predicted bounding box in $I_n$}
22: end for
```

4 EXPERIMENTAL EVALUATION

We compare our method with state-of-the-art tracking methods on standard data sets using a popular evaluation measure. Our experimental setup and obtained results are discussed in this section.

4.1 Data Sets

Experimental evaluation of our tracking method is carried out on nine standard publicly available data sets.[1] The video sequences in these data sets contain different visual tracking challenges like deformation, in-plane rotation, out-of-plane rotation, scale change, occlusions, etc. Figure 4 shows the first frames of these video sequences and the target object to be tracked.

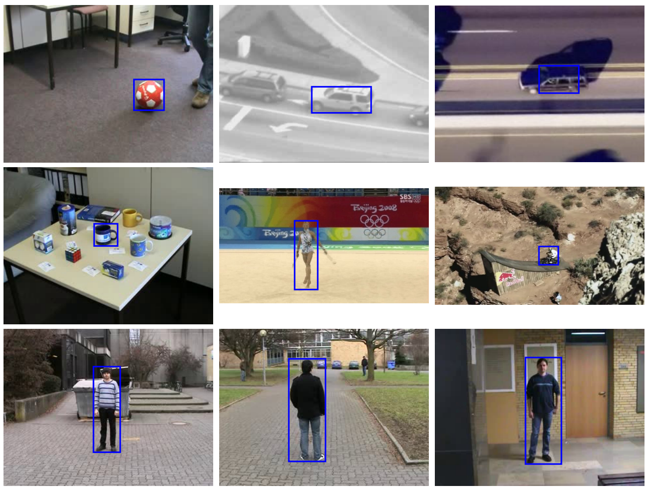


Figure 4. First frame and the ground truth bounding box are shown for each of the nine video sequences used in experimental evaluation. The video sequences from top-left to bottom-right are ball, car2, car chase, cup on table, gym, mountain bike, person, person crossing and person occlusion.

4.2 Evaluation Measure

Many measures exist in literature for quantitative evaluation of tracking methods. The center-error measure expresses the distance between the centroid of the predicted box and the centroid of the ground truth. This measure is not bounded and ignores the scale and the aspect ratio of the bounding boxes. We have selected the commonly used overlap measure:

$$o(b_n, b_g) = \frac{|b_n \cap b_g|}{|b_n \cup b_g|}, \tag{9}$$

where b_n refers to the predicted bounding box and b_g refers to the ground truth bounding box. This measure is bounded between 0 and 1, penalizes translation

[1] `http://www.gnebehay.com/cmt/`

and scale alterations, and is popularly known to be a better indicator for per-frame success [20].

In order to find an overall score for a sequence, a threshold τ is applied on Equation (9) to find true positives (TP). True positive rate (or recall) is then reported for all sequences.

$$\text{recall} = \frac{TP}{TP + FN}. \tag{10}$$

The value of recall gives the percentage of frames that are tracked correctly, i.e. when $o \geq \tau$.

Results are computed for three different values of τ, i.e., 0.25, 0.50 and 0.75. These threshold values are suggested by [20] with an interpretation as low, medium and high requirements on accuracy.

4.3 Comparison Methods

A comparison of our approach is performed with the state-of-the-art tracking approaches. The comparison methods include CMT (Consensus-based Matching and Tracking [20, 21]), STRUCK (Structured output Tracking [10]), TLD (Tracking-Learning-Detection [14]), LM (LearnMatch [11]), FT (Fragments-based Tracking [2]), HT (HoughTrack [6]) and SB (Semi-supervised online Boosting [7]).

4.4 Parameters Setting

Required parameters of our method were set once and then used for all of the data sets consistently. The setting was guided by initial experimental results.

The number of clusters parameter k which becomes the number of tracked segments is set to be 15. Number of particles P is set to be 200. Covariance matrix V for initial random Gaussian particles is set to be

$$V = \begin{bmatrix} 7 & 1 \\ 1 & 7 \end{bmatrix}, \tag{11}$$

and covariance matrix for random motions of the particles V_d is set to be

$$V_d = \begin{bmatrix} 2 & 1.5 \\ 1.5 & 2 \end{bmatrix}. \tag{12}$$

Covariance matrix V is used to generate initial random Gaussian particles. The shape of the target object (width and height of the bounding box) and the dominant direction of motion can help in determining this spread to be more in one direction or other (we fixed to 1 and 7 in our experiments). V controls the spread of particles and can be learned through some initial frames or adapted incrementally (not done in the current work). In the case of V_d, the covariance matrix of random motions, the values are small and almost identical for both horizontal and vertical directions (1.5 and 2).

Sequence	τ	**CMT**	**STR**	**TLD**	**FT**	**LM**	**HT**	**SB**	**TUC**
	0.25	**0.98**	0.30	0.40	0.31	0.14	0.15	0.30	0.90
ball	0.50	0.57	0.15	0.28	0.19	0.12	0.11	0.28	**0.58**
	0.75	**0.19**	0.10	**0.19**	0.13	0.09	0.10	0.12	0.15
	0.25	0.90	0.81	**1.00**	0.04	0.46	0.59	0.72	0.98
car2	0.50	0.88	0.47	**1.00**	0.04	0.36	0.47	0.72	0.94
	0.75	0.64	0.11	**0.95**	0.03	0.17	0.00	0.70	0.72
	0.25	0.30	0.08	0.16	0.04	0.00	0.04	0.08	**0.32**
carchase	0.50	**0.20**	0.03	0.15	0.03	0.00	0.04	0.08	0.13
	0.75	**0.07**	0.02	0.06	0.02	0.00	0.00	0.05	0.04
	0.25	0.83	**1.00**	0.89	**1.00**	0.68	**1.00**	0.47	**1.00**
cup on table	0.50	0.81	0.92	0.64	0.88	0.54	**1.00**	0.47	0.98
	0.75	**0.61**	0.35	0.06	0.40	0.31	0.48	0.34	0.53
	0.25	0.93	**1.00**	0.76	0.24	0.10	0.30	0.61	**1.00**
gym	0.50	0.86	**0.93**	0.32	0.22	0.05	0.00	0.58	0.89
	0.75	0.22	0.3	0.08	0.12	0.02	0.00	0.22	**0.36**
	0.25	0.99	0.99	0.37	0.65	0.11	0.99	0.20	**1.00**
mount-bike	0.50	**0.98**	0.93	0.36	0.63	0.08	0.40	0.17	0.88
	0.75	**0.48**	0.23	0.16	0.18	0.04	0.03	0.08	0.27
	0.25	0.95	**1.00**	0.92	**1.00**	0.75	0.49	0.52	**1.00**
person	0.50	0.82	0.95	0.71	0.95	0.67	0.00	0.52	**0.99**
	0.75	0.49	0.50	0.25	0.54	0.31	0.00	0.40	**0.57**
	0.25	0.76	0.51	0.86	0.88	0.80	0.18	**0.96**	0.87
person-cro	0.50	0.70	0.42	0.70	0.66	0.75	0.10	**0.91**	0.78
	0.75	**0.58**	0.12	0.10	0.15	0.42	0.04	0.16	0.13
	0.25	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	0.99	**1.00**
person-occ	0.50	0.94	0.91	0.87	0.91	**0.95**	0.93	0.91	0.92
	0.75	**0.82**	0.80	0.58	0.80	0.82	0.44	0.80	0.80
	0.25	*0.85*	*0.74*	*0.71*	*0.57*	*0.45*	*0.53*	*0.54*	***0.90***
Average	0.50	*0.75*	*0.63*	*0.56*	*0.50*	*0.39*	*0.34*	*0.52*	***0.79***
	0.75	***0.46***	*0.28*	*0.27*	*0.26*	*0.24*	*0.12*	*0.32*	*0.40*

Table 1. Comparison of our method (last column) with existing methods on 9 video sequences. Recall results are reported for 0.25, 0.50 and 0.75 threshold (τ) values of overlap with the ground truth. The top recall values are highlighted in bold and average values are presented in italic typeface.

4.5 Results and Discussion

Figure 5 shows results of our tracking method obtained using clustering alone, and after incorporating particle filtering and discriminative segments. Improvement in results is visible when particle filtering is added to the simple clustering based tracking. Removing ambiguous segments and keeping discriminative segments only, further improves the tracking accuracy.

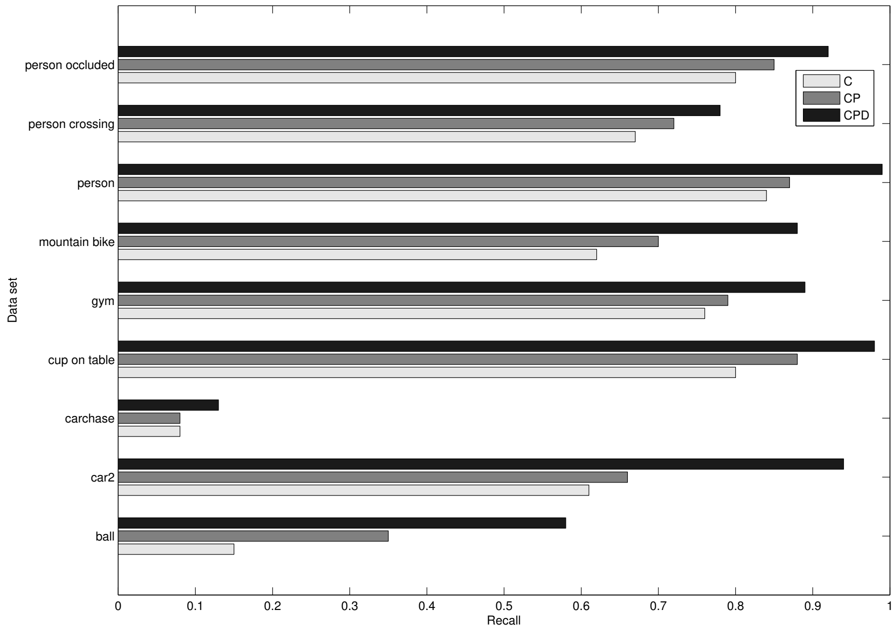


Figure 5. Comparison of our tracking method using clustering (C), and after incorporating particle filtering (CP) and discriminative segments (CPD). Overlap threshold $\tau = 0.50$. Combined method, i.e., clustering with particle filtering and discriminative segments (CPD) achieves the best performance.

Table 1 presents the comparison of our proposed method (TUC) with existing methods. Recall values for seven comparison methods are taken from [20]. Our method attains the highest average value for low and medium accuracy requirements, i.e., when overlap with the ground truth bounding box is greater than or equal to 0.25 and 0.50 threshold (τ) values, respectively. For high accuracy requirement, i.e., when τ is 0.75, our method achieves the second highest average value as CMT gets on the top. This slightly lower performance of our method in this case is attributable to the randomness involved in the method causing atremble movements of the bounding box sometimes. Note that this randomness, on the other hand, helps in keeping track of the object in other scenarios (low and medium accuracy requirements) where other methods show lower performance. After TUC and CMT, the next best results are achieved by STR and TLD.

Eminent performance of our method is clearly observable on sequences staging deformable objects (e.g., gym and person). Taking discriminative and using top 75 % parts of the object that match the reference model helps in achieving these high quality results, particularly for videos having deforming objects. We fixed the parameters for our method, e.g. variance (as described in Section 4.4), for all presented experiments. Adapting these parameters intelligently based on the object and its environment information in a sequence can further improve the overall results, and can make the method more generic in the future.

Figure 6 demonstrates the qualitative results of our method compared with the selected techniques. The figure gives the frame number and each frame shows the tracked object in the boundary from all 228 frames of the mountain-bike sequence.

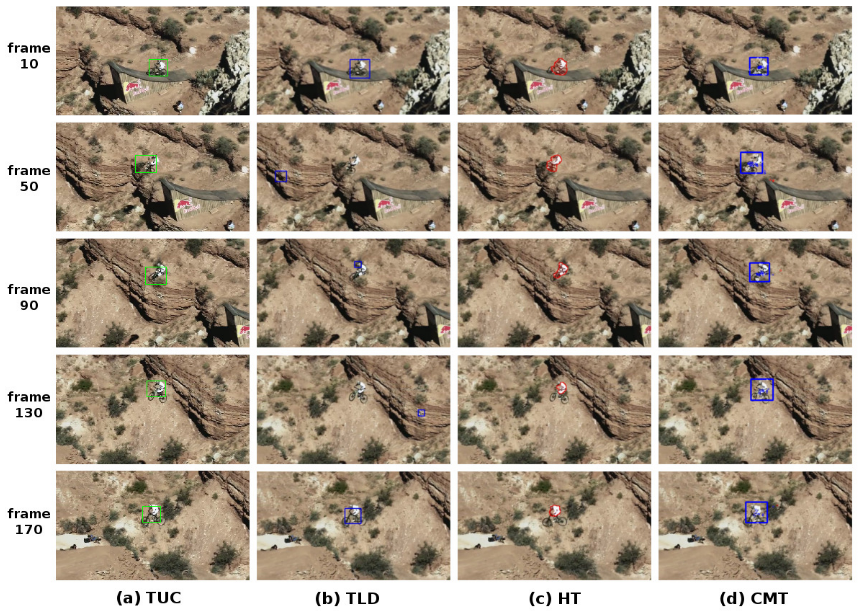


Figure 6. Qualitative results of our proposed tracking method compared with TLD, HT and CMT techniques (mountain-bike data set). Left column gives the frame number.

As discussed earlier, some of the values of control parameters are selected empirically, based on the best combination of correctness and efficient. Table 2 shows the recall values computed while trying different combinations of numbers of clusters and numbers of particles value. It is evident that after a certain number of clusters the segments become too sparse to track. Table 3 shows the recall values computed while trying different combinations of α and β value.

Currently, k-means clustering has been applied for object's segmentation. In the future, other clustering methods (e.g. density based) can be tested. In addition, more features and key-points detection and description methods can be explored to further improve the performance and to handle full occlusions more effectively. The method can also be extended to update the reference model at run-time and to generalize this technique to perform better in all cases. Super-pixel algorithm [1] (i.e. SLIC) can also be used for a fine and quick construction of the segmentation of the target object, as it is faster and more memory efficient. Moreover, some control parameters in this work are selected empirically, what we have considered

	Number of Clusters				
Number of Particles	5	10	15	20	25
100	0.52	0.7	0.92	0.85	0.85
150	0.65	0.79	0.81	0.59	0.97
200	0.6	0.85	0.98	0.82	0.87
250	0.58	0.83	0.94	0.84	0.86
300	0.56	0.68	0.89	0.72	0.83

Table 2. Recall values for combinations of number of clusters and number of particles experimented with the car2 video

	Alpha			
Beta	0.25	0.5	0.75	1
0.25	0.70	0.68	0.85	0.84
0.5	0.73	0.88	0.67	0.80
0.75	0.97	0.83	0.79	0.75
1	0.74	0.80	0.75	0.81

Table 3. Recall values for combinations of alpha and beta value experimented with the car2 video

as sufficient for the scope of this work. An adaptive parameter learning technique can be introduced for the further experimentation and extension of this work.

5 CONCLUSION

In this paper, we have proposed a single object tracking method, Tracking Using Clustering (TUC) by employing data clustering and particle filter. TUC outperforms state-of-the-art tracking methods in deformable object tracking while achieving competitive performance in general. Data clustering is applied to segment the target object into several unstructured parts. To reduce ambiguity, discriminative parts of the object are selected by removing its segments similar to the neighboring background segments. Particle filtering is employed to improve the accuracy and robustness of our method and overcome the lacking caused by the randomness inherited by data clustering methods. Experimental results on nine standard data sets demonstrate the effectiveness of our approach.

Acknowledgments

This work was in part supported by the Institute for Information and Communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (No. B0101-15-0525, Development of global multi-target tracking and event prediction techniques based on real-time large-scale video analysis), and by the National Strategic Project-Fine particle of the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT), and the Ministry of Environment (ME), and the Ministry of Health and Welfare (MOHW) (NRF-2017M3D8A1092022).

REFERENCES

[1] Achanta, R.—Shaji, A.—Smith, K.—Lucchi, A.—Fua, P.—Süsstrunk, S.: SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34, 2012, No. 11, pp. 2274–2282, doi: 10.1109/TPAMI.2012.120.

[2] Adam, A.—Rivlin, E.—Shimshoni, I.: Robust Fragments-Based Tracking Using the Integral Histogram. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '06), June 2006, Vol. 1, pp. 798–805, doi: 10.1109/CVPR.2006.256.

[3] Cai, Z.—Wen, L.—Lei, Z.—Vasconcelos, N.—Li, S. Z.: Robust Deformable and Occluded Object Tracking with Dynamic Graph. IEEE Transactions on Image Processing, Vol. 23, 2014, No. 12, pp. 5497–5509, doi: 10.1109/TIP.2014.2364919.

[4] Comaniciu, D.—Ramesh, V.—Meer, P.: Real-Time Tracking of Non-Rigid Objects Using Mean Shift. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2000), 2000, Vol. 2, pp. 142–149, doi: 10.1109/CVPR.2000.854761.

[5] Du, D.—Qi, H.—Li, W.—Wen, L.—Huang, Q.—Lyu, S.: Online Deformable Object Tracking Based on Structure-Aware Hyper-Graph. IEEE Transactions on Image Processing, Vol. 25, 2016, No. 8, pp. 3572–3584, doi: 10.1109/TIP.2016.2570556.

[6] Godec, M.—Roth, P. M.—Bischof, H.: Hough-Based Tracking of Non-Rigid Objects. Computer Vision and Image Understanding, Vol. 117, 2013, No. 10, pp. 1245–1256, doi: 10.1016/j.cviu.2012.11.005.

[7] Grabner, H.—Leistner, C.—Bischof, H.: Semi-Supervised On-Line Boosting for Robust Tracking. In: Forsyth, D., Torr, P., Zisserman, A. (Eds.): Computer Vision (ECCV 2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5302, 2008, pp. 234–247.

[8] Greminger, M. A.—Nelson, B. J.: Deformable Object Tracking Using the Boundary Element Method. Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003, Vol. 1, pp. I-289–I-294, doi: 10.1109/CVPR.2003.1211366.

[9] Greminger, M. A.—Nelson, B. J.: A Deformable Object Tracking Algorithm Based on the Boundary Element Method That Is Robust to Occlusions and Spurious Edges. International Journal of Computer Vision, Vol. 78, 2008, No. 1, pp. 29–45, doi: 10.1007/s11263-007-0076-6.

[10] Hare, S.—Saffari, A.—Torr, P. H. S.: Struck: Structured Output Tracking with Kernels. 2011 IEEE International Conference on Computer Vision (ICCV), 2011, pp. 263–270, doi: 10.1109/ICCV.2011.6126251.

[11] Hare, S.—Saffari, A.—Torr, P. H. S.: Efficient Online Structured Output Learning for Keypoint-Based Object Tracking. 2012 IEEE Conference on

Computer Vision and Pattern Recognition (CVPR), 2012, pp. 1894–1901, doi: 10.1109/CVPR.2012.6247889.

[12] Hilsmann, A.—Eisert, P.: Deformable Object Tracking Using Optical Flow Constraints. 4th European Conference on Visual Media Production (IETCVMP), 2007, pp. 1–8.

[13] Jackson, J. D.—Yezzi, A. J.—Soatto, S.: Tracking Deformable Moving Objects Under Severe Occlusions. 43rd IEEE Conference on Decision and Control (CDC), 2004, Vol. 3, pp. 2990–2995, doi: 10.1109/CDC.2004.1428922.

[14] Kalal, Z.—Mikolajczyk, K.—Matas, J.: Tracking-Learning-Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 34, 2012, No. 7, pp. 1409–1422.

[15] Kim, D. Y.—Jeon, M.: Data Fusion of Radar and Image Measurements for Multi-Object Tracking via Kalman Filtering. Information Sciences, Vol. 278, 2014, pp. 641–652, doi: 10.1016/j.ins.2014.03.080.

[16] Kim, D. Y.—Yang, E.—Jeon, M.—Shin, V.: Robust Auxiliary Particle Filter with an Adaptive Appearance Model for Visual Tracking. In: Kimmel, R., Klette, R., Sugimoto, A. (Eds.): Computer Vision (ACCV 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6494, 2010, pp. 718–731.

[17] Kristan, M.—Pflugfelder, R.—Leonardis, A.—Matas, J.—Čehovin, L.—Nebehay, G.—Vojíř, T.—Fernández, G.—Lukežič, A.—Dimitriev, A.—Petrosino, A.—Saffari, A.—Li, B.—Han, B.—Heng, C.—Garcia, C.—Pangeršič, D.—Häger, G.—Khan, F. S.—Oven, F.—Possegger, H.—Bischof, H.—Nam, H.—Zhu, J.—Li, J.—Choi, J. Y.—Choi, J.-W.—Henriques, J. F.—van de Weijer, J.—Batista, J.—Lebeda, K.—Öfjäll, K.—Yi, K. M.—Qin, L.—Wen, L.—Maresca, M. E.—Danelljan, M.—Felsberg, M.—Cheng, M.-M.—Torr, P.—Huang, Q.—Bowden, R.—Hare, S.—Lim, S. Y.—Hong, S.—Liao, S.—Hadfield, S.—Li, S. Z.—Duffner, S.—Golodetz, S.—Mauthner, T.—Vineet, V.—Lin, W.—Li, Y.—Qi, Y.—Lei, Z.—Niu, Z. H.: The Visual Object Tracking VOT2014 Challenge Results. In: Agapito, L., Bronstein, M. M., Rother, C. (Eds.): Computer Vision Workshops (ECCV 2014). Springer International Publishing, Lecture Notes in Computer Science, Vol. 8926, 2015, pp. 191–217.

[18] Li, Y.—Zhu, J.—Hoi, S. C. H.: Reliable Patch Trackers: Robust Visual Tracking by Exploiting Reliable Patches. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015, pp. 353–361, doi: 10.1109/CVPR.2015.7298632.

[19] Lucas, B. D.—Kanade, T.: An Iterative Image Registration Technique with an Application to Stereo Vision. Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81), Vol. 2, San Francisco, CA, USA, 1981, Morgan Kaufmann Publishers Inc., pp. 674–679.

[20] Nebehay, G.—Pflugfelder, R.: Consensus-Based Matching and Tracking of Keypoints for Object Tracking. 2014 IEEE Winter Conference on Applications of Computer Vision (WACV), 2014, pp. 862–869, doi: 10.1109/WACV.2014.6836013.

[21] Nebehay, G.—Pflugfelder, R.: Clustering of Static-Adaptive Correspondences for Deformable Object Tracking. Proceedings of the 2015 IEEE Conference on

Computer Vision and Pattern Recognition (CVPR), 2015, pp. 2784–2791, doi: 10.1109/CVPR.2015.7298895.

[22] Nguyen, H. T.—Smeulders, A. W. M.: Fast Occluded Object Tracking by a Robust Appearance Filter. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, 2004, No. 8, pp. 1099–1104.

[23] Oron, S.—Bar-Hillel, A.—Avidan, S.: Extended Lucas-Kanade Tracking. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (Eds.): Computer Vision (ECCV 2014). Springer International Publishing, Lecture Notes in Computer Science, Vol. 8693, 2014, pp. 142–156.

[24] Oron, S.—Bar-Hillel, A.—Levi, D.—Avidan, S.: Locally Orderless Tracking. International Journal of Computer Vision, Vol. 111, 2015, No. 2, pp. 213–228.

[25] Ristic, B.—Hernandez, M. L.: Tracking Systems. Radar Conference (RADAR '08), IEEE, 2008, pp. 1–2.

[26] Smeulders, A. W. M.—Chu, D. M.—Cucchiara, R.—Calderara, S.—Dehghan, A.—Shah, M.: Visual Tracking: An Experimental Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 36, 2014, No. 7, pp. 1442–1468.

[27] Staffa, M.—Rossi, S.—Giordano, M.—De Gregorio, M.—Siciliano, B.: Segmentation Performance in Tracking Deformable Objects via WNNS. 2015 IEEE International Conference on Robotics and Automation (ICRA), 2015, pp. 2462–2467, doi: 10.1109/ICRA.2015.7139528.

[28] Yilmaz, A.—Javed, O.—Shah, M.: Object Tracking: A Survey. ACM Computing Surveys (CSUR), Vol. 38, 2006, No. 4, Article No. 13, doi: 10.1145/1177352.1177355.

[29] Zhang, L.—van der Maaten, L.: Structure Preserving Object Tracking. 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013, pp. 1838–1845, doi: 10.1109/CVPR.2013.240.

[30] Zhang, L.—van der Maaten, L.: Preserving Structure in Model-Free Tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 36, 2014, No. 4, pp. 756–769.

[31] Zhang, M.—Kang, B.: An Improved Method of Tracking and Counting Moving Objects Using Graph Cuts. In: Wong, W. E. (Ed.): Proceedings of the 4th International Conference on Computer Engineering and Networks. Springer International Publishing, Lecture Notes in Electrical Engineering, Vol. 355, 2015, pp. 583–590.

[32] Zhang, X.—Hu, W.—Chen, S.—Maybank, S.: Graph-Embedding-Based Learning for Robust Object Tracking. IEEE Transactions on Industrial Electronics, Vol. 61, 2014, No. 2, pp. 1072–1084.

Muhammad Aasim Rafique received his M.Sc. degree in computer science from Quaid-e-Azam University, Islamabad, Pakistan. He then received his M.Sc. degree in computer science from Lahore University of Management and Sciences, Lahore, Pakistan in 2008. He received his Ph.D. degree from School of Electrical Engineering and Computer Sciences, GIST, Gwangju, Republic of Korea, in 2018 (when submitted this article he was Ph.D. student at GIST). He is now working as Assistant Professor at Quaid-e-Azam University, Islamabad, Pakistan. His research interests are artificial neural networks, their application in machine learning and computer vision.

Moongu Jeon received his B.Sc. degree in architectural engineering from the Korea University, Seoul, Korea, in 1988 and his M.Sc. and Ph.D. degrees in computer science and scientific computation from the University of Minnesota, Minneapolis, MN, USA, in 1999 and 2001, respectively. As Postgraduate Researcher, he worked on optimal control problems at the University of California at Santa Barbara, Santa Barbara, CA, USA, in 2001–2003, and then moved to the National Research Council of Canada, where he worked on the sparse representation of high-dimensional data and the level set methods for image processing until July 2005. In 2005, he joined the Gwangju Institute of Science and Technology, Gwangju, Korea, where he is currently Full Professor at the School of Electrical Engineering and Computer Science. His current research interests are in machine learning, computer vision, and intelligent transportation systems.

Malik Tahir Hassan received his M.Sc. and Ph.D. degrees in computer science from Lahore University of Management Sciences (LUMS), Lahore, Pakistan. He worked at Gwangju Institute of Science and Tehcnology (GIST), Gwangju, South Korea, as a Post-Doc fellow. Currently, he is working as Assistant Professor at School of Systems and Technology (SST) at University of Management and Technology (UMT), Lahore, Pakistan. His research interests include pattern recognition, text mining, recommender systems and autonomic computing.

Computing and Informatics, Vol. 37, 2018, 737–758, doi: 10.4149/cai_2018_3_737

SENTIMENT AND AUTHORITY ANALYSIS IN CONVERSATIONAL CONTENT

Kristína Machová, Martin Mikula
Martina Szabóová, Marián Mach

Department of Cybernetics and Artificial Intelligence
FEI, Technical University of Košice
Letná 9, 042 00 Košice, Slovakia
e-mail: {kristina.machova, martin.mikula, martina.tarhanicova, marian.mach}@tuke.sk

Abstract. This paper deals with mining conversational content from the social media. It focused on two issues: opinion and emotion classification and identification of authoritative reviewers. The paper also describes applications representing the results obtained in the given areas. Authority identification can be used by organizations to search for experts in their specific areas to employ them. The opinion and emotion analysis can be useful for providing decision-making support.

Keywords: Opinion analysis, emotion analysis, conversational content, conversation structure, authority identification

Mathematics Subject Classification 2010: 68U15

1 INTRODUCTION

The social web is a phenomenon of the present day where users create a huge amount of information called conversational content. It increases the number of interactions among web users affecting our decisions in real life situations. Therefore, application of both sentiment analysis and authority mining can be useful in the process of decision making. The input of this application can be a hypertext reference to the related web discussion or a key phrase from the area of interest being discussed. The output is a summarized opinion [2] (clearly positive, neutral or negative) represented in this discussion as well as information about authoritative discussants.

Authoritative users can influence us more than the others, therefore, having knowledge about them can improve the precision of sentiment analysis. In this paper, an authoritative user or authoritative reviewer will be referred to as "authority". The problem of *"authority identification"* can be solved through finding a function for estimation of the value of authority, which represents the measure of authoritativeness of a given discussant, and defining a threshold on the value of the variable authority. All discussants with the value of authority equal to or higher than the defined threshold will be *"identified as authorities"* of a given web discussion.

In this paper, we focus on two main issues. The first is sentiment analysis. The concept of the sentiment is represented by the opinion or some kind of emotion. Based on the expression of the sentiment we divided our work into opinion classification and emotion classification. We used two different dictionaries: one to classify opinions into positive and negative classes and the other to classify emotions into six basic emotions.

The second issue is based on authority identification. We define authority as a person, who understands the topic, writes comprehensive answers, has many good reactions to his/her comments and has a good reputation in the community. We analyse these features as well as the state threshold of the authority value and label all authors having a higher value than the authorities in the discussion.

Thus, the main contributions presented in this paper are new approaches to classifying opinions and emotions, and a completely original approach to estimation and identification of the authority in web discussions.

2 STATE OF THE ART

The tasks of both sentiment analysis and opinions and emotions analysis can be performed using two main approaches [22]:

- *Lexicon based approach*, which calculates the polarity of a comment from polarities of words or phrases in the text of the comment. According to [16] the lexicon-based approach uses the lexicons which can be created manually, semi-automatically or automatically by using another dictionary or corpus.
- *Classification based approach*, which builds classifiers from labelled examples of comments. This approach can use statistical or machine learning methods.

The *lexicon-based approach* takes into consideration only the words which can express the sentiment the best way and are stored in a classification dictionary. Based on the polarity of each word in the dictionary, the polarity of the whole comment and subsequently of the whole web discussion can be determined. The polarity of a word may also depend on the context. Consideration of the context requires using more complicated techniques. The simplest dictionaries enable to perform binary classification into positive or negative sentiment. More complex dictionaries can also determine the strength of this polarity. Such dictionaries can be created for a particular domain or application. Some well-known lexicons are, for example, WordNet[1], WordNet-Affect [26], SenticNet [6], SentiWordNet [1, 9], etc. An approach most similar to our solution of the sentiment analysis problem is presented in [27]. It uses a dictionary of words annotated with their polarity. This work splits this dictionary into four sub-dictionaries according to word classes (adjectives, nouns, verbs and adverbs), these dictionaries are checked for consistency and reliability. In this approach, more words in the dictionary can result in an increase of noise levels and subsequently decrease in the precision.

In the *classification based approach*, many of the well-known machine learning methods can be used. The machine learning methods estimate user's opinions or emotions on the basis of a set of training examples. These training examples are represented by annotations of comments to a web forum. The most common machine learning methods are, for example, Naïve Bayes classifier or Support Vector Machines (SVM), as well as statistical methods such as Maximal Entropy. All these algorithms were used in the work [10]. The author focused on the sentiment classification from the micro-blog service Twitter. When machine learning is used for sentiment analysis it is important to select the best features. Different kinds of features, such as pointwise mutual information, information gain, chi-quadrat and term frequency were tested in works [28, 32]. Another approach presented in [29] is focused on the SentiStrength detection algorithm, which solves some problems connected with sentiment analysis, for example: generation of the sentiment strength list, optimization of the sentiment word strengths, allocation of missing words, spelling correction, creation of the booster word list, negating word list and emoticon list, processing of the repeated letters and ignoring negative emotion in questions. Their approach is based on using machine learning techniques (Logistic Regression, Support Vector Machine, J48 Classification Tree, AdaBoost, Decision Table, Multilayer Perceptron and Naïve Bayes).

An important drawback of the machine learning approach is its dependency on a huge amount of annotated training examples. An annotation tool with some level of automation had to be utilized. The source [25] describes some annotation tools, for example, GATE (General Architecture for Text Engineering), SemTag (Semantic Annotation Tool), the annotation platform KIM (Knowledge and Information Management) and Luvak, or a more general semi-automatic annotation tool built on the Eclipse platform.

The second problem, which we tried to solve in relation to sentiment analysis, was the authority identification in some web discussions. The most known approaches determine the authority degree only from the conversation structure [4, 7, 31]. Our approach is not based on NLP (Natural Language Processing) or on information retrieval algorithm but on generating the estimation function for labeling the degree of authority prediction (see Section 4).

[1] `http://wordnet.princeton.edu`

3 SENTIMENT ANALYSIS

3.1 Opinion Classification

Our opinion analysis approach has focused on the classification of web users' opinion. It summarizes positive, neutral or negative polarity across the discussion. This classification is provided in several steps. At first, the polarity of particular words is identified. Next, the polarity of particular lexical units is distinguished and consequently the polarity of the whole comment is stated. The final step is classification of the whole discussion to positive, neutral or negative polarity. Other web discussions concerning the discussed topic can be also processed. It means that the resulting opinion can be compiled from more web sources.

The basic problems can be simply solved using classification dictionaries. These dictionaries focus on words, which express sentiment very well – mainly adjectives (e.g. "extraordinary") [3] and adverbs (e.g. "awfully") [17]. On the other hand, some other words must be also taken into account in order to achieve the satisfactory precision, for example nouns (e.g. "crash") or verbs (e.g. "damage") [27]. All of these words are identified in the text; they usually enter the dictionaries with their degree of polarity.

For the purpose of deeper processing of a text, *shifters* can be used, which are words that can change the polarity of a word. The positive influence of the shifters processing was studied in work [14]. Shifters can be divided into two groups:

- negation,
- intensification.

There are two basic approaches to *negation processing*: a switch negation and a shift negation. The *switch negation* is simply reversion of the polarity of a lexical unit. In this case, the reversion is changing the sign of a number, which represents the polarity degree (from minus to plus and vice versa). There are many words related to negation such as *not*, *any*, *never*, *nothing*. They are usually located next to the related word. Other negations as *without, don't, lack* etc., which can be situated at a significant distance from the lexical item should also be considered. These negations can be hardly processed by the switch negation. In some cases, the switch negation may not be sufficiently precise because the negation of a strong positive word is rarely a strong negative word and vice versa. More often the negation of a strong positive word is a slightly negative word. The *shift negation*, instead of changing the sign, shifts the polarity degree towards the opposite polarity by a fixed value (e.g. value 4 in the implementation [27]). For example: "She's not terrific ($5 - 4 = 1$) but not terrible ($-5 + 4 = -1$) either."

The *intensification processing* assumes the existence of a dictionary of intensifiers. An intensifier is a word, which can increase (or decrease) the intensity of polarity. According to [27], intensifiers can be of two categories: *amplifiers* (e.g. *very*) increase the semantic intensity of a neighbouring lexical unit, whereas *downtowners* (e.g. *slightly*) decrease it. All intensifiers are stored in the dictionary together with a sign and a value. The value represents the percentage of the change in the polarity intensity and the sign represents the type of this change ("plus" represents the increase in the polarity value by an amplifier and "minus" represents the decrease in the polarity value by a downtowner).

3.1.1 N-Grams Approach to the Opinion Classification

An n-gram can be defined as a series of items from a sequence. From the semantic point of view, it can be a sequence of characters or words. In practice, n-gram as a sequence of words is the most common. Our approach uses n-grams for splitting web discussion comments into lexical units and that is a dictionary-based approach. This application works with Slovak texts. The dictionary consists of two parts. The first part contains adjectives, nouns and verbs. The second part contains adverbs and negations. The first part of the dictionary is used to solve the basic problems of the opinion classification. The second part of the dictionary is used in negations and intensification processing, because only adverbs can increase ("*surprisingly nice*") or decrease ("*extremely low-class*") the intensity of a related word polarity. The first (basic) dictionary also contains some emoticons, which naturally can express emotions and opinions very well. In the case, when the analysed text is less clear, the emoticons can increase the precision of the classification. All words and emoticons from the first dictionary are quantified to the polarity degree from the interval $\langle -3, 3\rangle$ (Table 1). Intensifiers (adverbs) in the second part of the dictionary are assigned values from the interval $\langle -0.5, 1\rangle$ and negations are represented by the value -2 (Table 2).

Polarity Degree	Words and Emoticons
3	:D, godlike, extraordinary
2	:), super, excellent
1	nice, functional, OK
−1	unpleasant, weak
−2	:(, shocking, miserable
−3	:((, fatal, catastrophic

Table 1. Polarity degrees of example words and emoticons (the first part of the dictionary)

Polarity Degree	Words
1	very, totally, extraordinarily
0.5	suitably, really, actually
−0.5	little, overly, unnecessarily
−2	Negations: no, not, don't

Table 2. Polarity degrees of negations and intensifications (second part of the dictionary)

All the words from the analysed comments are compared with all words stored in the first and in the second part of the dictionary. In case that words match with the first dictionary, the values of all matching words are summed up. The resulting sum represents the solution of the basic problems of opinion classification (the first sum in the formula (1)). The second sum takes into account negations and intensifications of the related words incorporated in the first sum. This is the reason why multiplication (not an aggregation) was proposed to be used between the first and the second sum in the formula (1). The second sum aggregates the values obtained from the second dictionary for intensifiers and negations. But if comments do not contain any negation or any intensification, the second sum will be zero, and consequently the resulting polarity of the comment will be zero. Thus, value "1" was added to the second sum as a neutral value. This idea is represented by the formula (1):

$$P = \sum v(w_i^1) \left[1 + \sum v(w_J^2) \right] \tag{1}$$

where:

- P is the polarity degree of analysed text,
- $v(w_i^1)$ is the value of word w_i from the text found in the first dictionary,
- $v(w_i^2)$ is the value of word w_J from the text found in the second dictionary,
- $\sum v(w_i^1)$ is the first sum from the first dictionary (solution of the basic problem),
- $\sum v(w_J^2)$ is the second sum from the second dictionary (solution of the negation and intensification).

To illustrate the topic, we give a few examples:

- A sentence containing only positive and negative words without negations and intensifications can be processed using only the first dictionary
  - "*The mouse is nice but the processing is miserable and globally it is unsuccessful.*" is processed in the following way. Only three words from the first dictionary were found in the sentence with their degree of polarity in the parentheses:

    nice (+1) + *miserable* (−3) + *unsuccessful* (−1).

    There are polarity degrees were the sum and the final polarity of the sentence equals $P = -3$.

  A sentence containing negation
  - "*It is not a good solution.*" is processed in the following way. Only the word *good* (+1) was found in the first dictionary, so the result of the first sum is "1". The sentence contains also negation *not* (−2) from the second

dictionary, so the result of the second sum is "−2". According to formula (1) the final polarity is: $P = 1 * [1 + (-2)] = -1$.

A sentence containing intensification

– "*Globally, the processing is very polite.*" is processed in the following way. Only the word *polite* (+1) was found in the first dictionary, so the result of the first sum is "1". The sentence contains also intensification *very* (+1) from the second dictionary, so the result of the second sum is "1". According to formula (1) the final polarity is: $P = 1 * [1 + 1)] = +2$.

More information on this approach can be found in [20]. In our approach, the dictionary can be created directly from web discussions. It increases the precision of the opinion classification in the given domain. Our approach also uses a different method for the processing of negations and intensifications based on the formula (1). Our approach does not need an intensifier (negation) to be located in the neighbourhood of the related word. They can take any position around the lexical unit. This length is limited by the value "4" in 4-gram approach. There are two possibilities for the location of the intensifier and negation. They can be located before and/or after the related word.

The value $n = 4$ in the n-gram was determined experimentally. Experiments with the value n from 2 to 4 were performed to find the best value of n. The ideal value had to be sufficiently big to avoid isolation of the processed word but not too big to cover the whole sentence. The experiments showed that $n = 4$ was sufficient. Thus, the greater value, for example $n = 5$, would only make the processing more complex. When we compared 4-grams with other n-grams, 2-grams and 3-grams did not bring any benefit, because 4-grams could also cover phrases of two or three words. The work [15] also uses 4-grams in the sentiment analysis engine called Umigon for the sentiment analysis of tweets. The texts of tweets are decomposed into a list of 4-grams, and they are compared with terms in lexicons. Each term searched in the lexicon is processed using heuristics and decision rules for 4-gram polarity determination.

The n-grams approach applied to the opinion classification was tested on a set of discussion comments from the portal `http://www.mojandroid.sk` (discussion thread related to reviews of the mobile telephones HTC One X and HCT One S) and `http://www.pocitace.sme.sk` (discussion thread related to reviews of two products Asus Transformer Prime TF201 and Asus Transformer Pad TF300T). 200 comments including 100 positive and 100 negative were used in the experiments. Thus, objects were equally divided into the two classes. The evaluation of n-gram implementation was based on the pair of the well known metrics – precision and recall. These measures depended on the numbers of true positive, false positive and false negative classifications of our implementation in comparison with the opinion of a human expert on real positivity or negativity of the evaluated cases. The resulting precision and recall of our n-gram implementation is given in Table 4. The achieved precision and recall, mainly recall of negative comments, was low. These results were influenced by comments containing irony or the polarity hidden in the context. On the other hand, this implementation had quite a good precision in the processing of positive comments.

3.1.2 Opinion Classification Based on a Special Lexicon

Because of the unsatisfactory results of the implementation based on n-grams, we improved our dictionary-based approach. Our aim was to increase the efficiency of this approach and to extend it by a special lexicon[2] for the Slovak language. The lexicon with Slovak words was created by the translation of its available version in English used in work [13]. We added synonyms and negations of words from the Slovak thesaurus into this lexicon. The new lexicon contains 1 430 words (598 positive words, 772 negative words, 41 intensifiers and 19 negations). As in the previous case, our lexicon contains the following word classes: adjectives, adverbs, nouns and verbs. Each word has two attributes. The first attribute is a degree of polarity within the range from -3 to 3, where -3 is the most negative polarity and 3 is the most positive polarity. The second attribute denotes the type of a given word from four possibilities:

- p – positive word,
- n – negative word,
- i – intensifier,
- o – opposite/negation.

Negation words are assigned the value -1. Intensifiers have values of polarity degree in the range $\langle 1.0, 2.0 \rangle$. Examples of words and their values of the polarity degree and the type of word are illustrated in Table 3.

Polarity Degree	Type of Word	Word
3	p	extra, genius, super, brilliant
2	p	better, advanced, success
1	p	good, ok, strong, smart
-1	n	bad, weak, boring
-2	n	dangerous, hostile, stupid
-3	n	terrible, waste, worst
-1	o	no, not, never, haven't
1.25	i	really, rather
1.5	i	middle, pretty
1.75	i	too, complete
2	i	very, total, absolute

Table 3. Examples of words in the dictionary with the polarity degree and the type of word

[2] `http://klanaz.studenthosting.sk/sa.html`

The analysed comment text was split into individual sentences and diacritic marks were removed from the text. All words of the text were converted into nominative of the plural using the modified version of the Lancaster stemming algorithm[3]. The converted words were compared with the words in the dictionary. The sentence was processed word by word. Each positive or negative word was multiplied by intensifications and negations. Then, all polarities of positive and negative words were summed up according to the formula (2). In the case, when a comment did not contain any intensification or negation, the values of intensification and negation were set to 1.

$$P = \sum \left[w_w \prod w_i \prod w_n \right] \tag{2}$$

where:

- P – polarity degree of the analysed sentence,
- w_w – value of positive or negative word,
- w_i – value of intensifier,
- w_n – value of negation,
- $\prod w_i$ – multiplication of all intensifiers,
- $\prod w_n$ – multiplication of all negations.

The resulting polarity of the sentence was adjusted using logarithmic function in the formula (3) to avoid processing huge numbers – values of P. When the value of the sentence polarity before adjusting was 0, the adaptation was not used. When the value of the sentence polarity was lower than 0, the absolute value was adjusted and multiplied by -1 to maintain the negative polarity of the comment.

A dataset did not contain comments with just facts. Such comments were not collected and added to the dataset during its creation. Sentences which contained no sentiment words were evaluated as mistakes. For example, the sentence which was labelled as positive but contained no sentiment word was evaluated in the same way as a negative one. The resulting sentiment value was 0 in these cases.

$$P_l = 1 + \log_{10} |P| \tag{3}$$

where

- P_l – the new value of sentence polarity,
- P – the polarity value of sentence obtained by the formula (2).

The following examples illustrate the above computation of the polarity degree value. A sentence containing only positive and negative words without intensifications and negations

[3] `https://goo.gl/STmHiO`

- *"The mouse is nice but the processing is miserable and globally it is unsuccessful."* is processed in following way. Three words from the dictionary were found in the sentence with their degree of polarity in parentheses:

  $$\textit{nice}\ (+1) + \textit{miserable}\ (-2) + \textit{unsuccessful}\ (-1).$$

  These degrees of polarity were the sum and the final polarity of the sentence is $P = -2$. After adjustment according to the formula (3) $P_l = -1.3$.

A sentence with intensification (very) and negation

- *"It is not a very good solution."* is processed in following way. Three words from the dictionary were found in the sentence with their degree of polarity. The first word was *good* (+1), so $w_w = 1$. The sentence also contains intensification *very* (2), so $w_i = 2$ and negation *not* (−1), so $w_n = -1$. According to the formula (2) the final polarity is $P = -1 * 2 * 1 = -2$. After adjustment according to the formula (3) $P_l = -1.3$.

This approach is a little different in comparison with the previous n-gram based application. It uses only one dictionary for all types of words and this dictionary contains mainly adjectives and nouns in nominative plural (the dictionary also contains verbs in the form as they appear in the original text without any modification). In comparison with the n-grams implementation, this approach uses a different method to process intensification and negation. Moreover, this approach can process multiple intensifications (e.g. *very very good*).

Our approach was tested on two datasets. The first dataset was the same as the dataset for n-grams. It allowed to compare our method with the previous one based on n-grams. A new dataset was created to test our approach. This second dataset contained collected comments from different areas (e.g. films, electronics, politics, etc.) Each comment was labelled by a human annotator. The neutral ones and facts were removed from this dataset. The second experiment was performed on the set of 5 242 comments and 182 645 words, where 2 573 comments were positive and 2 669 negative so that the objects were equally divided into available classes. The dataset is available on the website[4].

The results of testing given in Table 4 show that quite a good recall was achieved for positive comments. We compared our approach with the n-grams approach in the first test. The precision obtained for positive comments was lower, but the results for other indicators were better. The reason could be the dictionary, which contains more words and more comprehensive computation of the polarity degree. This approach is more similar to human understanding of the text. In the second experiment, a good recall was achieved for positive comments, but it was low for negative comments. The results could be influenced by irony, sarcasm or description of opinions without polarity words. The number of missclassified comments was also increased by comments, which did not contain polarity words. These comments

[4] `http://klanaz.studenthosting.sk/dataset.txt`

were evaluated as mistakes and added to the same class as incorrectly evaluated comments.

Experiment	Precision (pos)	Precision (neg)	Overall Precision	Recall (pos)	Recall (neg)	Overall Recall
NGR 1	0.830	0.570	0.700	0.652	0.214	0.433
SD 1	0.654	0.727	0.691	0.850	0.471	0.661
SD 2	0.561	0.675	0.618	0.802	0.396	0.599

Table 4. The precision and recall of tests achieved by the n-grams approach (NGR) and the approach based on the special dictionary (SD)

The results of our experiments with opinion classification showed that using our first approach (Section 3.1.1) the best value of precision obtained was 0.830 and the best value of recall 0.652. Our experiments with modified approach (Section 3.1.2) achieved better results in recall (0.850) than in precision (0.727). In comparison with another method of opinion classification using an approach that is very close to the processing negation and intensifiers [27], our results were slightly worse. The results of F1 (combining precision and recall using equal weights) presented in the study by Taboada, were in the range from 0.58 to 0.89 according to types of reviews.

3.2 Emotion Classification

Emotion analysis is a similar problem as the opinion classification. Both of them (opinion and emotion classification) represent the sentiment analysis. Accordingly, in emotion classification words from dictionaries representing emotions (joy, anger, disappointment, ...) are searched for in the input text. The type of emotion expressed by these words denotes the kind of emotion presented in the given text.

According to [11], three major directions in emotion computing can be recognized: categorical/discrete, dimensional and appraisal based approaches. Despite the existence of other models, the categorical and dimensional approaches are the most commonly used models for automatic analysis and prediction of the emotion in the continuous input.

The Categorical Approach claims there is a small number of basic emotions that are hard-wired in our brain, and recognized across the world. Emotional states are classified by a single category. However, a couple of researchers proved that people show non-basic, subtle and rather complex emotional states that could be impossible to handle, such as embarrassment or depression [12].

The Dimensional Approach is based on Wundt's [30] proposal that feelings (which he distinguishes from emotions) can be described as pleasantness – unpleasantness, excitement – inhibition and tension – relaxation, as well as Osgood's work [21] on the dimensions of affective meaning (arousal, valence, and potency). Most recent models concentrate on only two dimensions, valence and arousal. Valence (pleasure/displeasure) depicts how positive or negative emotions can be. Arousal (activation/deactivation) depicts how exciting or apathetic emotions can be.

For our research purposes we decided to choose the categorical approach, i.e. Ekman's [8] six basic emotions: happiness (positive), sadness (negative), surprise (positive/negative), fear (negative), disgust (negative), and anger (negative).

There are four major approaches to emotion classification in the text: dictionary-based methods, machine learning methods, knowledge-based methods and hybrid methods. Our main interest is the *dictionary-based methods*. As our target language is Slovak, and we are not aware of any Slovak lexicon for emotional words, we created one. Every word (see Table 5) in the lexicon is labelled by an appropriate emotion (happiness, sadness, surprise, fear, disgust, and anger), part of speech (noun, adjective, adverb, verb) and intensity (in the range $\langle -3; 3\rangle$, -3 being the most negative and 3 the most positive). Emotions can be typically positive or negative. For example, sadness is a negative emotion, but a surprise can be both positive and negative. In such a case, the resulting polarity depends on the context in the form of the surrounding words. The dictionary used in our study was created using a web based application and contains about 19 000 words with information about polarity and emotions (see Table 5). We consider the wisdom of the crowd being the most straightforward way of obtaining data from users.

Word	Part of Speech	Emotion	Polarity
horlivo (eagerly)	adverb	joy	3
nenávidím (hate)	verb	angry	−3
horšia (worse)	adjective	sadness	−3
bezstarostnosť (carelessness)	noun	joy	0

Table 5. Examples of Slovak words in the dictionary with the part of speech tag, type of emotion and polarity degree

The experiment was performed on the same dataset, which was used in the second experiment SD2 within "Opinion Classification Based on a Special Lexicon". This dataset contained 5 242 comments and 182 645 words, where 2 573 comments were positive and 2 669 negative. Thus, in this experiment, objects were also equally divided into classes. The results of tests are given in Table 6. The highest recall was achieved for the emotion "happiness" and the lowest for the emotion "surprise". It could be caused by the fact that the emotion "surprise" is hard to label (for every other emotion it is easy to determine either positivity or negativity but surprise can represent both classes). The labelling of emotions was based on computing the probability of each emotion for the given text. For the rest of the emotions, we needed to improve our dictionary by adding new words to it because the values of precision and recall were low.

The results of our experiments also showed that we might need to reconsider using Plutchik's wheel of emotions illustrated in Figure 1 which adds two basic emotions (anticipation and trust) to Ekman's six emotions. It could cover a wider range of words and also increase a recall for each emotion.

In the tests presented in Table 6, the evaluation through precision and recall was based on comparison of our implementation of "opinion" on the resulting emotion

Emotion	Precision	Recall
Happiness	0.651	0.701
Sadness	0.589	0.590
Surprise	0.423	0.382
Fear	0.566	0.506
Disgust	0.473	0.424
Anger	0.622	0.651

Table 6. Achieved precision and recall of tests of the designed approach

with an "opinion" of human experts on the emotion presented in the text. These comparisons were made by a contingency table, from which all values of precision and recall were calculated. In this contingency table, all emotions were represented by six classes. Of course, there can be more than one emotion in one review. If that is the case, the resulting emotion is the emotion, which is the most probable, because

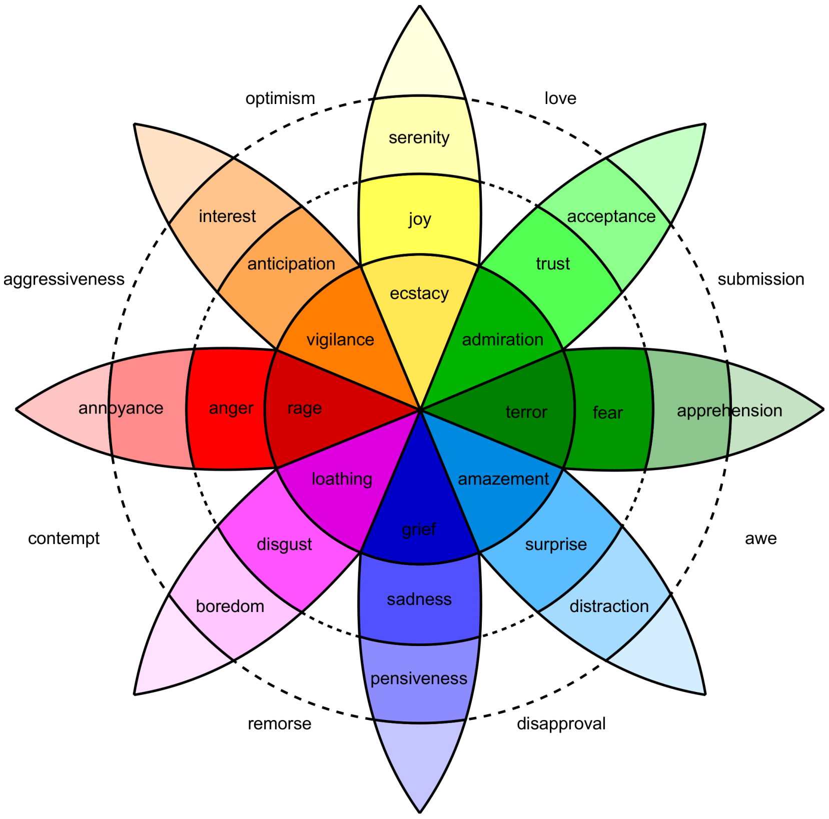


Figure 1. Plutchik's wheel of emotion [23]

the given review contains most words labelled with the considered emotion. When more than one emotion achieves the same number of votes, then all these emotions are taken into consideration. The result is that the given input text contains more emotions and this fact has to be a part of using the emotion classification for practical purposes. For example, a robot should adapt its behaviour to all these emotions. A problem can arise only when these emotions are in a contradiction. In this case, the result is not usable (e.g. in human–robot interactions).

4 AUTHORITY IDENTIFICATION

This section is focused on the identification of the authority of persons, who comment a topic on a web discussion forum. Authority identification represents a different problem than authorship identification. It does not answer the question "Who is the author?". Authority identification answers the question "Does the author have a deep knowledge of the topic?".

Authority can be formal (stated by a measure of power) and informal (stated by a measure of favour). The formal authority is given by a position or function in the organization. The informal authority is given by his/her credibility, wisdom, orientation, ability of good decision making, etc. This authority is enforced by other people's respect. Our approach has focused on the informal authority identification in web discussion forums. During the process of an online discussion, the structure of a conversation is created. This structure can be represented by an acyclic graph – the conversation tree.

People have various reasons for contributing to a discussion forum. Many of contributors are people, who want to find answers to their questions. These contributors create a core of the discussion, but they are not very authoritative. A smaller group of contributors is the group of troublemaking actors. They are provocateurs seeking an opportunity to present their opinions and invoking conflicts. They are not authoritative as well. They should be eliminated from the discussions. The last group of contributors are actors, who express their knowledge and share their ideas or opinions. These actors enter into the discussion seriously, add only truthful information, and join only if they are familiar with the topic. They are really authoritative contributors. We are interested in distinguishing them from the other actors.

The search for authoritative actors involves mining from web discussions. The input data contain the following aspects of comments:

- contributor name,
- reacting comment,
- length of the comment,
- position of the comment in the discussion represented graphically by the conversation tree.

The problem of authority identification is based on the estimation of authoritativeness *(A)*. The value of A is related to the contributors. Thus, data about each of 117 contributors in our dataset were collected including the following information:

- *NC* is the number of comments of a given contributor. We proposed that someone who understands the topic (authority) would contribute to the discussion more often than other actors.
- *ANR* is the average number of reactions to the comment(s) of a given author. This argument represents the number of reactions that support or negate a statement of the author, whose authority is examined. We started from the assumption that a more authoritative contributor could evoke a higher number of reactions.
- *AL* is the average number of all layers, at which the comments of a discussant are situated in the conversation tree. The conversation tree is a graphical representation of a web discussion. The *AL* represents the information, when the discussant joins the discussion, at the beginning or at the end. For example, a contributor, whose comments are located at the bottom level of the conversation tree, usually adds comprehensive comments answering all the questions. This can be the authoritative type of contributors.
- *NCH* is the number of characters, which represents the average length of comments of an author. This number is a common ratio of the number of all characters of the given contributor to the number of all his/her comments in the discussion. It penalizes authors with too short and thus less informative comments. We assume that an authoritative contributor does not post extremely short comments.
- *K* is karma of a contributor in the form of a number from 0 to 200, which represents the discussant's activity in the last 3 months from "`www.sme.sk`".
- *AE* is the average evaluation of a comment in the form of the ratio of the sum of all reactions (agree (+) and disagree (−)) to this comment of a given discussant to the number of all his/her comments. This average evaluation is available on the web discussion page. The AE range is a number from 0 to 80.

For the informal authority identification (detection), the main task is to estimate the function of authoritativeness A. In general, it is the function (4):

$$A = f(NC, ANR, AL, NCH, K, AE). \tag{4}$$

Firstly, we used a linear function with weights determined experimentally followed by regression analysis to compute weights of linear, polynomial and nonlinear functions. To compute these weights, it was necessary to know the values of the independent variables *NC*, *ANR*, *AL*, *NCH*, *K*, *AE*, as well as the values of the dependent variable *A*. The values of the variable *A* were derived from:

- evaluation of each discussant by "human expert",

- evaluation of each discussant by other discussants – it represents "wisdom of the crowd".

The following regression functions for authority estimation were generated in the process of learning:

- linear function learned from the "human expert",
- linear function learned from the "wisdom of the crowd",
- polynomial function learned from the "human expert",
- polynomial function learned from the "wisdom of the crowd",
- non-linear function learned from the "human expert",
- non-linear function learned from the "wisdom of the crowd".

All these 6 functions were tested and the validation was performed using the following measures:

- an average deviation – used for the validation of estimation functions,
- a precision – used for the validation of classification,
- a recall – used for the validation of classification.

We classified authors into two classes: Authority and Non-authority. A contributor with the estimated value of authoritativeness A greater than 70 was labelled as Authority otherwise he/she was labelled as Non-authority. The value of authoritativeness A could be in the interval (0, 100). The test dataset contained the data on 117 contributors. According to the results presented in Table 7, the best results were obtained by learning of linear function from the "wisdom of the crowd" in formula (5) and by learning of non-linear function from the "wisdom of the crowd" in formula (6). Surprisingly, learning of polynomial function from the "wisdom of the crowd" also provided good results but only in "recall".

$$\begin{aligned} A = {} & 0.4385AE + 0.325K + 0.002NCH - 0.2928AL - 0.0853ANR \\ & + 1.0728NC, \end{aligned} \tag{5}$$

$$\begin{aligned} A = {} & 0.0185AE^{1.8135} + 141.5704K^{-78.39} + 0.0018NCH^{1.0457} - 0.0011AL^{3.7717} \\ & - 0.5562ANR^{0.0001} + 37.6642NC^{0.0038}. \end{aligned} \tag{6}$$

Authority identification can be used in a variety of real situations. For example, an inexperienced web user searches for an authority that is able to provide him/her with advice and decision-making support. Another example – a technically oriented organization requires skilled employees, specialists who are authorities in the given field, and to whom such organization can offer interesting job positions. Thus, the person responsible for recruiting can search for authoritative users on web forums focused on the technologies used in this organization to fill in specific job positions.

	Deviation		Precision		Recall	
Version	Expert	Crowd	Expert	Crowd	Expert	Crowd
Linear	17.34	**3.29**	0.70	**0.98**	0.67	**0.80**
Polynomial	24.01	8.79	0.67	0.78	0.61	0.94
Non-linear	18.11	**6.56**	0.67	**0.97**	0.67	**0.80**

Table 7. Achieved average deviation, precision and recall of tests of the designed approach to the authoritativeness identification

5 CONCLUSIONS

The paper introduces a variety of approaches to social conversation data mining. The main attention is focused on two problems: sentiment analysis (opinion and emotion classification) and authority identification. It describes two approaches to opinion classification and one approach to authority identification.

Using the opinion classification approach, the comments were classified into two classes: positive and negative. The first opinion classification method used 4-grams to assign polarity to comments. It was able to process intensification and negation within the range of 4 words. This approach achieved good results for positive comments. The classification of negative comments was worse. That was the reason to develop the second opinion classification approach and create a new lexicon for this new approach. We also used a different method to process intensification and negation. This second method achieved better results than the one used previously.

The paper also describes the approach to emotion classification. Such approach mostly used to identify emotions in a text is similar to the approach used for identifying the polarity of the text. We focused on the lexicon-based approach in both cases – therefore, we created a lexicon that gave us information about emotions. By applying this lexicon we obtained interesting results. However, the results also showed that we might need to reconsider using Plutchik's wheel of emotions which adds two basic emotions (anticipation and trust) to Ekman's six emotions for the approach to be more precise in emotion classification. Changing the models also required reworking of the lexicon. In addition we had to take into consideration that emotions have no strict boundaries which means they often overlap each other, so it was a challenge to differentiate them properly.

Implementation of a new approach to authority identification in web discussions is presented and the resulting rating of authoritative contributors is provided. It should be noted that the linear model is better than other models. It is because of the character of input data (parameters of the web discussion) and also due to the character of the issue which is discussed on the web. Nevertheless, the linear model is sufficient for authority estimation. It is clear that learning from the "wisdom of the crowd" is better than learning from a "human expert". The reason might be that an expert's opinion can be biased whereas a combined opinion of many discussants is probably more objective.

In our future work, we want to combine knowledge gained from opinion classification and authority identification. We suppose that a more authoritative author has a greater influence on the resulting summarized sentiment. In the known approaches to sentiment analysis, each web forum comment has the same weight. We will use evolutionary algorithms [18, 5] to find an appropriate form of estimation function to calculate the authority value and then apply it to opinion classification. The comments written by authoritative users have higher weight and they will be classified with higher priority. We would like to apply the weighted opinion analysis in the domain where we could be able to recognize a person's aberration based on his/her written text [24] and also how to decrease the cognitive load for the web users [19].

Acknowledgements

The work presented in this paper was supported by the Slovak Grant Agency of the Ministry of Education and Academy of Science of the Slovak Republic under VEGA grant No. 1/0493/16 and by the Slovak Research and Development Agency under the contract No. APVV-017-0267.

REFERENCES

[1] Baccianella, S.—Esuli, A.—Sebastiani, F.: SentiWordNet 3.0: An Enhanced Lexical Resource for Sentiment Analysis and Opinion Mining. Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC '10), Valletta, 2010. European Language Resources Association (ELRA), 2010, pp. 2200–2204. ISBN 2-9517408-6-7.

[2] Balahur, A.—Kabadjov, M.—Steinberger, J.—Steinberger, R.—Montoyo, A.: Challenges and Solutions in the Opinion Summarization of User-Generated Content. Journal of Intelligent Information Systems, Vol. 39, 2012, No. 2, pp. 375–398, doi: 10.1007/s10844-011-0194-z.

[3] Benamara, F.—Cesarano, C.—Picariello, A.—Recupero, D. R.—Subramanian, V. S.: Sentiment Analysis: Adjectives and Adverbs are Better Than Adjectives Alone. Proceedings of the First International Conference on Weblogs and Social Media (ICWSM 2007), Boulder, 2007.

[4] Bouguessa, M.—Dumoulin, B.—Wang, S.: Identifying Authoritative Actors in Question-Answering Forums – The Case of Yahoo! Answers. Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '08), 2008, pp. 866–874, doi: 10.1145/1401890.1401994.

[5] Cadrik, T.—Mach, M.: Evolution Classifier Systems. Electrical Engineering and Informatics IV. Proceedings of the FEI Technical University of Košice, Košice, 2013, pp. 168–172 (in Slovak). ISBN 978-80-553-1440-2.

[6] CAMBRIA, E.—HUSSAIN, A.: Sentic Computing: A Common-Sense-Based Framework for Concept-Level Sentiment Analysis. Cham, Springer, 2015, 176 pp. ISBN 978-3-319-23654-4.

[7] DOM, B.—EIRON, I.—COZZI, A.—ZHANG, Y.: Graph-Based Ranking Algorithms for E-Mail Expertise Analysis. Proceedings of 8th ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD '03), 2003, pp. 42–48, doi: 10.1145/882082.882093.

[8] EKMAN, P.: An Argument for Basic Emotions. Cognition and Emotion, Vol. 6, 1992, No. 3-4, pp. 169–200, doi: 10.1080/02699939208411068.

[9] ESULI, A.—SEBASTIANI, F.: SentiWordNet: A Publicly Available Lexical Resource for Opinion Mining. Proceedings of the 5th Conference on Language Resources and Evaluation (LREC '06), 2006, pp. 417–422.

[10] GO, A.—BHAYANI, R.—HUANG, L.: Twitter Sentiment Classification Using Distant Supervision. Stanford University, available on: `http://cs.stanford.edu/people/alecmgo/papers/TwitterDistantSupervision09.pdf`, 2013.

[11] GRANDJEAN, D.—SANDER, D.—SCHERER, K. R.: Conscious Emotional Experience Emerges as a Function of Multilevel, Appraisal-Driven Response Synchronization. Consciousness and Cognition, Vol. 17, 2008, No. 2, pp. 484–495, doi: 10.1016/j.concog.2008.03.019.

[12] GUNES, H.—SCHULLER, B.: Categorical and Dimensional Affect Analysis in Continuous Input: Current Trends and Future Directions. Image and Vision Computing, Vol. 31, 2013, No. 2, pp. 120–136.

[13] HU, M.—LIU, B.: Mining and Summarizing Customer Reviews. Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04), ACM, New York, 2004, pp. 167–177. ISBN 1-58113-888-1, doi: 10.1145/1014052.1014073.

[14] KENNEDY, A.—INKPEN, D.: Sentiment Classification of Movie and Product Reviews Using Contextual Valence Shifters. Workshop on the Analysis of Informal and Formal Information Exchange During Negotiations (FINEXIN '05), Ottawa, 2005, pp. 11–22.

[15] LEVALLOIS, C.: Umigon: Sentiment Analysis for Tweets Based on Lexicons and Heuristics. Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013), Atlanta, Georgia, 2013, pp. 414–417.

[16] LIU, B.: Sentiment Analysis and Opinion Mining (Introduction and Survey). Morgan and Claypool Publisher, 2012, pp. 1–168.

[17] LU, Y.—KONG, X.—QUAN, X.—LIU, W.—XU, Y.: Exploring the Sentiment Strength of User Reviews. Proceedings of the 11th International Conference on Web-Age Information Management (WAIM '10). Springer-Verlag, Berlin, Lecture Notes in Computer Science, Vol. 6184, 2010, pp. 471–482. ISBN 978-3-642-14245-1, doi: 10.1007/978-3-642-14246-8_46.

[18] MACH, M.: Evolution Algorithms – Problems Solving. FEI Technical University, Košice, 2013, 135 pp. (in Slovak). ISBN 978-80-553-1445-7.

[19] Machová, K.—Klimko, I.: Classification and Clustering Methods in the Decreasing of the Internet Cognitive Load. Acta Elektrotechnica et Informatica, Vol. 6, 2006, No. 2, pp. 52–56. ISSN 1335-8243.

[20] Machova, K.—Marhefka, L.: Opinion Classification in Conversational Content Using N-Grams. In: Badica, A., Trawinski, B., Nguyen, N. (Eds.): Recent Developments in Computational Collective Intelligence. Springer, Cham, Studies in Computational Intelligence, Vol. 513, 2013, pp. 177–186.

[21] Osgood, C. E.—May, W. H.—Miron, M. S.: Cross-Cultural Universals of Affective Meaning. University of Illinois Press, Urbana, 1975.

[22] Pang, B.—Lee, L.: A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics (ACL '04). Association for Computational Linguistics, Stroudsburg, 2004, doi: 10.3115/1218955.1218990.

[23] Plutchik, R.: The Nature of Human Emotions. Scienceweek, 3 August 2001, pp. 1–2.

[24] Saloun, P.—Ondrejka, A.—Malčík, M.—Zelinka, I.: Personality Disorders Identification in Written Texts. In: Duy, V., Dao, T., Zelinka, I., Choi, H. S., Chadli, M. (Eds.): Recent Advances in Electrical Engineering and Related Sciences (AETA 2015). Springer Verlag, Lecture Notes in Electrical Engineering, Vol. 371, 2016, No. 1, pp. 143–154. ISBN 978-331927245-0. ISSN 1876-1100, doi: 10.1007/978-3-319-27247-4_13.

[25] Smatana, M.—Koncz, P.—Paralič, J.: Semi-Automatic Annotation Tool for Aspect-Based Sentiment Analysis. FEI Technical University of Košice, 2013, pp. 1–3.

[26] Strapparava, C.—Valitutti, A.: WordNet-Affect: An Affective Extension of WordNet. Proceedings of the 4th International Conference on Language Re-Sources and Evaluation. European Language Resources Association (ELRA), 2004, pp. 1083–1086.

[27] Taboada, M.—Brooke, J.—Tofiloski, M.—Voll, K.—Stede, M.: Lexicon-Based Methods for Sentiment Analysis. Computational Linguistics, Vol. 37, 2011, No. 2, pp. 267–307, doi: 10.1162/COLI_a_00049.

[28] Tan, S.—Zhang, J.: An Empirical Study of Sentiment Analysis for Chinese Documents. Expert Systems with Applications, Vol. 34, 2008, No. 4, pp. 2622–2629. ISSN 0957-4174.

[29] Thelwall, M.—Buckley, K.—Paltoglou, G.—Cai, D.—Kappas, A.: Sentiment Strength Detection in Short Informal Text. Journal of the American Society for Information Science and Technology, Vol. 61, 2010, No. 12, pp. 2544–2558, doi: 10.1002/asi.21416.

[30] Wundt, W.: Grundriss der Psychologie [Fundamentals of Psychology]. 7th revised edition. Engelmann, Leipzig, 1905 (in German).

[31] Zhang, J.—Ackerman, M. S.—Adamic, L.: Expertise Networks in On-Line Communities: Structure and Algorithms. Proceedings of the 16th ACM International World Wide Web Conference (WWW '07), 2007, pp. 221–230, doi: 10.1145/1242572.1242603.

[32] Zhang, Z. et al.: Sentiment Classification of Internet Restaurant Reviews Written in Cantonese. Expert Systems with Applications, Vol. 38, 2011, No. 6, pp. 7674–7682. ISSN 0957-4174.

Kristína Machová graduated (M.Sc.) in 1985 at the Department of Technical Cybernetics at the Technical University in Košice. She defended her Ph.D. thesis in the field of machine learning in 1996. She is Associate Professor at the Department of Cybernetics and Artificial Intelligence of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. Her scientific research has focused on data mining within conversational web content, dictionary approach and machine learning approach to sentiment analysis, opinion and emotion classification to be applied in robotics, authority identification, text document processing using classification and clustering machine learning methods.

Martin Mikula received his M.Sc. degree in 2014 at the Department of Cybernetics and Artificial Intelligence at the Technical University in Košice. Currently, he is a postgraduate student. He studies business informatics at the Faculty of Electrical Engineering and Informatics at the Technical University in Košice. His research interests are big data, including knowledge discovery, data mining, text mining and sentiment analysis in conversational content.

Martina Szabóová graduated (M.Sc.) in 2012 at the Department of Cybernetics and Artificial Intelligence at the Technical University in Košice. She is in her last year of Ph.D. study. Her main research activities intersect two fields. The first one lays in natural language processing within the scope of emotion analysis from texts. The second field covers interactive learning systems in the social human-robot interaction.

Marián Mach graduated (M.Sc.) in 1985 at the Department of Technical Cybernetics at the Technical University in Košice. His Ph.D. thesis on uncertainty processing in expert systems was defended in 1992. He is Associate Professor at the Department of Cybernetics and Artificial Intelligence of the Faculty of Electrical Engineering and Informatics at the Technical University of Košice. His scientific interests are knowledge management, data and web mining, opinion classification, information retrieval, semantic technologies, and knowledge modeling.

Computing and Informatics, Vol. 37, 2018, 759–780, doi: 10.4149/cai_2018_3_759

MALWARE DETECTION USING A HETEROGENEOUS DISTANCE FUNCTION

Martin Jureček, Róbert Lórencz

Faculty of Information Technology
Czech Technical University in Prague
Thákurova 9, 160 00 Prague, Czech Republic
e-mail: {martin.jurecek, lorencz}@fit.cvut.cz

Abstract. Classification of automatically generated malware is an active research area. The amount of new malware is growing exponentially and since manual investigation is not possible, automated malware classification is necessary. In this paper, we present a static malware detection system for the detection of unknown malicious programs which is based on combination of the weighted k-nearest neighbors classifier and the statistical scoring technique from [12]. We have extracted the most relevant features from portable executable (PE) file format using gain ratio and have designed a heterogeneous distance function that can handle both linear and nominal features. Our proposed detection method was evaluated on a dataset with tens of thousands of malicious and benign samples and the experimental results show that the accuracy of our classifier is 98.80 %. In addition, preliminary results indicate that the proposed similarity metric on our feature space could be used for clustering malware into families.

Keywords: Malware detection system, feature selection, similarity measure, k-nearest neighbors classifier, partitioning around medoids

1 INTRODUCTION

The problem of automated malware detection presents challenges for antivirus vendors (AV). Most AV rely primarily on a signature detection technique which is relatively simple and efficient rule-based method for detecting known malware [10]. Signature (unique hex code strings) of the malware is extracted and added to the database. The antivirus engine compares the contents of a file with all malware signatures in its database and if a match is found, the file is reported as malware. A good signature must capture malware with a minimal false positive probability.

The major weakness of signature detection is its inability to detect obfuscated and zero-day malware. A number of non-signature based malware detection techniques have been proposed [19, 11, 20]. These techniques are used in an effort to detect new or unknown malware and can be grouped into two main approaches: static and dynamic heuristic methods. Static methods can be based on an analysis of the file format without actually running the program. Dynamic analysis aims to examine a program which is executed in a real or virtual environment. Non-signature based malware detection techniques suffer from two main problems: high false positive rate and large processing overhead.

In this paper, we present a static malware detection system based on combination of the statistical classifier and the k-nearest neighbors (KNN) classifier. Experimental results indicate that the combination of the classifiers may provide a potential benefit for detecting samples not detected by KNN.

In our work we propose the following four main contributions:

- We present a feature space extracted from PE file format by using feature selection method based on information gain.
- We design a new distance function that can handle both nominal and linear attributes.
- We present a malware detection system for detecting previously unknown malicious PE files. In order to achieve a higher detection rate, the system uses a combination of two different kinds of classifiers.
- We evaluate the effectiveness of our detection system and distance function on a real-world malware collection.

The rest of the paper is organized in the following way. Section 2 provides an overview of previous work on malware classification. In Section 3 we present the feature space and the distance function used in KNN classifier. Section 4 discusses our proposed detection technique, while Section 5 covers our experimental results. Finally, conclusions are given in Section 6.

2 RELATED WORK

In this section, we survey some relevant previous work in the area of classification schemes for malware detection. To maintain the focus, we mainly discuss the work using static detection based on machine learning techniques. Then we briefly discuss various existing statistical-based scores and also several methods that rely on dynamic analysis.

Schultz et al. [19] introduced the concept of data mining for detecting previously unknown malware. In their research they presented three different static feature sources for malware classification: information from the portable executable

(PE) header and strings and byte sequences extracted from binaries. These features were used in three different kinds of algorithms: an inductive rule-based learner, a probabilistic method, and a multi-classifier system. A rule induction algorithm called Ripper [4] was applied to find patterns in the dynamic-link library (DLL) data (such as the list of DLLs used by the binary, the list of DLL function calls, and the number of different system calls used within each DLL). The well-known probabilistic method, learning algorithm Naive Bayes, was used to find patterns in the string data and n-grams of byte sequences. Multinomial Naive Bayes algorithm that combined the output of several classifiers reached the highest detection rate of 97.76 %. The authors tested the data mining methods against standard signatures and their results indicate that the data mining detection rate of a previously unknown malware was twice as high in comparison to the signature-based methods.

Kolter and Maloof [11] improved the Schulz's third technique by using overlapping byte sequences instead of non-overlapping sequences. They used different kinds of classifiers: naive Bayes, instance-based learner, similarity-based classifier called TFIDF, Support Vector Machine (SVM), Decision Trees (DT) and boosted variants of SVM, DT and TFIDF. Authors evaluated their classifiers performance by computing the area under a receiver operating characteristic curve. Boosted Decision tree model (J48) achieved the best accuracy, an area under the ROC curve of 0.996 and outperformed the rest of the classifiers.

In other studies, operational code (opcode) has been used as static information for malware detection. Common techniques are based on the frequency of appearance of opcode-sequences [18], examination of opcode frequency distribution difference between malicious and benign code [2], or identification of critical instruction sequences [22]. Other techniques use similarity of executables based on opcode graphs [17]. However, some executable files cannot be disassembled properly, therefore the opcode approach is not always feasible [2].

The more recent work [23] contains three statistical-based scoring techniques, namely Hidden Markov models, Simple substitution distance, and Opcode graph-based detection. Authors showed that a combination of these scoring techniques with a Support Vector Machine yields significantly more robust results than those obtained using any of the individual scores.

We also briefly mention a few existing detection methods that rely on dynamic analysis. Examples of the information we can obtain from dynamic analysis include application programming interface (API) and system calls, instruction traces, memory writes, registry changes, and so on. In [24], an artificial neural network was employed to detect previously unknown worms based on the computer's behavioral measures. Eskandari et al. [25] extracted a set of program API calls and combined them with control flow graph. Qiao et al. [26] proposed a new malware analysis method based on frequency analysis of API call sequences. Note that dynamic analysis is time-consuming as each malware sample must be executed for a certain time period.

3 FEATURE SPACE AND METRIC

We design our proposed detection system for the portable executable (PE) file format [5], which is the most widely used file format for malware samples. In order to classify an executable file in the PE format, we extract static format information and translate it into a feature vector suitable for classification.

3.1 Feature Space

Before presenting attributes used in our feature vector, let us firstly look at the general outline of the PE file format. A simplified overview of the PE file format is illustrated in Figure 1.

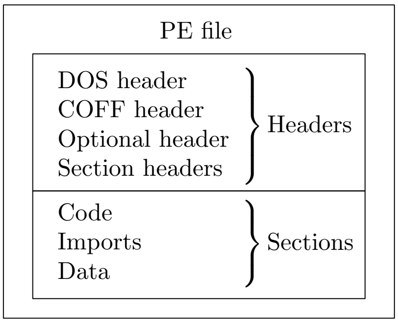


Figure 1. PE file structure

A PE file consists of headers and sections that encapsulate the information necessary to manage the executable code. The PE file header provides all the descriptive information concerning the locations and sizes of structures in the PE file to the loader process. The header of a PE file consists of the DOS header, the PE signature, the COFF file header, the optional header, and the section headers. The optional file header is followed immediately by the section headers which provide information about sections, including locations, sizes, and characteristics. Sections divide the file content into code, resources, and various types of data.

Based on our empirical studies and analysis of the PE format, we selected a set of static features that are helpful in distinguishing malware and benign files and used gain ratio for selection the most relevant features.

3.1.1 Features Selection

In order to determine which attribute in a given training set is the most useful for discriminating between the classes, we use entropy-based measure, information gain (IG) [13]. The information gain is the expected reduction in entropy caused by knowing the value of attribute a. $\mathrm{IG}(\mathcal{T}, a)$ of an attribute a relative to training dataset $\mathcal{T}$ and is defined as

$$\mathrm{IG}(\mathcal{T}, a) = \mathrm{Entropy}(\mathcal{T}) - \sum_{v \in V(a)} \frac{|\mathcal{T}_v|}{|\mathcal{T}|} \mathrm{Entropy}(\mathcal{T}_v) \tag{1}$$

where $V(a)$ denotes the set of all possible values for attribute a, and $\mathcal{T}_v$ denotes the subset of $\mathcal{T}$ for which attribute a has value v. Note that the entropy of the training dataset $\mathcal{T}$ is given by:

$$\mathrm{Entropy}(\mathcal{T}) = -\sum_{c \in C} p_c \log_2 p_c \tag{2}$$

where p_c is the proportion of $\mathcal{T}$ belonging to class c.

The information gain measure is biased towards attributes with many values. One way of avoiding this difficulty is to use a modification of the measure called the gain ratio (GR) [15]. The gain ratio measure penalizes attributes with large numbers of possible values by incorporating a term called split information (SI):

$$\mathrm{SI}(\mathcal{T}, a) = -\sum_{i=1}^{d} \frac{|\mathcal{T}_i|}{|\mathcal{T}|} \log_2 \frac{|\mathcal{T}_i|}{|\mathcal{T}|} \tag{3}$$

where $\mathcal{T}_i$ are the d subsets of training dataset $\mathcal{T}$ resulting from partitioning $\mathcal{T}$ by the d-valued attribute a. Split information $\mathrm{SI}(\mathcal{T}, a)$ is the entropy of $\mathcal{T}$ with respect to the values of attribute a. The gain ratio is then defined as

$$\mathrm{GR}(\mathcal{T}, a) = \frac{\mathrm{IG}(\mathcal{T}, a)}{\mathrm{SI}(\mathcal{T}, a)} \tag{4}$$

and we select only features with the highest values of gain ratio.

3.1.2 Our Proposed Feature Space

The following feature set was extracted using gain ratio and used in our work:

- Many fields from the PE headers, such as the number of sections, date/time stamp, major or minor versions of linker, operating system, image, subsystem; sizes and addresses of data directories; DLL characteristics, etc. Table 1 lists all features that are derived from the PE headers. For detailed description of these features, see Chapter 3 in [5].
- Features from sections and their headers: VirtualSize, VirtualAddress, SizeOfRawData, PointerToRawData, Section Flags (see Chapter 4 in [5]), and features not contained within the PE structure, including entropies and checksums of sections.
- Resources of the PE file are used to provide supporting content, such as icons, fonts, strings and other elements. In case of malicious files, resources are often

used to store code and configuration data. For example, the number of resources and the number of types of resources were used in our work.

- Overlay is a data that is appended at the end of the executable file. We considered the size of the overlay.
- Other features: the size of all imports, the number of DLLs referred, the number of APIs referred.

Feature	*Feature*
NumberOfSections	MajorOperatingSystemVersion
TimeDateStamp	MajorImageVersion
SizeOfOptionalHeader	MajorSubsystemVersion
Characteristics	MinorSubsystemVersion
MajorLinkerVersion	SizeOfImage
MinorLinkerVersion	CheckSum
AddressOfEntryPoint	Subsystem
ImageBase	DllCharacteristics
SectionAlignment	NumberOfRvaAndSizes
FileAlignment	Addresses and sizes of data directories

Table 1. List of features from the PE headers

Note that our feature set is similar to that in the existing works [12, 21, 1].

3.2 Distance Function

Many classifiers require some measure of dissimilarity or distance between feature vectors, and its performance depends upon a good choice of distance function. Especially the KNN classifier depends significantly on the metric used to compute distances between two feature vectors.

In this section, we propose a similarity metric on our feature space. We used this metric to compute distances to find k nearest neighbors used in KNN classifier. Note that the features used in our work are of various types and sizes. These features can be divided into three types: numbers, bit fields, and strings. For example, number of sections or various addresses can be represented by numbers, section flags or characteristics by bit fields, and checksums by strings. Furthermore, some features have different ranges. For example, the number of sections is considerably smaller than the total number of called API functions. The proposed distance function can handle these types of features and also takes into account their different ranges.

The most commonly used metric is the Euclidean distance which works well for numerical attributes. However, it does not appropriately handle nominal attributes. The Value Difference Metric (VDM) [27] was proposed to define a suitable distance function for nominal attributes. A version of the VDM without a weighting scheme is defined for values x and y of an attribute a as:

$$\mathrm{VDM}_a(x,y) = \sum_{c=1}^{C} \left| \frac{n_{a,x,c}}{n_{a,x}} - \frac{n_{a,y,c}}{n_{a,y}} \right|^q \tag{5}$$

where

- C is the number of classes,
- $n_{a,x,c}$ is the number of instances in the training set $\mathcal{T}$ which have value x for attribute a and the instance belongs to class c,
- $n_{a,x}$ is the number of instances in $\mathcal{T}$ that have value x for attribute a.

Since the Euclidean distance is not suitable for nominal attributes, and VDM is inappropriate for numeric attributes, heterogeneous metric can be used to handle our feature space. Wilson and Martinez introduced Heterogeneous Value Difference Metric (HVDM) [29] which is defined for feature vectors $\mathbf{x}$ and $\mathbf{y}$ as:

$$\mathrm{HVDM}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)} \tag{6}$$

where m is the number of attributes of the feature vector and the definition of distance function $d_a(x,y)$ depends on the type of attribute a as follows:

$$d_a(x,y) = \begin{cases} 1, & \text{if } x \text{ or } y \text{ is unknown,} \\ \mathrm{NORM\_VDM}_a(x,y), & \text{if } a \text{ is nominal,} \\ \mathrm{NORM\_DIFF}_a(x,y), & \text{if } a \text{ is linear.} \end{cases} \tag{7}$$

Functions $\mathrm{NORM\_VDM}_a(x,y)$ and $\mathrm{NORM\_DIFF}_a(x,y)$ are defined as:

$$\mathrm{NORM\_VDM}_a(x,y) = \sum_{c=1}^{C} \left| \frac{n_{a,x,c}}{n_{a,x}} - \frac{n_{a,y,c}}{n_{a,y}} \right|, \tag{8}$$

$$\mathrm{NORM\_DIFF}_a(x,y) = \frac{|x-y|}{4\sigma_a} \tag{9}$$

where σ_a is the standard deviation of the values of numeric attribute a.

3.2.1 Our Proposed Distance Function

Since the feature vector described in Section 3.1.2 contains more types of nominal attributes we propose the following distance function:

$$\mathcal{D}(\mathbf{x},\mathbf{y}) = \sqrt{\sum_{a=1}^{m} d_a^2(x_a, y_a)} \tag{10}$$

where

$$d_a(x,y) = \begin{cases} \mathcal{H}(x,y), & \text{if } a \text{ is an array of bits,} \\ \delta(x,y), & \text{if } a \text{ is a checksum,} \\ \text{NORM\_DIFF}_a(x,y), & \text{if } a \text{ is numeric,} \\ \text{NORM\_VDM}_a(x,y), & \text{otherwise } (a \text{ is a string).} \end{cases} \tag{11}$$

$\mathcal{H}(x,y)$ denotes Hamming distance defined for binary $x = (x_1, \dots, x_n), y = (y_1, \dots, y_n)$ as

$$\mathcal{H}(x,y) = |\{i \mid x_i \neq y_i, i = 1, \dots, n\}| \tag{12}$$

and $\delta(x,y)$ is the characteristic function defined as

$$\delta(x,y) = \begin{cases} 0, & \text{if x = y,} \\ 1, & \text{otherwise.} \end{cases} \tag{13}$$

Since the distance functions d_a are metrics, $\mathcal{D}$ is also a metric. Properties of a metric, especially the triangle inequality, can be used in effective finding of nearest neighbors in a metric space.

Note that we distinguish between a checksum and a string attribute that is not a checksum. For the demonstration of distance function $\mathcal{D}$ on our feature space, we present the examples of attributes of each type:

- array of bits: Section flags, Characteristics, DllCharacteristics,
- numeric attribute: number of sections, number of DLLs, size of all imports,
- checksum: checksums of various pieces of the file content,
- string: major/minor version of linker, operating system, subsystem.

4 PROPOSED SYSTEM FOR DETECTING MALWARE

In this section, we present a system for detecting malware which is composed of a KNN classifier and a statistical scoring technique.

4.1 The k-Nearest Neighbors Classifier

The k-nearest neighbors (KNN) classifier is one of the most popular supervised learning methods introduced by Fix and Hodges [8]. It is one of the simplest and best-known nonparametric algorithms in pattern classification.

Let $T = \{(x_1, c_1), \dots, (x_m, c_m)\}$ be the training set, where x_i is training vector and c_i is the corresponding class label. Given a query point x_q, its unknown class c_q is determined as follows. First, select the set $T' = \{(x_1, c_1), \dots, (x_k, c_k)\}$ of k nearest neighbors to the query point x_q. Then assign the class label to the query point x_q by majority vote of its nearest neighbors:

$$c_q = \arg\max_c \sum_{(x_i,c_i)\in T'} \delta(c, c_i) \tag{14}$$

where c is a class label, c_i is the class label for i^{th} neighbor among k nearest neighbors of the query point, and $\delta(c, c_i)$ takes a value of one if $c = c_i$ and zero otherwise. Cover and Hart [6] found that if the number of samples approaches infinity, the nearest-neighbor error rate is bounded from above by twice the Bayes error rate.

Distance-weighted k-nearest neighbor procedure (WKNN) was first introduced in [7] as an improvement to KNN. This extension is based on the idea that closer neighbors are weighted more heavily than such neighbors that are far away from the query point. KNN implicitly assumes that all k nearest neighbors are equally important in making a classification decision, regardless of their distances to the query point. In WKNN, nearest neighbors are weighted according to their distances to the query point as follows. Let $x_1, \ldots, x_k$ be k nearest neighbors of the query object and $d_1, \ldots, d_k$ the corresponding distances arranged in increasing order. The weight w_i for i-th nearest neighbor is defined as:

$$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1}, & \text{if } d_k \neq d_1, \\ 1, & \text{otherwise.} \end{cases} \tag{15}$$

The resulting class of the query point is then defined by the majority weighted vote as follows:

$$c_q = \arg\max_c \sum_{(x_i,c_i)\in T'} w_i \cdot \delta(c, c_i). \tag{16}$$

Note that finding the nearest neighbors is a very expensive operation due to the enormous size of our dataset. The nearest neighbors can be found more efficiently by representing the training dataset as a tree.

4.2 The Statistical-Based Classifiers

In this section, we present the scoring techniques that we used in our research. In the case of the statistical-based classifier, we ignore the positions of points in our metric space and we focus on statistical properties of attribute values, in contrast to the KNN classifier.

4.2.1 Naive Bayes

This section introduces the Naive Bayes classifier [28] for binary (two-class) classification problems. A Naive Bayes classifier is a probabilistic algorithm based on Bayes' Theorem that predicts the class with the highest *a posteriori* probability. Assume a set of two classes $\{\mathcal{C}, \mathcal{M}\}$, where $\mathcal{C}$ denotes the class of benign samples and $\mathcal{M}$ denotes the class of malware. Training datasets are provided and a new (unknown) sample, which is represented by a feature vector $x = (x_1, \ldots, x_n)$, is presented. Let $P(\mathcal{M}|x)$ denote the probability that a sample is malicious given the feature vector x that describes the sample. Similarly, $P(\mathcal{C}|x)$ denotes the probability that a sample is benign given the feature vector x that represents the sample. The Naive Bayes classification rule is stated as

$$\begin{aligned}&\text{If } P(\mathcal{M}|x) < P(\mathcal{C}|x), x \text{ is classified as benign sample,}\\ &\text{If } P(\mathcal{M}|x) > P(\mathcal{C}|x), x \text{ is classified as malware.}\end{aligned} \tag{17}$$

The *a posteriori* probabilities $P(C|x)$ may be expressed in terms of the *a priori* probabilities and the $P(x|C)$ probabilities using Bayes' theorem as

$$P(C|x) = \frac{P(x|C)\ P(C)}{P(x)}. \tag{18}$$

Assuming that the values of the attributes (features) are conditionally independent on one another, Equation (18) may be expressed as

$$P(C|x) = \frac{\prod_{i=1}^{n} P(x_i|C)\ P(C)}{P(x)}. \tag{19}$$

Probabilities $P(x_i|C)$ can be estimated from the training set by counting the attribute values for each class. More precisely, the probability $P(x_i = h|C)$ is represented as the number of samples of class C in the training set having the value h for attribute x_i, divided by the number of samples of class C in the training set. The output of the classifier is the highest probability class C':

$$C' = \arg\max_{C} \left(P(C) \prod_{i=1}^{n} P(x_i|C) \right). \tag{20}$$

4.2.2 Statistical Classifier – STATS

The following statistical classifier was introduced in [12]. Let $x = (x_1, \ldots, x_n)$ be a vector from our feature space and $\mathcal{M}$ a class of malware. Then probability

$$P(x \in \mathcal{M}|x_i = h) = \frac{n_{x_i,h,\mathcal{M}}}{n_{x_i,h}} \tag{21}$$

is the conditional probability that the output class of x is malware given that attribute x_i has the value h. Denote this probability by p_i, $i = 1, \ldots, n$. Note that the notations $n_{x_i,h,\mathcal{M}}$ and $n_{x_i,h}$ were used in the definition of VDM discussed in Section 3.2. Define a function f with two parameters p_i and S_c as

$$f(p_i, S_c) = \max\{0, p_i - S_c\} \tag{22}$$

where S_c is an empirical constant. For each file x we define a score as

$$\text{score} = \sum_{i=1}^{n} f(p_i, S_c). \tag{23}$$

From this score, we can determine a threshold S_s, above which we will classify a file as malware. The decision rule is then defined as follows:

$$x \text{ is classified as } \begin{cases} \text{malware}, & \text{if score} > S_s, \\ \text{benign file}, & \text{otherwise.} \end{cases} \tag{24}$$

The pseudocode of the statistical-based classifier is described in Algorithm 1.

Algorithm 1 Statistical classifier – STATS

```
Input: original training set, query point $x$, distance metric $\mathcal{D}$
Output: label of $x$
 1: score = 0
 2: Compute probability vector $(p_1, \ldots, p_n)$
 3: for $i = 1$ to $n$ do
 4:   if $p_i > S_c$ then
 5:     score += $p_i - S_c$
 6:   end if
 7: end for
 8: if score > $S_s$ then
 9:   return malware
10: else
11:   return benign file
12: end if
```

In the rest of this paper, the statistical-based classifier is denoted as STATS.

4.3 Our Approach

We propose a malware detection approach based on a combination of the well-known KNN classifier and the chosen statistical motivated classifier. In order to achieve higher detection rates, there should be some kind of diversity between the classifiers. KNN is a geometric-based classifier which uses labels of the nearest neighbors in some metric space to classify an unlabeled point. On the other hand, statistical-based approaches like Naive Bayes or the STATS classifier mentioned above use conditional probabilities of attributes of sample point and do not use information about its position in feature space.

The proposed detection method works as follows. First, set the threshold to some sufficiently high value. Then compute score using the chosen statistical scoring technique. If the score of the unknown file exceeds the threshold, then the resulting class will be malware, otherwise apply distance-weighted KNN. The pseudocode for the classification scheme is shown in Algorithm 2.

Algorithm 2 Our detection system

```
Input: original training set, query point $x$, distance metric $\mathcal{D}$
Output: label of $x$
1: compute score from the statistical scoring technique
2: if score > threshold then
3:     return malware
4: else
5:     apply WKNN
6: end if
```

The reason why we chose KNN is that it is a relatively accurate classifier for large datasets and the results of our experiments demonstrate that the statistical classifier is able to correctly classify samples lying in the area of feature space where the accuracy of KNN is low. The statistical classifier uses information from the training dataset in a different way than the KNN classifier. It checks whether a feature vector contains values typical for malware, in contrast to KNN that considers only differences between feature vectors.

For example, consider a feature vector $x = (x_1, \dots, x_n)$ containing only a few values (typically checksums), for which there is a high probability that x belongs to malware. Many other attributes could have previously unseen values or ones with a low prevalence. Therefore, malicious nearest neighbors could not be closer than benign nearest neighbors and in this case, KNN classifier would not be an appropriate method.

4.3.1 The System Architecture

The system consists of three major components: a PE parser, a database of conditional probabilities and a classification module, as illustrated in Figure 2.

The functionality of the PE parser is to extract all PE format file's header information, DLLs, and API functions from programs in the dataset and store all the extracted information in a database. Recall that our system is applied only to Windows PE files and the PE parser extracts only the most useful features for discriminating between benign and malicious files. These features were determined by the feature selection algorithm mentioned in Section 3.

During the training phase, once the structural information of the PE files is extracted, the conditional probabilities $P(x \text{ is malware} | x_i = h)$ are computed for each PE attribute x_i and for each possible value h of attribute x_i. Note that only the PE features extracted from labeled samples of the training dataset are used in the computation of the conditional probabilities.

After extracting PE features and computing the conditional probabilities, feature vectors are created for every known PE file. The set of these feature vectors

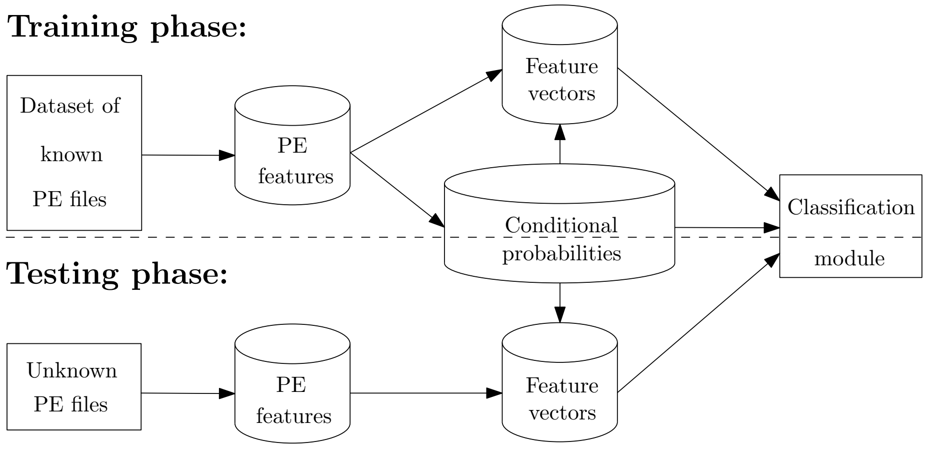


Figure 2. Architecture of the classification model

called training set will be used in the classification module where the classification algorithm is applied to feature vectors of unknown PE files.

5 EVALUATION RESULTS AND ANALYSIS

In this section, we introduce the performance metrics and present the results of our experiments. We compare our approach with several other machine learning methods for malware detection.

5.1 Performance Metrics

We present the evaluation metric we used to measure the accuracy of our proposed approach for the detection of unknown malicious codes. For evaluation purposes, the following classical quantities are employed:

- True Positive (TP) represents the number of malicious samples classified as malware,
- True Negative (TN) represents the number of benign samples classified as benign,
- False Positive (FP) represents the number of benign samples classified as malware,
- False Negative (FN) represents the number of malicious samples classified as benign.

The performance of our classifier on the test set is measured using three standard parameters. The most intuitive and commonly used evaluation measure in Machine

Learning is the Accuracy (ACC):

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}. \tag{25}$$

It is defined on a given test set as the percentage of correctly classified instances. However, since our dataset is not well-balanced, the accuracy measure could be an inappropriate measure of performance. If we use a classifier which labels every sample as benign, then TN will be very high and TP will be very low. As a result, the accuracy obtained on our dataset will be very high.

The second parameter, True Positive Rate (TPR) (or detection rate), is defined as:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}. \tag{26}$$

TPR is the percentage of truly malicious samples that were classified as malware. The third parameter is False Positive Rate (FPR) and is defined as follow:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}. \tag{27}$$

FPR is the percentage of benign samples that were wrongly classified as malware.

We also evaluate our classifier using Receiver Operating Characteristic (ROC) analysis [3]. ROC curve is represented as a two-dimensional plot, in which true positive rate is plotted against false positive rate at various threshold settings. The area under the ROC curve (AUC) serves as the performance measure of our detection techniques. An AUC of 1.0 represents the ideal case where both false positive and false negative equal zero. On the other hand, AUC of 0.5 means that the classifier's performance is no better than flipping a coin.

5.2 Dataset

The dataset used in this research consists of a total of 101,604 Windows programs in the PE file format, out of which 21,087 are malicious and 80,517 are legitimate or benign programs. There were no duplicate programs in our dataset. The malicious and benign programs were obtained from the laboratory of the industrial partner.

In order to expose any biases in the data, we used the 5-fold cross-validation procedure. Generally in k-fold cross-validation [14], the dataset is randomly divided into k subsets of equal size, where k-1 subsets are used for training and 1 subset is used for testing. In each of the k folds a different subset is reserved for testing and the accuracies obtained for each fold are averaged to produce a single cross validation estimate.

In the cluster analysis we used five prevalent malware families that have appeared during the year 2016. Specifically, we have used the following malware families:

- Allaple – a polymorphic network worm that spreads to other computers and performs denial-of-service (DoS) attacks.

- Dinwod – a trojan horse that silently downloads and installs other malware on the compromised computer.
- Virlock – a ransomware that locks victims' computer and demands a payment in order to unlock it.
- Virut – a virus with backdoor functionality that operates over an IRC-based communications protocol.
- Vundo – a trojan horse that displays pop-up advertisements and also injects JavaScript into HTML pages.

5.3 Classification Results

We implemented the classifiers as described in Section 4. The feature space and the distance function proposed in Section 3 were used in the KNN and the WKNN classifiers. The combination of the WKNN and the statistical scoring technique from [12] is denoted as WKNN_STATS and the combination of the WKNN and the Naive Bayes classifier is denoted as WKNN_NB. For each experiment, we performed 5-fold cross-validation that gives approximately unbiased estimate of a classifier's accuracy.

In our first experiment, we attempt to distinguish between benign and malicious PE files. We used accuracy, discussed in Section 5.1, as a comparison criterion for comparing classifiers. The accuracies obtained after applying the WKNN and the KNN classifiers for various numbers of nearest neighbors are depicted in Figure 3.

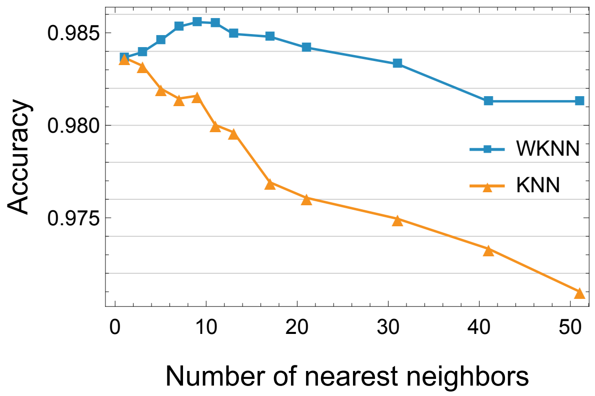


Figure 3. Classification accuracies of the WKNN and the KNN classifiers, for various numbers of nearest neighbors

The WKNN classifier achieved the highest accuracy using nine nearest neighbors, while the KNN classifier achieved the highest accuracy using only three nearest neighbors.

The classification results of the classifiers implemented in this research are listed in Table 2.

Classifier	*TPR*	*FPR*	*Accuracy*
KNN	96.23 %	1.07 %	98.37 %
WKNN	96.82 %	0.99 %	98.56 %
NB	82.78 %	1.17 %	95.50 %
STATS	90.08 %	**0.76** %	97.34 %
WKNN_NB	97.37 %	1.05 %	98.62 %
WKNN_STATS	**98.08** %	1.01 %	**98.80** %

Table 2. Classification results of six approaches implemented in this work

Among these classifiers, the WKNN_STATS outperformed others with the highest accuracy of 98.8 %. Note that the WKNN_STATS classifier was tested for various threshold values, and the best result was achieved with the following parameters:

- the number of nearest neighbors $k = 9$ used in the WKNN classifier,
- the thresholds $S_c = 0.8$ and $S_s = 0.53$ used in the STATS classifier.

In addition to that, we constructed the ROC curves which are shown in Figure 4 for three chosen classifiers.

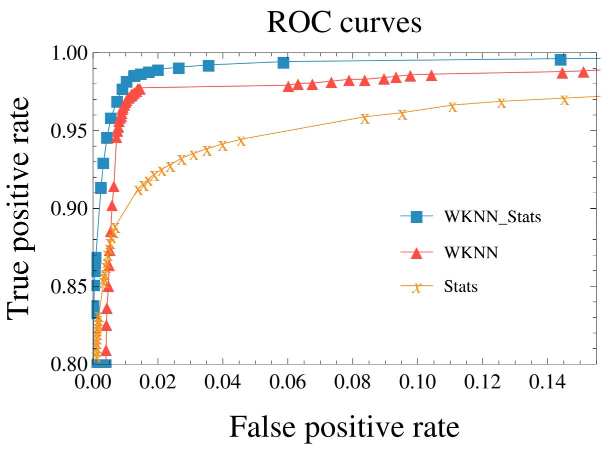


Figure 4. ROC curves for the WKNN, STATS and WKNN_STATS classifiers

We can conclude from Figure 4 that a combination of the classifiers outperforms both individual classifiers.

Table 3 reports the AUC for three classifiers discussed in Section 4 and two related static methods: KM [11] and PE Miner [21]. As the table illustrates, WKNN_STATS classifier provides the best AUC value with 0.998. The ROC curve and AUC values confirm that our experiment provides excellent results regarding malware detection.

Classifier	*AUC*
WKNN	0.993
STATS	0.983
WKNN_STATS	**0.998**
KM	0.996
PE Miner	0.992

Table 3. Comparison of the AUC value for five static methods

5.4 Clustering Results

In the second experiment we apply cluster analysis to five prevalent malware families described in Section 5.2. First, we present the clustering algorithm used in this experiment and then describe the evaluation measures and show the results.

5.4.1 Partitioning Around Medoids

Partitioning around medoids (PAM) proposed by Kaufman and Rousseeuw [9] is a well-known technique for performing non-hierarchical clustering. The reason why we have decided to use the PAM algorithm is that it allows clustering with respect to any distance metric. The pseudocode of the PAM algorithm is described in Algorithm 3.

Algorithm 3 PAM algorithm

```
Input: Number of clusters $k$, set of data points $T$
Output: $k$ clusters
1: Initialize: randomly select $k$ data points from $T$ to become the medoids
2: Assign each data point to its closest medoid
3: for all cluster do
4:    identify the observation that would yield the lowest average distance if it were
      to be re-assigned as the medoid
5:    if the observation is not current medoid then
6:       make this observation the new medoid
7:    end if
8: end for
9: if at least one medoid has changed then
10:   go to step 2
11: else
12:   end the algorithm.
13: end if
```

5.4.2 Evaluation Measures

We evaluated the quality of clusters through the measures of purity and silhouette coefficient (SC). Let n_{ij} be the number of samples of class i in cluster C_j and let $p_{ij} = \frac{n_{ij}}{|C_j|}$. The probability p_{ij} is the probability that a randomly selected sample from cluster C_j belongs to class i. The purity of cluster C_j is defined as $\text{Purity}(C_j) = \max_i p_{ij}$.

The overall purity value is defined as the weighted sum of individual purities for each cluster, taking into account the size of each cluster:

$$\text{Purity} = \frac{1}{n} \sum_{j=1}^{k} |C_j| \text{Purity}(C_j). \tag{28}$$

To measure the quality of clusters, we compute the average silhouette coefficient [16] for each cluster. Suppose there are n samples $x_1, \ldots, x_n$ that have been divided into k clusters $C_1, \ldots, C_k$. Consider a sample $x_i \in C_j$, and define the average distance between x_i to all other samples in cluster C_j:

$$a(x_i) = \frac{1}{|C_j| - 1} \sum_{\substack{y \in C_j \\ y \neq x_i}} d(x_i, y). \tag{29}$$

Let $b_k(x_i)$ be the average distance from sample $x_i \in C_j$ to all samples in cluster C_k not containing x_i:

$$b_k(x_i) = \frac{1}{|C_k|} \sum_{y \in C_k} d(x_i, y). \tag{30}$$

After computing $b_k(x_i)$ for all clusters C_k, where $k \neq j$, we select the minimum of those numbers:

$$b(x_i) = \min_{k \neq j} b_k(x_i). \tag{31}$$

The silhouette coefficient of x_i is obtained by combining $a(x_i)$ and $b(x_i)$ as follows:

$$s(x_i) = \frac{b(x_i) - a(x_i)}{\max(a(x_i), b(x_i))}. \tag{32}$$

The value of $s(x_i)$ in Equation (32) can vary between -1 and 1. It is desirable to have the value $s(x_i)$ as close to 1 as possible, since then the clusters are well-separated. The average silhouette coefficient for a given cluster is defined as the average value of $s(x_i)$ over all samples in the cluster.

5.4.3 Experimental Results

We computed the silhouette coefficient, as discussed above. For computing the silhouette coefficient, we used our proposed distance function on the feature space

Majority Class	*Size*	*Purity*	*SC*
Allaple	424	0.9343	0.3298
Dinwod	285	0.7429	0.7172
Virlock	452	0.9771	0.5635
Virut	337	0.68	0.2389
Vundo	252	0.6886	0.1921
Overall	1 750	0.8298	0.3883

Table 4. The purity and the silhouette coefficient for clusters

discussed in Section 3. Table 4 summarizes the results of silhouette coefficient based experiments using the PAM algorithm.

According to the experiences of authors of SC [16], silhouette coefficient values between 0.7 and 1.0 indicate excellent clustering results. SC values between 0.5 and 0.7 indicate a reasonable structure of cluster. SC values below 0.25 indicate that no substantial structure has been found.

Regarding the clustering malware into families, our results show that the quality of clusters varies widely, depending on the particular family. From the results in Table 4, we see that the PAM algorithm can correctly classify the malware family with an accuracy of about 68 % to over 97 %, depending on the particular family.

Note that such accuracies are lower than those obtained with classifiers presented in Section 4. The reason is that distinguishing between malware families is a more challenging problem than a binary classification of malware and benign files.

6 CONCLUSION

In this paper, we proposed a new detection system using a combination of the k-nearest neighbors classifier and the statistical-based classifier. The system can automatically detect unknown malware samples. The feature set used in our work was a collection of properties extracted from the PE file format. We designed a new distance function that is capable of handling various types of features.

Experimental results indicate that the combination of the classifiers may provide a potential benefit to detect samples not detected by KNN. We compared the different classification methods and concluded that the combination of the weighted k-nearest neighbors classifier and the statistical-based classifier achieves the highest accuracy, 98.8 %. The results also indicate that the proposed heterogeneous distance function and the feature space are appropriate for malware detection and could be also used for clustering malware into families.

The proposed static malware detection system is relatively easy to implement, and can be utilized to support commercial antivirus systems. For future work, it would be interesting to experiment with additional statistical scoring techniques in the context of malware classification.

Acknowledgements

The authors acknowledge the support of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 "Research Center for Informatics".

REFERENCES

[1] Asquith, M.: Extremely Scalable Storage and Clustering of Malware Metadata. Journal of Computer Virology and Hacking Techniques, Vol. 12, 2016, No. 2, pp. 49–58, doi: 10.1007/s11416-015-0241-3.

[2] Bilar, D.: Opcodes as Predictor for Malware. International Journal of Electronic Security and Digital Forensics, Vol. 1, 2007, No. 2, pp. 156–168, doi: 10.1504/IJESDF.2007.016865.

[3] Bradley, A. P.: The Use of the Area Under the ROC Curve in the Evaluation of Machine Learning Algorithms. Pattern Recognition, Vol. 30, 1997, No. 7, pp. 1145–1159, doi: 10.1016/S0031-3203(96)00142-2.

[4] Cohen, W. W.: Learning Trees and Rules with Set-Valued Features. Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI/IAAI), Vol. 1, 1996, pp. 709–716.

[5] Microsoft Corporation: Visual Studio, Microsoft Portable Executable and Common Object File Format Specification, Revision 9.3, 2015.

[6] Cover, T.—Hart, P.: Nearest Neighbor Pattern Classification. IEEE Transactions on Information Theory, Vol. 13, 1967, No. 1, pp. 21–27, doi: 10.1109/TIT.1967.1053964.

[7] Dudani, S. A.: The Distance-Weighted k-Nearest-Neighbor Rule. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-6, 1976, No. 4, pp. 325–327, doi: 10.1109/TSMC.1976.5408784.

[8] Fix, E.—Hodges Jr., J. L.: Discriminatory Analysis – Nonparametric Discrimination: Consistency Properties. Technical Report, DTIC Document, 1951.

[9] Kaufman, L.—Rousseeuw, P. J.: Finding Groups in Data: An Introduction to Cluster Analysis. John Wiley and Sons, Wiley Series in Probability and Statistics, Vol. 334, 2009.

[10] Kephart, J. O.—Arnold, W. C.: Automatic Extraction of Computer Virus Signatures. 4th Virus Bulletin International Conference, 1994, pp. 178–194.

[11] Kolter, J. Z.—Maloof, M. A.: Learning to Detect and Classify Malicious Executables in the Wild. The Journal of Machine Learning Research, Vol. 7, 2006, pp. 2721–2744.

[12] Merkel, R.—Hoppe, T.—Kraetzer, C.—Dittmann, J.: Statistical Detection of Malicious PE-Executables for Fast Offline Analysis. In: De Decker, B., Schaumüller-Bichl, I. (Eds.): Communications and Multimedia Security (CMS 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6109, 2010, pp. 93–105.

[13] Mitchell, T. M.: Machine Learning. New York, 1997.

[14] Picard, R. R.—Cook, R. D.: Cross-Validation of Regression Models. Journal of the American Statistical Association, Vol. 79, 1984, No. 387, pp. 575–583, doi: 10.1080/01621459.1984.10478083.

[15] Quinlan, J. R.: Induction of Decision Trees. Machine Learning, Vol. 1, 1986, No. 1, pp. 81–106, doi: 10.1007/BF00116251.

[16] Rousseeuw, P. J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Journal of Computational and Applied Mathematics, Vol. 20, 1987, pp. 53–65, doi: 10.1016/0377-0427(87)90125-7.

[17] Runwal, N.—Low, R. M.—Stamp, M.: Opcode Graph Similarity and Metamorphic Detection. Journal in Computer Virology, Vol. 8, 2012, No. 1–2, pp. 37–52, doi: 10.1007/s11416-012-0160-5.

[18] Santos, I.—Brezo, F.—Nieves, J.—Penya, Y. K.—Sanz, B.—Laorden, C.—Bringas, P. G.: Idea: Opcode-Sequence-Based Malware Detection. In: Massacci, F., Wallach, D., Zannone, N. (Eds.): Engineering Secure Software and Systems (ESSoS 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5965, 2010, pp. 35–43.

[19] Schultz, M. G.—Eskin, E.—Zadok, F.—Stolfo, S. J.: Data Mining Methods for Detection of New Malicious Executables. Proceedings of the 2001 IEEE Symposium on Security and Privacy (S & P 2001), IEEE Computer Society, 2001, pp. 38–49, doi: 10.1109/SECPRI.2001.924286.

[20] Shabtai, A.—Moskovitch, R.—Elovici, Y.—Glezer, C.: Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-Art Survey. Information Security Technical Report, Vol. 14, 2009, No. 1, pp. 16–29, doi: 10.1016/j.istr.2009.03.003.

[21] Shafiq, M. Z.—Tabish, S. M.—Mirza, F.—Farooq, M.: PE-Miner: Mining Structural Information to Detect Malicious Executables in Realtime. In: Kirda, E., Jha, S., Balzarotti, D. (Eds.): Recent Advances in Intrusion Detection (RAID 2009). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5758, 2009, pp. 121–141.

[22] Siddiqui, M.—Wang, M. C.—Lee, J.: Data Mining Methods for Malware Detection Using Instruction Sequences. Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications (AIA '08), 2008, pp. 358–363.

[23] Singh, T.—Di Troia, F.—Corrado, V. A.—Austin, T. H.—Stamp, M.: Support Vector Machines and Malware Detection. Journal of Computer Virology and Hacking Techniques, Vol. 12, 2016, No. 4, pp. 203–212.

[24] Stopel, D.—Boger, Z.—Moskovitch, R.—Shahar, Y.—Elovici, Y.: Application of Artificial Neural Networks Techniques to Computer Worm Detection. Proceedings of the 2006 IEEE International Joint Conference on Neural Networks (IJCNN '06), 2006, pp. 2362–2369.

[25] Eskandari, M.—Hashemi, S.: A Graph Mining Approach for Detecting Unknown Malwares. Journal of Visual Languages and Computing, Vol. 23, 2012, No. 3, pp. 154–162, doi: 10.1016/j.jvlc.2012.02.002.

[26] Qiao, Y.—Yang, Y.—Ji, L.—He, J.: Analyzing Malware by Abstracting the Frequent Itemsets in API Call Sequences. 2013 12th IEEE International Conference

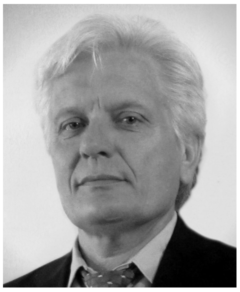
on Trust, Security and Privacy in Computing and Communications (TrustCom), 2013, pp. 265–270.

[27] Stanfill, C.—Waltz, D.: Toward Memory-Based Reasoning. Communications of the ACM, Vol. 29, 1986, No. 12, pp. 1213–1228, doi: 10.1145/7902.7906.

[28] Webb, A. R.—Copsey, K. D.: Statistical Pattern Recognition. Third Edition. Wiley, 2011.

[29] Wilson, D. R.—Martinez, T. R.: Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research, Vol. 6, 1997, No. 1, pp. 1–34.

Martin Jureček graduated from the Charles University in Prague, Faculty of Mathematics and Physics, with the specialization in mathematical methods of information security. He is now a Ph.D. student at the Faculty of Information Technology of the Czech Technical University in Prague. His main research interests focus on the application of machine learning and artificial intelligence approaches to malware detection. Another area of his interest is cryptography and information security.



Róbert Lórencz graduated from the Faculty of Electrical Engineering of the Czech Technical University in Prague in 1981. He received his Ph.D. degree in 1990 from the Institute of Measurement and Measuring Methods, Slovak Academy of Sciences in Bratislava. Currently he is Full Professor at the Faculty of Information Technology of the Czech Technical University in Prague. His research interests are cryptography and arithmetic units for cryptography primitives, various cryptoanalysis methods of block and stream ciphers. Another topic of his interest is alternative arithmetic for numerical computation.