Computing and Informatics, Vol. 38, 2019, 525–554, doi: 10.31577/cai_2019_3_525

INVESTIGATION OF CLOUD SCHEDULING ALGORITHMS FOR RESOURCE UTILIZATION USING CLOUDSIM

Altaf Hussain, Muhammad Aleem, Muhammad Azhar Iqbal
Muhammad Arshad Islam

Faculty of Computing
Capital University of Science and Technology
Islamabad, Pakistan
e-mail: aleem@cust.edu.pk

Abstract. Compute Cloud comprises a distributed set of High-Performance Computing (HPC) machines to stipulate on-demand computing services to remote users over the internet. Clouds are capable enough to provide an optimal solution to address the ever-increasing computation and storage demands of large scientific HPC applications. To attain good computing performances, mapping of Cloud jobs to the compute resources is a very crucial process. Currently we can say that several efficient Cloud scheduling heuristics are available, however, selecting an appropriate scheduler for the given environment (i.e., jobs and machines heterogeneity) and scheduling objectives (such as minimized makespan, higher throughput, increased resource utilization, load balanced mapping, etc.) is still a difficult task. In this paper, we consider ten important scheduling heuristics (i.e., opportunistic load balancing algorithm, proactive simulation-based scheduling and load balancing, proactive simulation-based scheduling and enhanced load balancing, minimum completion time, Min-Min, load balance improved Min-Min, Max-Min, resource-aware scheduling algorithm, task-aware scheduling algorithm, and Sufferage) to perform an extensive empirical study to insight the scheduling mechanisms and the attainment of the major scheduling objectives. This study assumes that the Cloud job pool consists of a collection of independent and compute-intensive tasks that are statically scheduled to minimize the total execution time of a workload. The experiments are performed using two synthetic and one benchmark GoCJ workloads on a renowned Cloud simulator CloudSim. This empirical study presents a detailed analysis and insights into the circumstances requiring a load balanced scheduling mechanism to improve overall execution performance in terms of makespan, throughput, and resource utilization. The outcomes have revealed that the Sufferage and task-aware scheduling algorithm produce minimum makespan for the Cloud jobs. However, these two scheduling heuristics are not efficient enough to exploit the full computing capabilities of Cloud virtual machines.

Keywords: Distributed computing, scheduling algorithm, high-performance computing, scheduling

Mathematics Subject Classification 2010: 68W15

1 INTRODUCTION

Cloud is a large pool of virtualized resources that are provisioned on-demand in a scalable manner. The efficient job scheduling mostly increases the user's satisfaction, improves the system utilization, reduces the job execution time, and minimizes the energy consumption. Scheduling heuristics are generally classified as static and dynamic. A static scheduling heuristic forms a complete job mapping plan before execution while a dynamic scheduling technique generally relies on the runtime parameters to schedule jobs in a best-effort with the use of resources in a more scalable manner as per user requirements. The static Cloud scheduling heuristics avoid the migration of Virtual Machines (VMs) to avoid communication overheads to reduce the execution time. In addition, most of the static techniques produce good turnaround time and irrefutable Quality of Service (QoS) because of the pre-ensured availability of the computing resources for the workload execution [1]. However, the static scheduling techniques may produce inefficient and lower resource utilization due to runtime changes in workload and computing environment [2].

In Cloud environment, scheduling is employed at two levels:

1. VM scheduling is concerned with the mapping of virtual machines to the physical hosts in a Cloud data-center, and
2. job scheduling is concerned with the assignment of jobs to the virtual machines.

Various metrics are harnessed to determine the performance of job scheduling algorithms. These performance metrics include makespan, throughput, resource utilization, response time, and energy consumption. A crucial aspect of scheduling is to map Cloud jobs in a load balanced manner to reduce the makespan of a job pool. Load balanced mapping refers to a distribution of jobs (among VMs) so that all the VMs accomplish the execution of assigned workload within the approximately same time duration. Importantly, a balanced load ensures improved resource utilization, higher throughput, and lower execution time for a job pool.

Several Cloud scheduling heuristics [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16] have been presented by the scientific community. However, the selection of an appropriate scheduler according to a given environment (jobs computing requirements and the available computing resources) to achieve desired scheduling objectives (such as reduced makespan, higher throughput, etc.) is still a difficult task. Since each heuristic contains different underlying assumptions; therefore, a precise comparison cannot be made. In this regard, we empirically scrutinize and experimentally compare ten state-of-the-art static scheduling heuristics (i.e., Opportunistic Load Balancing (OLB) [1, 10, 13], Proactive Simulation-Based Scheduling and Load Balancing (PSSLB) [17], Proactive Simulation-Based Scheduling and Enhanced Load Balancing (PSSELB) [17], Minimum Completion Time (MCT) [6, 18], Min-Min [7, 8, 9, 10, 11], Load Balance Improved Min-Min (LBIMM) [11, 17], Max-Min [5, 10, 13], Resource-Aware Scheduling Algorithm (RASA) [12, 19], Task-Aware Scheduling Algorithm (TASA) [16], and Sufferage [10, 14, 15]).

In an empirical analysis, Syed Hamid Hussain Madni et al. provides the investigation of First Come First Serve (FCFS), Minimum Execution Time (MET), MCT, Min-Min, Max-Min, and Sufferage scheduling algorithms [3]. Based on their analysis, Madni et al. concluded that the hybridization of these techniques may result in more improved results and overcome the limitations of each other to achieve the optimization of task scheduling in cloud computing [3]. Therefore, in this research work, some hybridized scheduling techniques (i.e. RASA [12, 19], TASA [16], LBIMM [11, 17], PSSLB [17], and PSSELB [17] algorithms) are also considered for performance investigation of resource utilization in cloud computing. Figure 1 shows ten scheduling algorithms; where the techniques on the tail of each arrow are the modified and hybridized techniques based on the scheduling techniques directed by the arrow symbols (i.e., the mechanism of RASA is based on Max-Min and Min-Min, the mechanism of TASA is based on Sufferage and Min-Min, LBIMM is the modified version of Min-Min, PSSLB is the modified version of Max-Min, and PSSELB is the modified version of PSSLB technique).

In this study, we consider the following assumptions for the empirical-based comparison of the employed scheduling heuristics. One such assumption is that a workload is referred to as a collection of independent and compute-intensive tasks (without inter-task data dependencies). The mapping of these tasks is performed statically to minimize the scheduling overhead and to evade job migrations [13]. This empirical study provides a detailed analysis of the scheduling heuristics and insights of the scheduling mechanisms where a higher throughput and reduced execution time is attained; however, a considerable load imbalance is observed in the experimentation. We argue that the existing state-of-the-art static scheduling heuristics should address the load-balancing issue to attain exquisite resource utilization in Cloud computing. A near-optimal resource utilization will produce higher throughput, reduced execution time (for the Cloud job pool), and energy efficient execution.

In summary, we present an analysis of the resource utilization of virtual resources in terms of workload distribution among all the VMs. The empirical investigation reveals that most of the scheduling heuristics are not efficient enough to exploit the full computing capabilities of virtual machines in Cloud infrastructure. This empirical study has highlighted various pressing research gaps that must be overcome

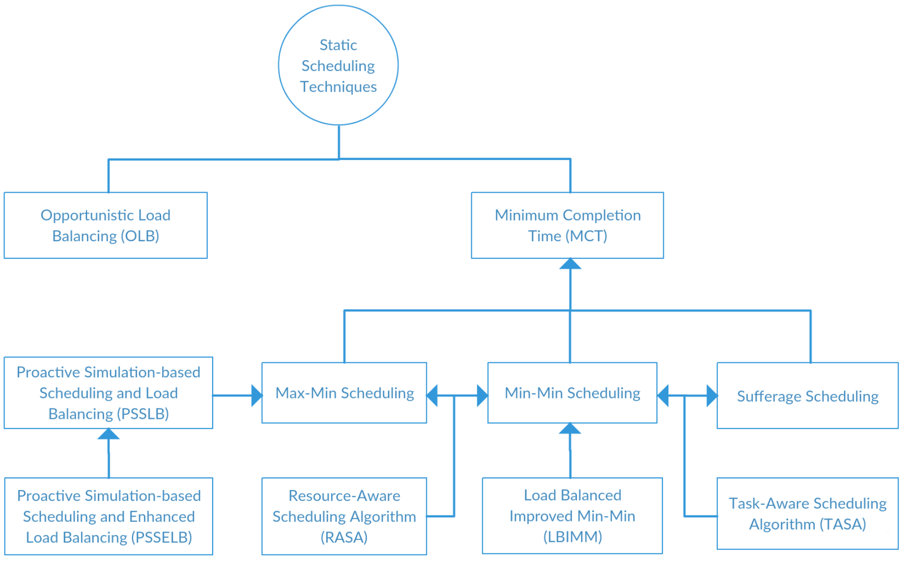

Static Scheduling Techniques
Opportunistic Load Balancing (OLB)
Minimum Completion Time (MCT)
Proactive Simulation-based Scheduling and Load Balancing (PSSLB)
Max-Min Scheduling
Min-Min Scheduling
Sufferage Scheduling
Proactive Simulation-based Scheduling and Enhanced Load Balancing (PSSELB)
Resource-Aware Scheduling Algorithm (RASA)
Load Balanced Improved Min-Min (LBIMM)
Task-Aware Scheduling Algorithm (TASA)

Figure 1. Hybridization of Cloud scheduling algorithms

to improve the scheduling performance and to reduce cost at Cloud service provider level (discussed in Section 5). Major contributions of this work include:

1. a critical analysis and synthesis of the existing state-of-the-art static Cloud scheduling algorithms to identify the pros and cons of each algorithm;
2. in-depth performance analysis (in terms of turnaround time, resource utilization, and throughput) and empirical assessment of the existing static scheduling algorithms using two synthetic datasets and one benchmark dataset for Cloud and distributed computing [37];
3. identification of the potential research directions that could assist the scientific community to cope with the challenges pertaining to the static scheduling heuristics for Cloud computing.

The rest of the paper is organized as follows. Section 2 discusses the working semantics of scheduling heuristics. Section 3 illustrates the experimental setup and workload compositions. Section 4 examines the experimental results. The resource utilization and load imbalance in workload distribution caused by scheduling heuristics are presented in Section 5 and the potential research directions are identified. Section 6 concludes this research work.

2 CLOUD SCHEDULING HEURISTICS

In this study, we synthesize and evaluate ten prominent static Cloud scheduling heuristics which are: MCT [3, 6], OLB [1, 10, 13], Min-Min [7, 8, 9, 10, 11], Max-Min [5, 10, 13], Sufferage [10, 14, 15], RASA [12, 20], PSSLB [17], PSSELB [17], LBIMM [11, 17], and TASA [16]. The working of these heuristics is delineated below.

2.1 OLB Algorithm [1, 4, 10, 13]

This scheduling technique assigns each job, in an arbitrary order, to the next available machine regardless of considering the job's execution time on that particular machine. OBL is a simple scheduling scheme having low scheduling overhead and complexity. A major scheduling objective of the OBL scheme is to make all the Cloud machines as busy as possible [13]. However, OBL scheduling heuristic mostly results in poor makespan because it is not resource-aware.

2.2 MCT Algorithm [3, 8, 13]

MCT technique assigns a candidate job to a machine that consumes minimum time for the job [13]. The MCT heuristic examines the current load of machines to find a suitable target machine for job assignment [8]. At each scheduling step, MCT heuristic has to scan all the available machines to find the most appropriate computing resource (i.e., machine producing the minimum completion time for a job). The expensive search mechanism employed by the MCT (at each scheduling step) causes a significant scheduling overhead.

2.3 Min-Min Algorithm [11, 13, 21, 22]

Min-Min scheduling first determines the minimum completion time of all the unallocated jobs and proceeds with the assignment of a job having overall minimum completion time on a certain machine. Both Min-Min and MCT scheduling heuristics rely on the completion time of a job on a certain machine [6]. MCT considers the current job only for scheduling decision (at a certain scheduling step), whereas the Min-Min considers the minimum completion time for all the unallocated jobs (in each scheduling decision). Min-Min scheduling heuristic favors (i.e., schedules first) the small-sized jobs while penalizing (causing delayed execution) for larger jobs [8, 9, 10, 11, 12]. Therefore, Min-Min mostly overloads the faster machines with larger number of small-sized jobs, while the slower machines are assigned fewer but larger jobs. Thus, the larger jobs mapped on slower resources often cause a higher makespan for the execution of the job-pool [23].

2.4 Max-Min Algorithm [3, 21, 24]

Max-Min scheduling first computes the expected minimum completion time for all the jobs and then the job requiring a maximum completion time is assigned to the concerned machine. The scheduling process is repeated until all the un-allocated Cloud jobs are scheduled. To avoid longer response time (for the larger jobs), Max-Min scheduling heuristic selects larger jobs to be executed early [6, 14]. Max-Min heuristic mostly performs better in the scenario when there is a large number of small-sized jobs with a few larger size jobs [11, 14, 15]. The inherent mechanism of both Max-Min and Min-Min heuristics adversely affects the resource utilization in Cloud (as evident in our experimental results presented in Section 4).

2.5 Sufferage Algorithm [11, 21, 22]

Sufferage scheduling calculates the sufferage value for each job. To calculate the sufferage value (i.e., a penalty in terms of longer execution time), the minimum completion time and the second best minimum completion time producing VMs are determined for each job (in each scheduling iteration). Afterward, the job experiencing the highest sufferage value is assigned to the machine (producing minimum completion time for that job). Sufferage heuristic produces good results often with reduced makespan; however, this scheduling mechanism causes higher scheduling overhead (due to the calculation of sufferage value for each job in each scheduling iteration) as compared to OLB, MCT, Max-Min, and Min-Min [18, 21, 22].

2.6 RASA Algorithm [12, 19]

RASA technique contemplates both Min-Min and Max-Min heuristics in the alternate scheduling decisions until all the jobs are scheduled. RASA exploits the merits of both Min-Min and Max-Min to evade corresponding limitations of these two scheduling algorithms in certain cases (as discussed above). Mostly, RASA results in a lower makespan when it considers smaller and larger jobs in alternate scheduling steps [12]. However, RASA penalizes smaller size jobs (causing delayed execution) when the number of larger jobs is higher in the workload [24].

2.7 TASA Algorithm [16, 30]

TASA favors the smaller jobs in the first scheduling step (based on Min-Min heuristic) and finds an appropriate machine for the job (using the Sufferage heuristic) in the second scheduling step [16]. In most of the cases, TASA produces better makespan as compared to other scheduling heuristics such as Min-Min, Max-Min, and OLB [16].

2.8 LBIMM Algorithm [11, 17]

LBIMM [11, 17] assigns jobs to machines using the Min-Min scheduling technique in the first phase. In the subsequent phase, LBIMM finds the smallest job on the most loaded machines and determines its completion time on the other machines. After that, the minimum completion time of that job is compared with makespan. If this time value is less than the makespan, the job is assigned to a new machine and the ready time of both machines are modified. This procedure is repeated for the next smallest job on the most loaded machine, too. The process is repeated until there is no other machine that can produce the minimum completion time for the smallest job on the heavily loaded machine than makespan on another machine. This technique shares a load of heavy machines with the idle or lighter machines. LBIMM produces better makespan and load balancing than Min-Min heuristic.

2.9 PSSLB Algorithm [17]

PSSLB and PSSELB Algorithms are proposed to assign the large-sized jobs to the machines that can execute them faster than the other machines. PSSLB finds the matrix (i.e., each row has a completion time of a specified job on all machines) of completion time of each job on each machine. The matrix is sorted in a way that the last column stores minimum completion time for each job. Therefore, the longest job on the last column is selected and assigned to machine producing minimum completion time for it.

2.10 PSSELB Algorithm [17]

PSSELB Algorithm is the modified version of PSSLB that produces a load balanced schedule. The largest job among the unallocated jobs is assigned to the machine using PSSLB, and completion time of this job is considered as a pivot. After that, the jobs that produce completion time (i.e., on other machines) equal to or less than the pivot are iteratively determined and assigned to the concerned machines (i.e., producing MCT for a job equal to or less than the pivot value). Next, the largest job is assigned to the concerned machine using PSSLB, and the pivot is updated with the completion time of the largest job. Again, the jobs with MCT on other machines (i.e., except the machine with last largest job assigned) that is equal to or less than the pivot are determined and assigned to the concerned machine. This scheduling procedure is repeated till all the unallocated jobs are assigned to machines in the same way.

Assuming N number of Cloud jobs to be scheduled on M machines, Table 1 presents the summary of strengths, weaknesses, and time-complexity of the eight scheduling heuristics, and the employed simulation tool (by the authors of the mentioned research work).

Heuristics	**Strengths**	**Weaknesses**	**Complexity**	**Tools Used**
OLB [5, 10, 13]	Low complexity, Minimal overhead, keeps machines busy [5, 10]	Load-imbalance, and No-fairness in scheduling [4]	$O(M)$	tGSF Simulator [18], CloudSim Simulator [4].
MCT [3, 8, 13]	Improved makespan than OLB [9], Machine-aware scheduling [24]	Load-imbalance [5, 15], Overloads faster VMs	$O(M \cdot N)$	CloudSim [4], NS Simulator [7]
MinMin [3, 24, 26]	Favors smaller jobs [12], Reduced makespan for smaller jobs [1, 8]	Overloads faster VMs with smaller Jobs [12], Penalizes larger jobs.	$O(M \cdot N^2)$	C-Language [24], Matlab [15], Java-based Simulation
LBIMM [11, 17, 27]	Improved makespan than Min-Min [27], improved resource utilization than Min-Min [11]	A few smaller jobs are penalized while rescheduled [11]	$O(M \cdot N^2)$	Matlab [11]
MaxMin [3, 21, 24]	Favors larger jobs [1, 28], Reduced makespan for larger jobs [22].	Penalizes smaller jobs [16], Load-imbalance for job pool with more larger jobs [12].	$O(M \cdot N^2)$	Matlab and Java-based Simulation [8]
RASA [12, 19, 20]	Fair treatment of larger and smaller jobs [14].	Penalizes smaller jobs in dataset with more larger jobs [23]	$O(M \cdot N^2)$	GridSim [11].
Sufferage [3, 9, 14]	Improved makespan than MCT, Min-Min, and Max-Min [21], Job allocation to appropriate VM [22].	High scheduling overhead due to Sufferage value calculation [10].	$O(M \cdot N^2)$	C++ and Java implementations [14], Matlab
TASA [16, 30]	Improved makespan than Max-Min, Min-Min and RASA [16], Favors smaller jobs.	Load balancing is not considered [16].	$O(M \cdot N^2)$	CloudSim [16].
PSSLB [17]	Reduces completion and response time for larger jobs [17]	Penalizes smaller jobs [17]	$O(M \cdot N^2)$	CloudSim [17].
PSSELB [17]	Improved makespan than PSSLB [17]	Results in load imbalance compared to PSSLB [17]	$O(N^2/2)$	CloudSim [17].

Table 1. Summary of scheduling algorithms in related work

3 EXPERIMENTAL SETUP

Evaluation of scheduling and resource allocation policies on real Cloud (with a varying load and system size) is a challenging problem. The use of real testbeds restricts the experiments to the scale of the test environment. Cloud computing model is based on a pay-per-use model, thus repeatable experiments on real Cloud may incur a high monetary cost. Therefore, an ideal alternative to evaluate resource management related Cloud policies is to use a simulation environment that enables Cloud developers to conduct experiments by employing the desired and varying configurations related to computing infrastructure and dataset (i.e., Cloud jobs). In this work, we use a renowned Cloud simulator called CloudSim [31] (version 3.0.2). A user job is represented as cloudlet in CloudSim and the job's size (computational requirement) is measured in terms of Million Instructions (MI). We perform the simulation-based experiments on a machine equipped with Intel Core i3-4030U Quad-core processor (having 1.9 GHz clock speed) and 4 GB of main memory. Liu and Cho [32] characterize the computing machines and workloads on a Google cluster and found that 93 % of the machines are fairly homogeneous on Google cluster with approximately 6 % of the machines with a greater computing capability [32]. Using the characteristics of the real computing machines (found in Liu and Cho's study [32]) we build an experimental setup for empirical evaluation. Table 2 illustrates the configuration details of the employed simulation environment. Figure 2 presents the overall statistics of the employed VMs with computing powers in terms of Million Instructions Per Seconds (MIPS).

Parameters	**Details**
Power of Cloud Host Machines	4 Dual core (4 000 MIPS), 26 Quad core (4 000 MIPS)
Total Host Machines	30 Host Machines
Total VMs	50 Virtual Machines
Total Cloudlets	100, 150, 200, 250, 300, 350, 400, 450, 500, 550, 600, 650, 700, 750, 800, 850, 900, 950, 1 000

Table 2. Configuration of simulation environment

3.1 Workload Generation

In this research work, one GoCJ benchmark dataset [37] and two synthetic datasets are used for the performance assessment of scheduling algorithms in simulation-based experimentation. The detail of these datasets are described as follows.

3.1.1 GoCJ Dataset [37]

The data confidentiality and other such policies maintained by the Cloud service providers [33] hinder to acquire real Cloud workload for the empirical investigations. The contemporary state-of-the-art has been scrutinized to explore a real workload

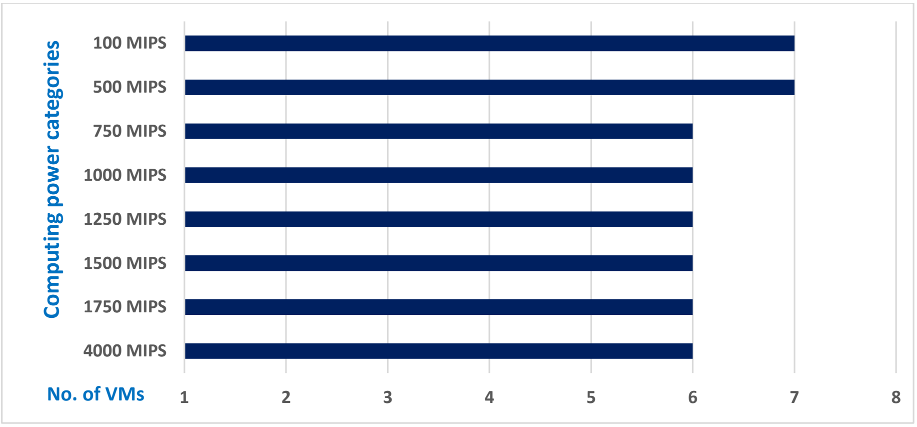


Figure 2. VMs in the cloud datacenter

behavior in Google cluster traces [32, 33, 34, 35, 36] and MapReduce logs from the M45 supercomputing cluster [36].

Liu and Cho studied a large-scale Google cluster usage traces of 29 days to examine the machine properties, workload behavior, and resource utilization [32]. The analysis of the Google cluster traces affirms that the majority of jobs execute for fairly a short duration (i.e., less than 15 minutes), while the low number of jobs execute over 300 minutes [32]. Further, the study [32] establishes the fact that approximately two thirds of the jobs in the Google cluster traces execute for less than five minutes and approximately 20 % of the jobs execute for less than one minute. The median length of a job in the Google cluster traces is approximately 3 minutes. Another similar study of Google cluster is presented by Chen et al. [35] and Reiss et al. [34].

In addition, Kavulya et al. [36] have scrutinized MapReduce logs of the M45 supercomputing cluster (logs of 10 months released by Yahoo). The study of MapReduce logs affirms that 95 % of the jobs complete the execution within 20 minutes and approximately 4 % of the jobs exceed execution up to 30 minutes [36]. Literature review [32, 33, 34, 35, 36] reveals that most of the Cloud jobs are of a short size and execute for less than 5 minutes.

Based on the analysis, we have generated a benchmark workload entitled Google Cloud Jobs (GoCJ) [37]. Considering the computing power of VMs in a Cloud datacenter (see Figure 1), the cloudlet completion time follows a long-tailed distribution (with 90 % of cloudlets in GoCJ workload completing their execution within 1.6 minutes). The longest executing cloudlet observed in the GoCJ workload lasts up to 15 minutes (6 % cloudlets execute for less than 5 minutes and 4 % execute for 15 minutes). The average size of a job in GoCJ workload is 5 minutes. Figure 3 presents the ratios and sizes of cloudlets distribution in GoCJ workload in terms of percentage and MIs, respectively.

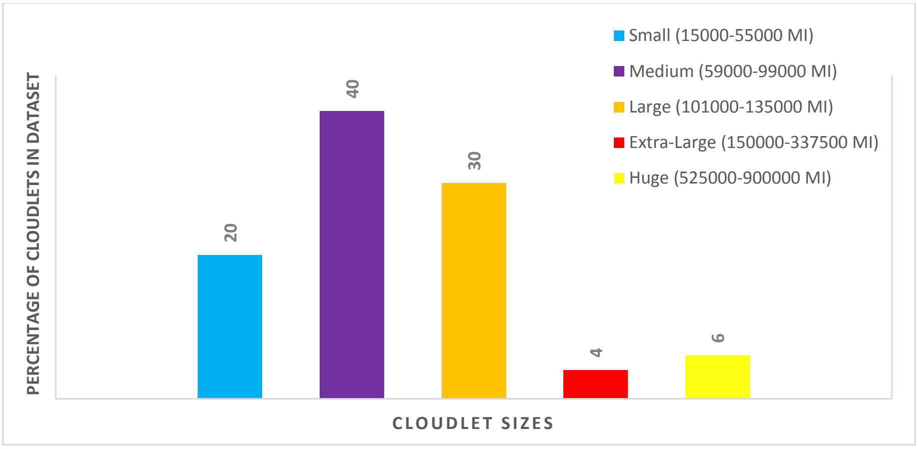


Figure 3. Composition of benchmark GoCJ dataset [37]

3.1.2 Synthetic Datasets

In addition to GoCJ dataset, we study the literature [28, 38, 39] to generate two synthetic datasets containing fix-sized jobs. Mehdi et al. [28] conducted the experimentation of a genetic scheduler with heterogeneous VMs (1.0 GHz to 4.0 GHz speed) to execute up to 100 cloudlets. In another work, Mehdi et al. [38] have examined the Cloud scheduling using 100 VMs (computing power of 1.0 GHz, 2.0 GHz, 2.5 GHz, and 3.0 GHz) to execute up to 500 cloudlets. Behzad et al. [39] presented a comparative analysis of different scheduling algorithms by using 7 000 and 15 000 jobs with a varying number of CPUs (i.e., 4 to 64 processors).

The synthetic-I workload is created with five fixed-size cloudlets (see Figure 4). The majority of cloudlets (i.e., 75 % of the cloudlets in synthetic-I workload) complete execution within 2 seconds and a small tail of the cloudlet distribution executes up to 45 seconds (i.e., 15 % of the cloudlets run for 15 seconds and 5 % of the cloudlets run for 45 seconds). The fixed sizes of tiny, small, medium, large and extra-large cloudlets in synthetic-I dataset are 200, 1 000, 5 000, 15 000, and 45 000 MIs, respectively.

The synthetic-II workload is generated using a random number generation mechanism by employing five cloudlet-size ranges (see Figure 4). The majority of cloudlets (i.e., 85 % in synthetic-II workload) are of a short size and a small tail of the cloudlet distribution completes execution within 45 seconds (i.e., 10 % of the cloudlets run for 10 seconds and 5 % of the cloudlets run for 45 seconds). The cloudlet-size ranges of tiny, small, medium, large, and extra-large cloudlets are 1–200, 800–1 200, 1 800–2 500, 7 000–10 000, and 30 000–45 000 MIs, respectively.

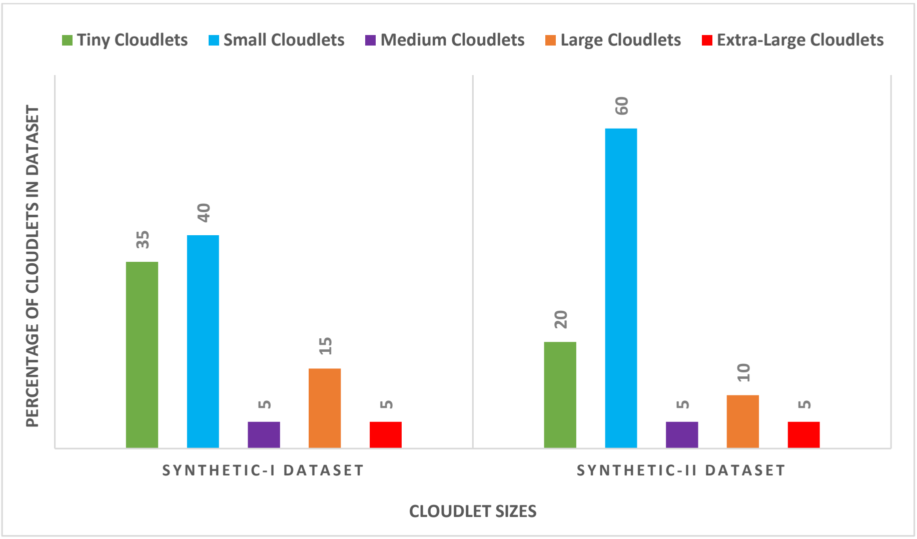


Figure 4. Composition of synthetic workloads

4 EXPERIMENTAL RESULTS

Three performance metrics are measured and presented for experimental evaluation, i.e., makespan, *Average Resource Utilization Ratio* (ARUR), and throughput.

4.1 Makespan-Based Results

We use term makespan to represent the completion of all the cloudlets execution in a workload. The smaller value of makespan represents a better execution performance. The makespan is mathematically expressed as follows:

$$Makespan = \max_{\forall j=1,2,3,\ldots,m} (VM\_CT_j) \tag{1}$$

where m represents the total number of VMs (which is 50 in our experiments) and VM_CT_j is the completion time of VM_j by executing its assigned cloudlets. VM_CT_j is computed as:

$$VM\_CT_j = \sum_{i=1}^{n_j} \frac{Cloudlet_i.MI}{VM_j.MIPS} \tag{2}$$

where $Cloudlet_i.MI$ represents the size of $cloudlet_i$ in terms of *Million Instructions* (MIs), $VM_j.MIPS$ is the computing power of VM_j in terms of *Million Instructions*

Per Second (MIPS) and n_j represents the total number of cloudlets assigned to VM_j.

Figure 5 shows the makespan results of 10 scheduling algorithms for Synthetic-I, Synthetic-II, and GoCJ benchmark workloads. For more clarity, the average makespan (i.e., separately for synthetic-I, synthetic-II, and GoCJ workloads) of all the experiments using different number of cloudlets is calculated as follows:

$$Avg\_Makespan = \frac{\sum_{i=1}^{NE} Makespan_i}{NE} \tag{3}$$

where NE represents the number of experiments performed for each scheduling algorithm (i.e., using specified workload) and $Makespan_i$ represents the makespan of i^{th} experiment. Each experiment is repeated using a varying number of cloudlets (i.e., cloudlets 100–1 000, as presented in Table 2). The average makespan results for all scheduling algorithms are presented in Figure 6. The LBIMM, TASA, and Sufferage techniques produce the shortest makespan for Synthetic-I, Synthetic-II, and GoCJ workload, respectively. However, there is a minor difference with respect to makespan of LBIMM, TASA and sufferage algorithms for the three workloads. On the other hand, OLB achieves the largest makespan for Synthetic-I, Synthetic-II, and GoCJ benchmark workloads.

4.2 Throughput-Based Results

Throughput is the number of jobs executed during the span of per unit time. In our experiments, the throughput is referred as the number of cloudlets executed per second. Throughput can be calculated as follows:

$$Throughput = \frac{n}{Makespan} \tag{4}$$

where n is the number of employed cloudlets. A scheduling technique producing higher throughput value is assumed a better performing algorithm. For more clarity in results, the average throughput for synthetic-I, synthetic-II, and GoCJ workloads is calculted as:

$$Avg\_Throughput = \frac{\sum_{i=1}^{NE} Throughput_i}{NE}. \tag{5}$$

Figure 7 presents the throughput results for execution of Synthetic-I, Synthetic-II, and GoCJ benchmark workloads. While, for more clarity in the simulation results, Figure 8 represents the average throughput for all scheduling algorithms using the given workloads. Likewise average makespan results, the LBIMM, TASA, and Sufferage algorithms achieve the highest throughput for Synthetic-I, Synthetic-II, and GoCJ benchmark workloads, respectively. Similarly, OLB technique achieves the least throughput using given three workloads.

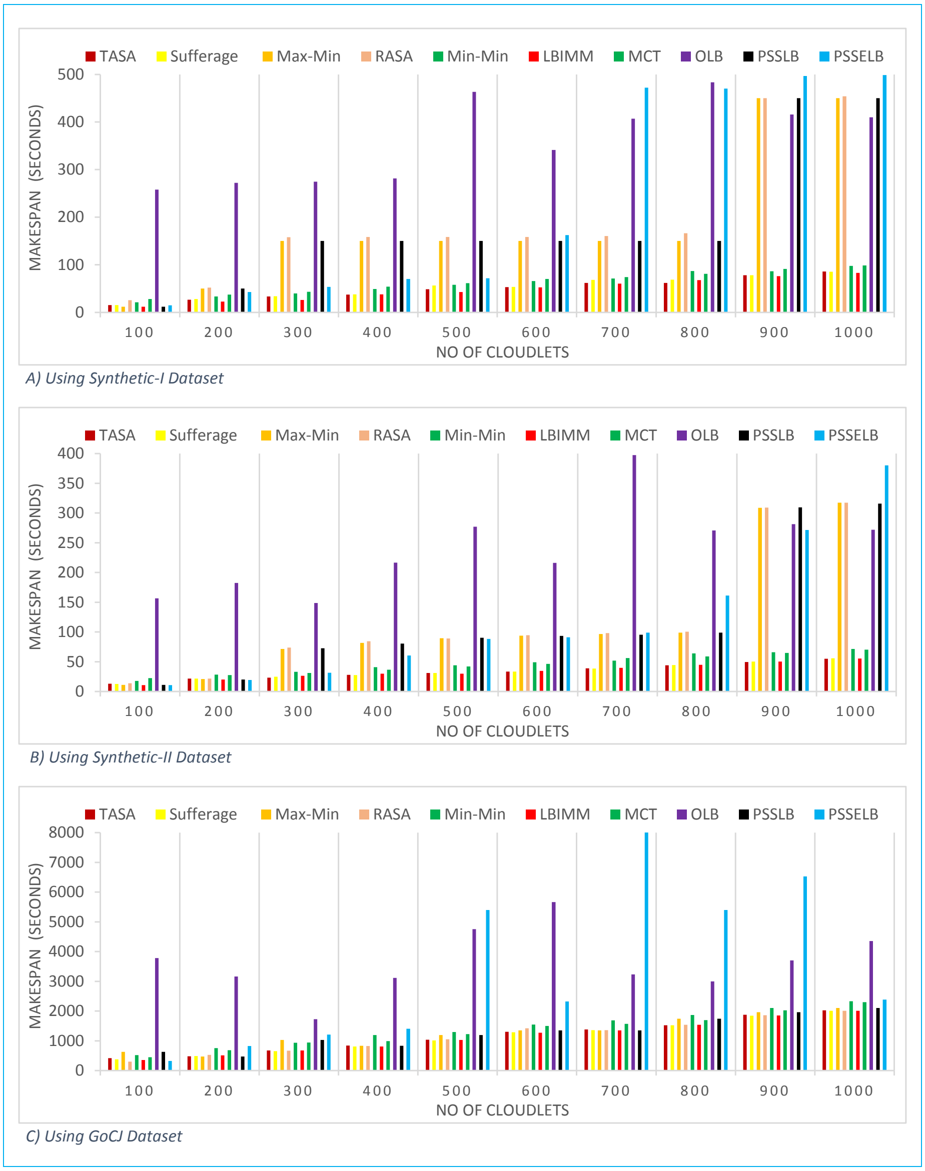


Figure 5. Makespan results

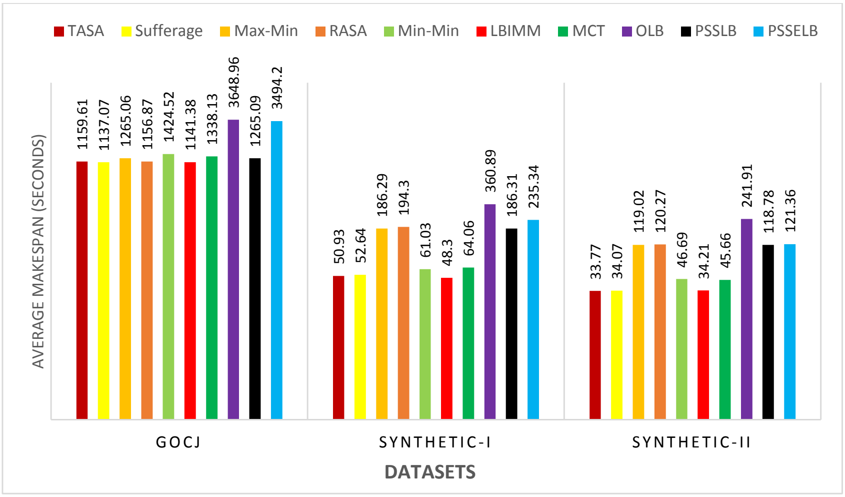


Figure 6. Average makespan results

4.3 ARUR-Based Results

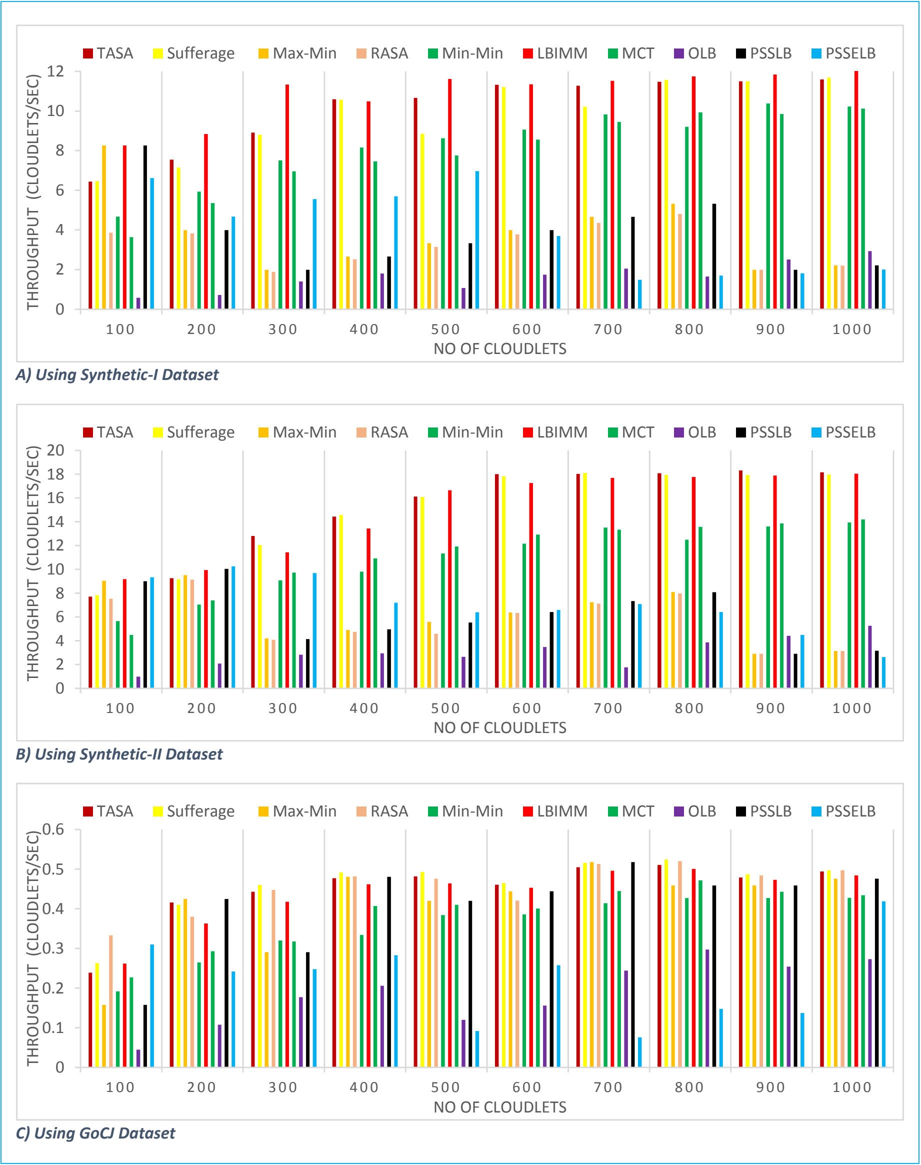
ARUR shows the average resource utilization ratio for a compute Cloud. ARUR is the ratio of average makespan to the makespan of the Cloud system and is calculated as follows [10].

$$ARUR = \frac{\frac{\sum_{j=1}^{m} VM\_CT_j}{m}}{Makespan}. \quad (6)$$

ARUR value remains between 0 and 1, where value close to 1 shows exceptional resource utilization (i.e., nearest to 100 % resource utilization). Figure 9 shows the ARUR-based experimental results of ten scheduling algorithms for Synthetic-I, Synthetic-II, and GoCJ benchmark workloads. Mean ARUR value for each heuristic is reported based on the following equation:

$$Mean\_ARUR = \frac{\sum_{i=1}^{NE} ARUR_i}{NE}. \quad (7)$$

Figure 10 presents the Mean ARUR results for the execution of Synthetic-I, Synthetic-II, and GoCJ benchmark workloads. The LBIMM technique attains the highest ARUR (76.5 % resource utilization), as compared to other scheduling algorithms for Synthetic-I workload. However, in case of Synthetic-II and GoCJ benchmark workloads, Sufferage algorithm produces the highest resource utilization (75.7 % and 86.3 % resource utilization, respectively), as compared to other scheduling techniques. The OLB scheduling produces the least resource utilization among all scheduling techniques using given three workloads (i.e., 18.8 %, 20.7 %,



A) Using Synthetic-I Dataset

B) Using Synthetic-II Dataset

C) Using GoCJ Dataset

Figure 7. Throughput results

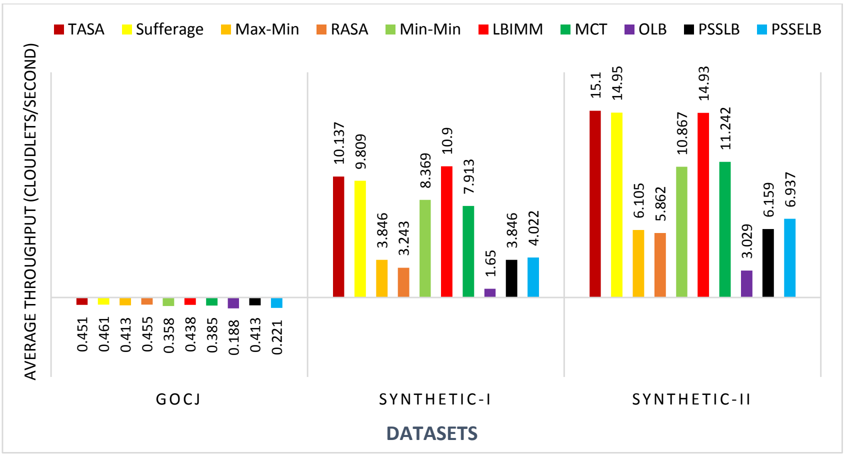


Figure 8. Average throughput results

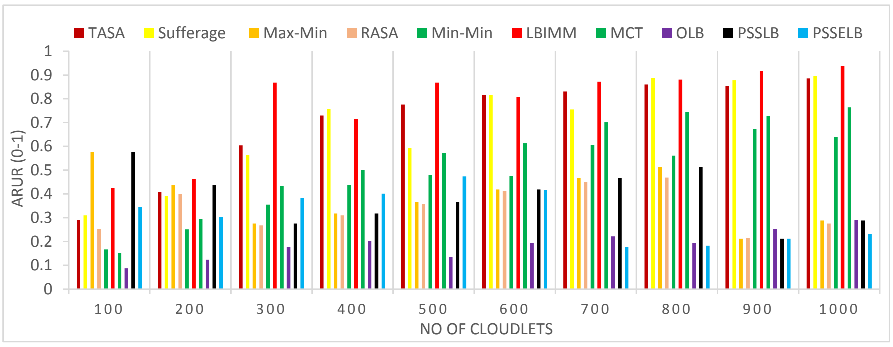
and 41.2 % resource utilization for Synthetic-I, Sythetic-II, and GoCJ workloads, respectively).

4.4 Comparative Discussion

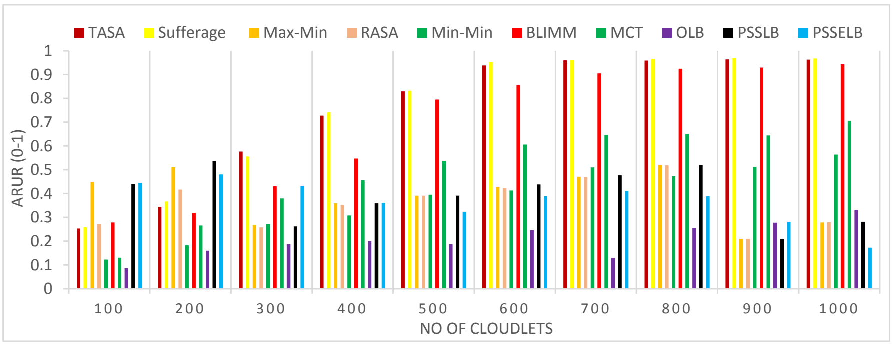
OLB produces poor makespan and very low resource utilization. However, OLB technique requires simple implementation, causes minimal scheduling overhead, and results in lower time complexity. MCT provides improved makespan for the workload execution and minimal completion time for each job. However, MCT results in low resource utilization for both skewed and non-skewed workloads because it overloads the faster machines, what results in an imbalanced distribution of workload.

A workload is referred to as positively skewed if it contains a large number of shorter size jobs with a few very long jobs [25]. On the other hand, if the workload comprises a large number of longer jobs with a few shorter jobs then the workload is referred to as negatively skewed [25]. The skewness in synthetic and GoCJ benchmark workloads used in this study is shown in Figures 3 and 4.

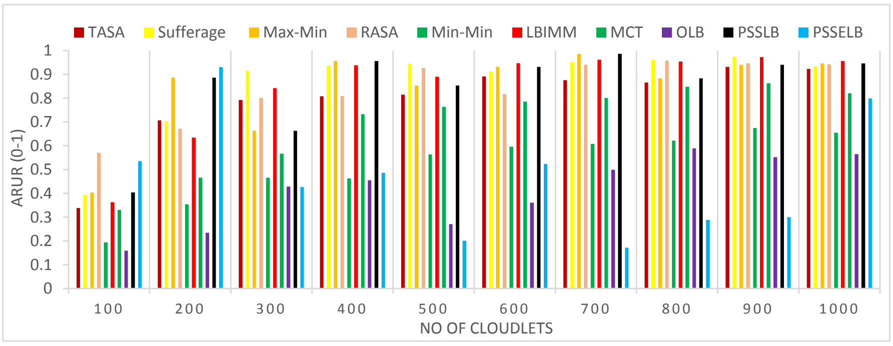
Min-Min and Max-Min do not produce good results (in terms of execution time) for a skewed workload [25]. Min-Min scheduling attains improved execution time when the workload has shorter size jobs or cloudlets. On the other hand, Min-Min produces longer makespan for a positively skewed workload (due to the inherent penalty for larger jobs). In case of workload containing most of the shorter jobs with few longer jobs, Max-Min achieves improved makespan by executing longer jobs on faster machines; and concurrently executing the shorter jobs on comparatively slower machines. However, Max-Min and Min-Min perform worst for a skewed workload and provide improved results for a non-skewed workload [25].



A) Using Synthetic-I Dataset



B) Using Synthetic-II Dataset



C) Using GoCJ Dataset

Figure 9. ARUR results

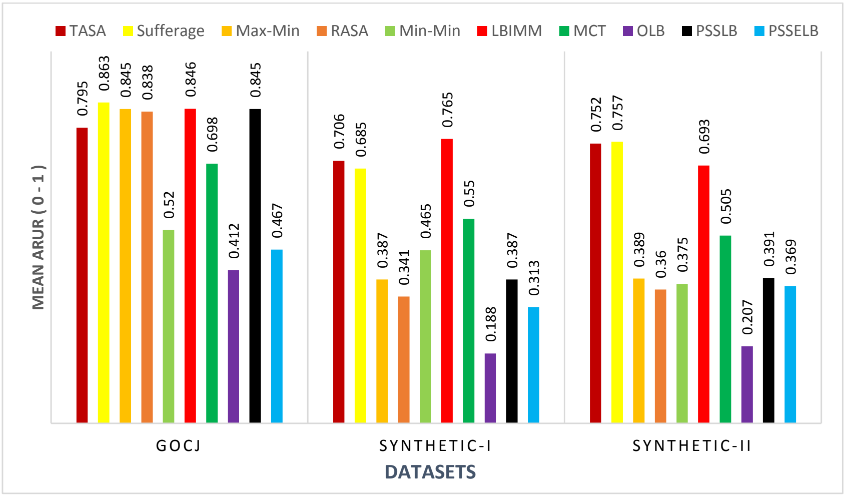


Figure 10. Mean ARUR results

RASA scheduling mechanism benefits from the merits of Min-Min and Max-Min producing a lower response time for both smaller and larger size jobs [12]. TASA, Sufferage, and LBIMM are modified versions of Min-Min [12, 16]. Therefore, these scheduling techniques provide better results (for a non-skewed workload) as compared to Min-Min. Additionally, TASA produces higher resource utilization (as evident by the results shown in Section 4.3). Similarly, PSSLB and PSSELB algorithms are the modified versions of Max-Min techniques (as shown in Figure 1). PSSELB provides better resource utilization as compared to Max-Min, while it introduces a slight degradation in makespan and throughput of PSSELB, as compared to Max-Min. On the other hand, PSSLB shows resemblance in results (i.e., makespan, resource utilization and throughput results), as compared to Max-Min technique.

5 RESOURCE UTILIZATION AND LOAD-IMBALANCE

We scrutinize the literature [1, 3, 4, 11, 12, 13, 16, 17] to examine the scheduling aspects related to load balancing and resource utilization for Cloud computing platform. The detailed analysis of the literature revealed that most of the existing scheduling algorithms [1, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 16, 17, 21, 22, 23, 24, 25, 26, 28, 29] are more inclined towards the decrease in turnaround and response time of a Cloud workload. For example, MCT, Min-Min, Max-Min, RASA, TASA, and Sufferage algorithms are designed and proposed to minimize makespan of the Cloud workload. The OLB, LBIMM, PSSLB, and PSSELB techniques consider additional consideration of load balancing, too. However, most of these scheduling algorithms are still unable to fully utilize the computing resources and result in an imbalanced work distribution among the virtual machines.

Our empirical analysis reveals that LBIMM, TASA, and Sufferage techniques produce comparatively better utilization of computing resources (i.e., higher resource utilization), as compared to other scheduling algorithms presented in this work. In Table 3, the results show that the LBIMM heuristic attains higher resource utilization (i.e., 8.36 % and 11.52 % higher, respectively), as compared to the TASA and Sufferage for the execution of Synthetic-I workload. The Sufferage heuristic attained higher resource utilization (i.e., 0.66 % and 9.24 % higher, respectively), as compared to the TASA and LBIMM for the execution of Synthetic-II workload. Similarly, the Sufferage scheduling achieves higher resource utilization (i.e., 2.01 % and 8.55 % higher resource utilization, respectively), as compared to the LBIMM and TASA for the GoCJ workload. Moreover, the Sufferage, Max-Min, and PSSLB heuristics attain higher resource utilization too for the execution of a GoCJ workload. The Max-Min and RASA have achieved higher resource utilization of 5.40 % and 6.29 %, respectively, as compared to TASA for GoCJ workload. This minor improvement in resource utilization by RASA and Max-Min techniques over TASA is due to the lower resource utilization incurred by the Min-Min heuristic (as compared to Max-Min and RASA for the execution of GoCJ workload). Further, it is observed that the LBIMM, TASA and Sufferage schedulers achieve the minimal completion time and better resource utilization as compared to Max-Min, RASA, Min-Min, MCT, OLB, PSSLB, and PSSELB techniques. The experimental results reveal that the LBIMM, TASA and Sufferage mechanisms attain on average lower makespan compared to the other state-of-the-art. However, the most of these mechanisms still lack a higher resource utilization (see Table 3) that could be improved to further lessen the makespan for the execution of a Cloud workload.

Algorithms	**GoCJ Dataset**	**Synthetic-I Dataset**	**Synthetic-II Dataset**
TASA	79.5 %	70.6 %	75.2 %
Sufferage	86.3 %	68.5 %	75.7 %
Max-Min	84.5 %	38.7 %	38.9 %
RASA	83.8 %	34.1 %	36.0 %
Min-Min	52.0 %	46.5 %	37.5 %
LBIMM	84.6 %	76.5 %	69.3 %
MCT	69.8 %	55.0 %	50.5 %
PSSLB	84.5 %	38.7 %	39.1 %
PSSELB	46.7 %	31.3 %	36.9 %
OLB	41.2 %	18.8 %	20.7 %

Table 3. Percentage of resource utilization of scheduling algorithms

Improving resource utilization is very crucial to reduce the cost and energy consumption for workload execution in a Cloud datacenter [23, 40]. Therefore, the issue concerning low resource utilization should be addressed in a comprehensive manner, while designing a Cloud scheduling technique. For the optimal resource utilization, the workload should be assigned to the computing resources according to the computing capabilities and application need. A resource- and application-aware Cloud scheduling algorithm with load balancing will greatly benefit in terms of reduced execution time and cost. Therefore, we have investigated machine-level load balancing (i.e., VM category and VM-level load distribution) by using these ten scheduling algorithms.

5.1 Discussion on VM Category and VM-wise Load Imbalance

For the empirical investigation, we employ the simulation environment based on 50 VMs (8 different sizes, as shown in Figure 2). For a balanced workload distribution, the cloudlets must be submitted to a compute Cloud for execution by considering the computing capabilities of the employed VMs. Moreover, it is critical to consider the current load of a VM, too. The computing load or share of each VM_j is presented as $Share_j$ and can be calculated as:

$$Share_j = \sum_{i=1}^{n} Cloudlet_i.MI \times \frac{VM_j.MIPS}{\sum_{k=1}^{m} VM_k.MIPS} \tag{8}$$

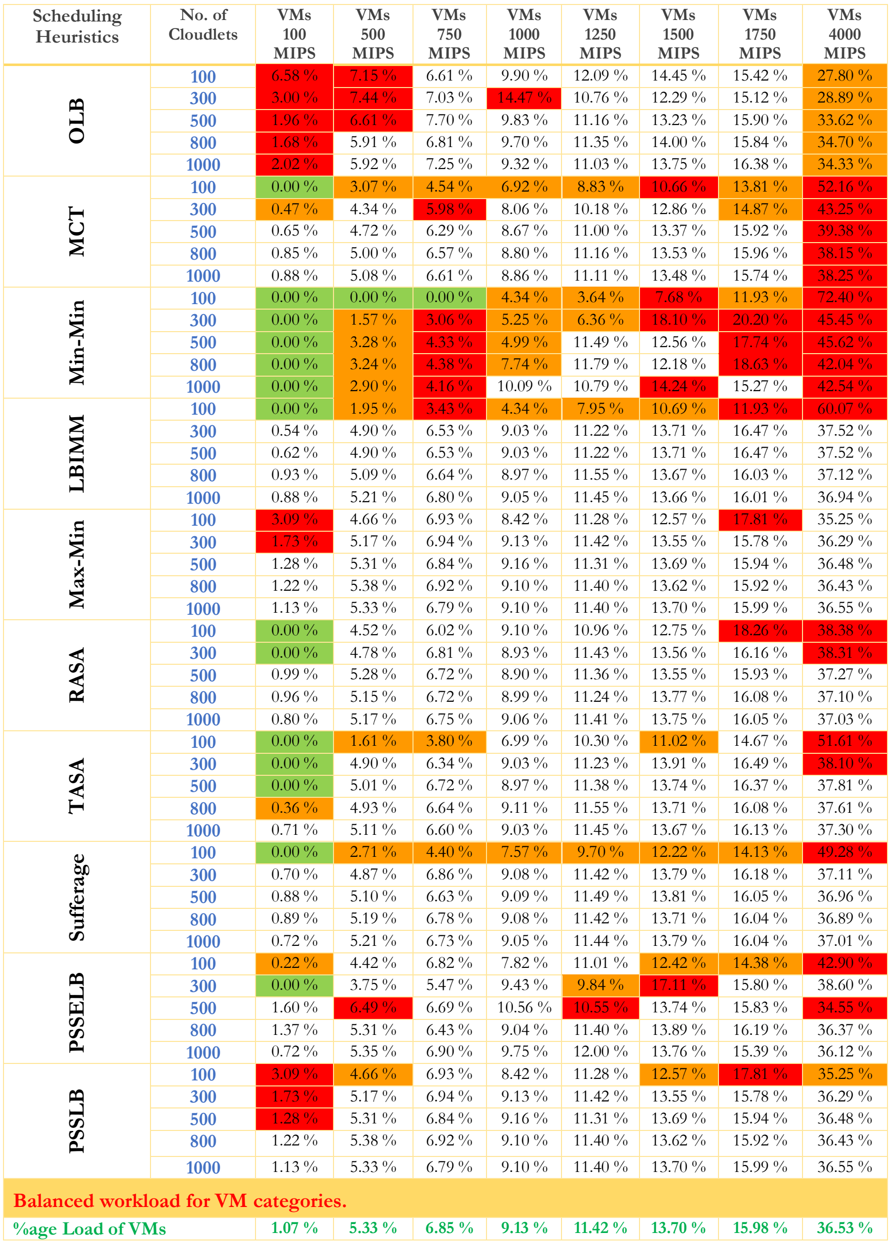
where $Share_j$ is the amount of workload in terms of MI that needs to be allocated to VM_j to attain a load-balanced scheduling. The balance share for each VM category in terms of percentage workload is represented as $VMCat\_Share_c$ and is calculated as follows:

$$VMCat\_Share_c = \frac{\sum_{a=1}^{cm} Share_a}{\sum_{i=1}^{n} Cloudlet_i.MI} \times 100 \tag{9}$$

where c represents the VM category and cm is the number of VMs in the VM category c.

Percentage workload distribution by the 10 scheduling algorithms is presented in a tabular form to highlight the load imbalance (see Figure 11). $VMCat\_Share_c$ of VM categories (see Figure 2) in a simulation environment is presented as a reference for load distribution attained by the employed scheduling algorithms (see Figure 11, presented in the last row). The imbalanced workload allocations are depicted such as the underutilized resources are filled with orange-color background, heavily loaded resources with red-background, and idle resources with a green background.

All VMs based on 100 MIPS (14 % of computing nodes in the experimental setup) and a few of VMs with 500 and 750 MIPS remain idle when the GoCJ workload is scheduled using Min-Min algorithm. Additionally, the Min-Min overloads the fastest VMs (based on 4 000 MIPS). On the other hand, the Max-Min scheduling produces better workload distribution as compared to the Min-Min; however, only a few VMs (both the slower and faster) are overloaded. This load mapping scenario is mainly contributed by the composition of GoCJ workload; where a small portion of large-sized cloudlets is present along with a majority of small-sized cloudlets. The Max-Min algorithm overcomes the imbalance produced by the Min-Min due to the presence of a few large-sized cloudlets which suits the Max-Min scheduling.



Scheduling Heuristics	No. of Cloudlets	VMs 100 MIPS	VMs 500 MIPS	VMs 750 MIPS	VMs 1000 MIPS	VMs 1250 MIPS	VMs 1500 MIPS	VMs 1750 MIPS	VMs 4000 MIPS
OLB	100	6.58 %	7.15 %	6.61 %	9.90 %	12.09 %	14.45 %	15.42 %	27.80 %
	300	3.00 %	7.44 %	7.03 %	14.47 %	10.76 %	12.29 %	15.12 %	28.89 %
	500	1.96 %	6.61 %	7.70 %	9.83 %	11.16 %	13.23 %	15.90 %	33.62 %
	800	1.68 %	5.91 %	6.81 %	9.70 %	11.35 %	14.00 %	15.84 %	34.70 %
	1000	2.02 %	5.92 %	7.25 %	9.32 %	11.03 %	13.75 %	16.38 %	34.33 %
MCT	100	0.00 %	3.07 %	4.54 %	6.92 %	8.83 %	10.66 %	13.81 %	52.16 %
	300	0.47 %	4.34 %	5.98 %	8.06 %	10.18 %	12.86 %	14.87 %	43.25 %
	500	0.65 %	4.72 %	6.29 %	8.67 %	11.00 %	13.37 %	15.92 %	39.38 %
	800	0.85 %	5.00 %	6.57 %	8.80 %	11.16 %	13.53 %	15.96 %	38.15 %
	1000	0.88 %	5.08 %	6.61 %	8.86 %	11.11 %	13.48 %	15.74 %	38.25 %
Min-Min	100	0.00 %	0.00 %	0.00 %	4.34 %	3.64 %	7.68 %	11.93 %	72.40 %
	300	0.00 %	1.57 %	3.06 %	5.25 %	6.36 %	18.10 %	20.20 %	45.45 %
	500	0.00 %	3.28 %	4.33 %	4.99 %	11.49 %	12.56 %	17.74 %	45.62 %
	800	0.00 %	3.24 %	4.38 %	7.74 %	11.79 %	12.18 %	18.63 %	42.04 %
	1000	0.00 %	2.90 %	4.16 %	10.09 %	10.79 %	14.24 %	15.27 %	42.54 %
LBIMM	100	0.00 %	1.95 %	3.43 %	4.34 %	7.95 %	10.69 %	11.93 %	60.07 %
	300	0.54 %	4.90 %	6.53 %	9.03 %	11.22 %	13.71 %	16.47 %	37.52 %
	500	0.62 %	4.90 %	6.53 %	9.03 %	11.22 %	13.71 %	16.47 %	37.52 %
	800	0.93 %	5.09 %	6.64 %	8.97 %	11.55 %	13.67 %	16.03 %	37.12 %
	1000	0.88 %	5.21 %	6.80 %	9.05 %	11.45 %	13.66 %	16.01 %	36.94 %
Max-Min	100	3.09 %	4.66 %	6.93 %	8.42 %	11.28 %	12.57 %	17.81 %	35.25 %
	300	1.73 %	5.17 %	6.94 %	9.13 %	11.42 %	13.55 %	15.78 %	36.29 %
	500	1.28 %	5.31 %	6.84 %	9.16 %	11.31 %	13.69 %	15.94 %	36.48 %
	800	1.22 %	5.38 %	6.92 %	9.10 %	11.40 %	13.62 %	15.92 %	36.43 %
	1000	1.13 %	5.33 %	6.79 %	9.10 %	11.40 %	13.70 %	15.99 %	36.55 %
RASA	100	0.00 %	4.52 %	6.02 %	9.10 %	10.96 %	12.75 %	18.26 %	38.38 %
	300	0.00 %	4.78 %	6.81 %	8.93 %	11.43 %	13.56 %	16.16 %	38.31 %
	500	0.99 %	5.28 %	6.72 %	8.90 %	11.36 %	13.55 %	15.93 %	37.27 %
	800	0.96 %	5.15 %	6.72 %	8.99 %	11.24 %	13.77 %	16.08 %	37.10 %
	1000	0.80 %	5.17 %	6.75 %	9.06 %	11.41 %	13.75 %	16.05 %	37.03 %
TASA	100	0.00 %	1.61 %	3.80 %	6.99 %	10.30 %	11.02 %	14.67 %	51.61 %
	300	0.00 %	4.90 %	6.34 %	9.03 %	11.23 %	13.91 %	16.49 %	38.10 %
	500	0.00 %	5.01 %	6.72 %	8.97 %	11.38 %	13.74 %	16.37 %	37.81 %
	800	0.36 %	4.93 %	6.64 %	9.11 %	11.55 %	13.71 %	16.08 %	37.61 %
	1000	0.71 %	5.11 %	6.60 %	9.03 %	11.45 %	13.67 %	16.13 %	37.30 %
Sufferage	100	0.00 %	2.71 %	4.40 %	7.57 %	9.70 %	12.22 %	14.13 %	49.28 %
	300	0.70 %	4.87 %	6.86 %	9.08 %	11.42 %	13.79 %	16.18 %	37.11 %
	500	0.88 %	5.10 %	6.63 %	9.09 %	11.49 %	13.81 %	16.05 %	36.96 %
	800	0.89 %	5.19 %	6.78 %	9.08 %	11.42 %	13.71 %	16.04 %	36.89 %
	1000	0.72 %	5.21 %	6.73 %	9.05 %	11.44 %	13.79 %	16.04 %	37.01 %
PSSELB	100	0.22 %	4.42 %	6.82 %	7.82 %	11.01 %	12.42 %	14.38 %	42.90 %
	300	0.00 %	3.75 %	5.47 %	9.43 %	9.84 %	17.11 %	15.80 %	38.60 %
	500	1.60 %	6.49 %	6.69 %	10.56 %	10.55 %	13.74 %	15.83 %	34.55 %
	800	1.37 %	5.31 %	6.43 %	9.04 %	11.40 %	13.89 %	16.19 %	36.37 %
	1000	0.72 %	5.35 %	6.90 %	9.75 %	12.00 %	13.76 %	15.39 %	36.12 %
PSSLB	100	3.09 %	4.66 %	6.93 %	8.42 %	11.28 %	12.57 %	17.81 %	35.25 %
	300	1.73 %	5.17 %	6.94 %	9.13 %	11.42 %	13.55 %	15.78 %	36.29 %
	500	1.28 %	5.31 %	6.84 %	9.16 %	11.31 %	13.69 %	15.94 %	36.48 %
	800	1.22 %	5.38 %	6.92 %	9.10 %	11.40 %	13.62 %	15.92 %	36.43 %
	1000	1.13 %	5.33 %	6.79 %	9.10 %	11.40 %	13.70 %	15.99 %	36.55 %
Balanced workload for VM categories.									
%age Load of VMs		1.07 %	5.33 %	6.85 %	9.13 %	11.42 %	13.70 %	15.98 %	36.53 %

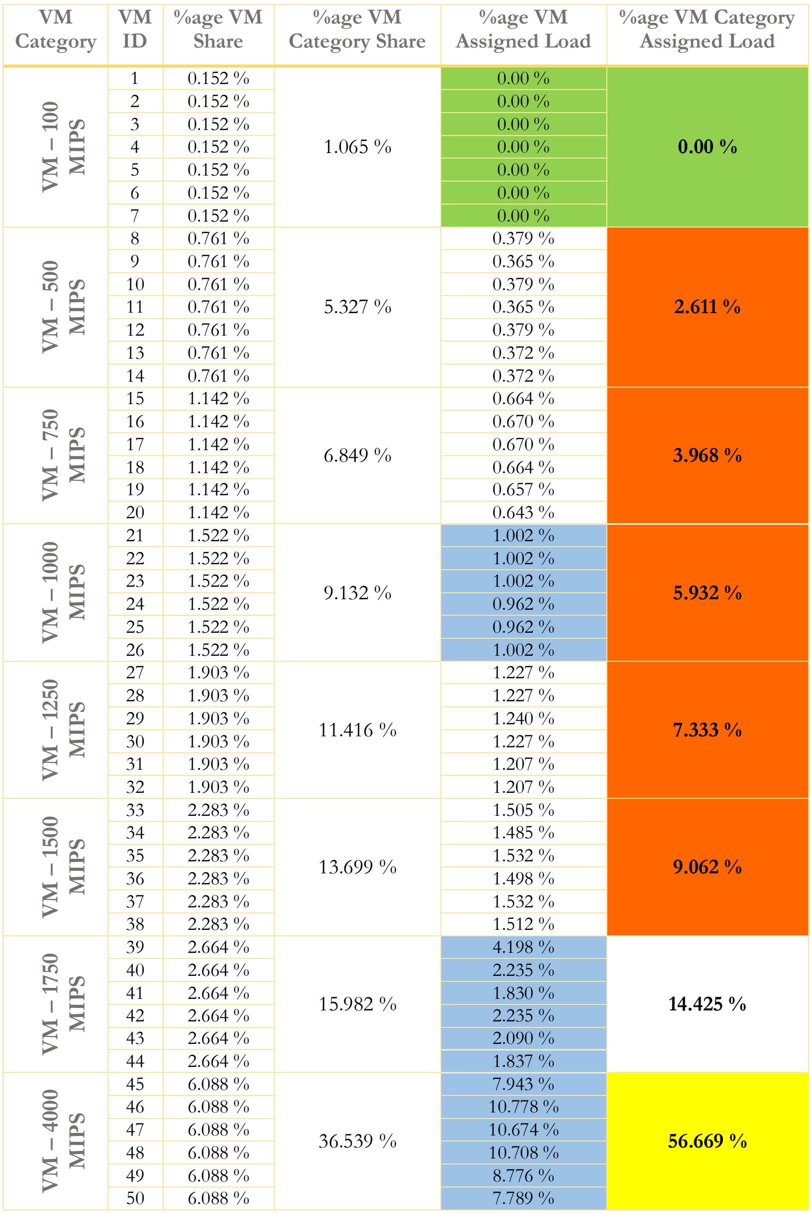
Figure 11. VM Category-wise percentage workload distribution for GoCJ workload

The PSSLB algorithm shows almost the same behavior for the workload distribution like Max-Min because both of these algorithms favor larger jobs. The LBIMM, Sufferage, RASA, Max-Min, and TASA produce comparatively a load-balanced schedule. Among these algorithms, Sufferage produces the highest resource utilization because of the resource-aware mechanism. However, an interesting observation is that the VMs based on 100 MIPS remain idle. Moreover, the recourse-aware mechanism employed by the Sufferage also produces a notable load imbalance, when the scheduling is performed using lesser number of cloudlets (i.e., 100 cloudlets), as shown in Figure 11. Similarly, LBIMM algorithm shows an improved load balancing in workload distribution. Also, RASA scheduling produces better load balancing due to the inherent usage of Min-Min and Max-Min (in alternate scheduling steps). However, the slowest machines remain idle (due to the inherent use of Min-Min algorithm) and the fastest VMs remain overloaded when a small number of cloudlets are scheduled by the RASA (see Figure 11). The employed alternate Min-Min and Max-Min mechanisms (by the RASA) produce a fair scheduling for both the large and small size cloudlets. Similarly, TASA technique produces the minimal makespan among the ten employed scheduling heuristics. However, most of the slower VMs (i.e., 100 and 500 MIPS based) become idle due to the use of Min-Min in alternate scheduling steps. On the other hand, TASA overcomes the load imbalance (caused by the Min-Min algorithm) to some extent with the help of inherent Sufferage based mechanism (in alternate scheduling steps). The Sufferage scheduling heuristic produces a better load-balanced schedule; however, very few slow VMs (with 100 MIPS) remain idle.

The results reveal that there is sufficient possibility of imbalance workload distribution (among VMs) even a scheduling technique attains an improved ARUR value; (as presented in Section 4.3). It is empirically evident that most of the existing scheduling mechanisms produce a reduced makespan with a higher throughput. However, often these algorithms result in a load imbalanced scheduling.

For example, Sufferage produces a higher ARUR value 0.863 (i.e., 86.3 % resource utilization) using the GoCJ workload (see Figure 22). However, the scheduling by Sufferage in this scenario does not utilize the VMs with computer power of 100 MIPS (i.e., those VMs remained idle). In addition, VMs with the computing capability of 500 and 750 MIPS are underutilized too and the VMs with 4 000 MIPS are heavily loaded (for the schedule of 100 cloudlets-based job pool (see Figure 11)). On the other hand, the VMs with the computing power of 100 MIPS were being utilized by the Sufferage scheduling when the number of cloudlets in the job pool increased.

Similarly, LBIMM algorithm attains 0.846 ARUR (i.e., 84.6 % resource utilization); however, VMs with 100 MIPS remain idle. Moreover, VMs with 4 000 MIPS are observed heavily overloaded and all the other VMs (in the employed experimental setup) are observed as underutilized (for 100 cloudlets-based scheduling). TASA technique produces 0.795 ARUR (i.e., 79.5 % resource utilization) for the GoCJ workload (see Figure 10); however, VMs with 100 MIPS remain idle (see Figure 11). TASA utilizes the VMs with 100 MIPS; however, most of these VMs (100 MIPS based) remained underutilized when the number of cloudlets to be scheduled are



VM Category	VM ID	%age VM Share	%age VM Category Share	%age VM Assigned Load	%age VM Category Assigned Load
VM – 100 MIPS	1	0.152 %	1.065 %	0.00 %	**0.00 %**
	2	0.152 %		0.00 %	
	3	0.152 %		0.00 %	
	4	0.152 %		0.00 %	
	5	0.152 %		0.00 %	
	6	0.152 %		0.00 %	
	7	0.152 %		0.00 %	
VM – 500 MIPS	8	0.761 %	5.327 %	0.379 %	**2.611 %**
	9	0.761 %		0.365 %	
	10	0.761 %		0.379 %	
	11	0.761 %		0.365 %	
	12	0.761 %		0.379 %	
	13	0.761 %		0.372 %	
	14	0.761 %		0.372 %	
VM – 750 MIPS	15	1.142 %	6.849 %	0.664 %	**3.968 %**
	16	1.142 %		0.670 %	
	17	1.142 %		0.670 %	
	18	1.142 %		0.664 %	
	19	1.142 %		0.657 %	
	20	1.142 %		0.643 %	
VM – 1000 MIPS	21	1.522 %	9.132 %	1.002 %	**5.932 %**
	22	1.522 %		1.002 %	
	23	1.522 %		1.002 %	
	24	1.522 %		0.962 %	
	25	1.522 %		0.962 %	
	26	1.522 %		1.002 %	
VM – 1250 MIPS	27	1.903 %	11.416 %	1.227 %	**7.333 %**
	28	1.903 %		1.227 %	
	29	1.903 %		1.240 %	
	30	1.903 %		1.227 %	
	31	1.903 %		1.207 %	
	32	1.903 %		1.207 %	
VM – 1500 MIPS	33	2.283 %	13.699 %	1.505 %	**9.062 %**
	34	2.283 %		1.485 %	
	35	2.283 %		1.532 %	
	36	2.283 %		1.498 %	
	37	2.283 %		1.532 %	
	38	2.283 %		1.512 %	
VM – 1750 MIPS	39	2.664 %	15.982 %	4.198 %	**14.425 %**
	40	2.664 %		2.235 %	
	41	2.664 %		1.830 %	
	42	2.664 %		2.235 %	
	43	2.664 %		2.090 %	
	44	2.664 %		1.837 %	
VM – 4000 MIPS	45	6.088 %	36.539 %	7.943 %	**56.669 %**
	46	6.088 %		10.778 %	
	47	6.088 %		10.674 %	
	48	6.088 %		10.708 %	
	49	6.088 %		8.776 %	
	50	6.088 %		7.789 %	

Figure 12. VM-wise percentage workload distribution by Min-Min using 250 cloudlets of GoCJ workload

increased in the experiments. Min-Min scheduling technique produces 0.52 ARUR (i.e., 52 % resource utilization) for the GoCJ workload. Figure 11 depicts a load imbalance profile of the workload distribution by the Min-Min scheduling algorithm. The VMs with 100 MIPS remain idle due to the imbalanced scheduling by Min-Min algorithm. The VMs with 500 and 750 MIPS are assigned with cloudlets; however, these VMs are significantly underutilized (as shown in Figure 11), when the number of cloudlets increases. Similarly, VMs with 1 000, 1 250, 1 500, and 1 750 MIPS also remain underutilized for the Min-Min based scheduling. Contrarily, VMs with 4 000 MIPS are heavily overloaded by Min-Min technique (see Figure 11).

This empirical investigation reveals that the scheduling algorithms producing better ARUR value still result in a machine level load-imbalance for workload distribution. The imbalanced distribution of workload among the VMs within the same VM category is also observed. Figure 12 presents the workload allocation among all VMs by the Min-Min scheduling algorithm for the GoCJ workload (using 250 cloudlets). Figure 12 highlights the imbalanced distribution of workload. The underutilized VM categories are highlighted in orange color. The heavily overloaded or idle VM categories are highlighted with the yellow and green background, respectively. Similarly, the load imbalance of VMs within a specific VM category is shown with a light-blue color. Despite balancing the workload assigned to VM category with 1 750 MIPS, it can be seen that the VM with ID 39 is heavily overloaded (i.e., 4.198 % workload is assigned) and VMs with ID 41 and ID 45 are underutilized with only 1.830 and 1.837 % workload assignment, respectively (see Figure 12).

A higher resource utilization can be attained if all the VMs in Cloud exhibit approximately the same completion time. The load balance execution guarantees that all the computing resources (i.e., VMs) are being fully utilized and there are no idle resources. Ultimately, the minimal makespan with maximal throughput will be ensured. The load balanced execution in a compute Cloud is a highly desirable aspect that will ensure lower makespan in amalgamation with higher throughput, higher resource utilization, and less energy cost. In summary, this empirical investigation highlights the following issues and potential research directions:

- a balanced distribution of workload among computing resources to be accomplished to achieve improved resource utilization with reduced makespan, and increased throughput in Cloud computing;
- designing and implementing a resource-aware holistic scheduling that not only considers application's computing requirements, but also contemplates virtual machine level attributes to provide a higher ARUR and near-optimal load balancing for the Cloud workload execution.

6 CONCLUSIONS

The inefficient utilization of resources by investigating the static heuristics for workload execution is empirically analyzed in this study. For this purpose, ten renowned Cloud scheduling heuristics are scrutinized and a comprehensive empirical study is conducted using the CloudSim simulation tool. The experiments are conducted using three workloads: two synthetics (i.e., Synthetic-I and Synthetic-II) and one benchmark GoCJ workloads. All the three workloads are based on static, non-pre-emptive, and compute-intensive Cloud jobs. Sufferage, LBIMM, Max-Min, and RASA produced the higher ARUR (i.e., 86.3 %, 84.6 %, 84.5 %, and 83.8 % resource utilization, respectively) using GoCJ workload. For LBIMM, Sufferage, and RASA based scheduling, most of the VMs remain idle or underutilized and the faster VMs (4 000 MIPS) are overloaded with the imbalanced workload. TASA scheduling mechanism has achieved a resource utilization of up to 79.5 % for the GoCJ workload while the majority of machines (based on 100 to 500 MIPS) mostly remained idle or underutilized and the faster machines (i.e., 4 000 MIPS) were overloaded. Similarly, the RASA scheduling mechanism produces a schedule that results in slower machines being idle (for the small-sized job pool); however, a gradual improvement in load-balanced was observed for the large size job pool. These results reveal that the outperforming heuristics are also unable to utilize the full computing capacity of Cloud resources. This empirical study carves out that the improper resource utilization and load imbalance is a crucial research issue that needs to be addressed comprehensively. This study identifies that the workload should be mapped in a balanced manner among the virtual machines considering both the computing capabilities of the Cloud resources and the applications computing requirements. In a consequent to this work, the authors are designing and implementing a resource-aware scheduler that considers the machines computing capabilities, the applications' computing requirements, and a balanced workload distribution constraint.

REFERENCES

[1] Aditya, A.—Chatterjee, U.—Gupta, S.: A Comparative Study of Different Static and Dynamic Load Balancing Algorithm in Cloud Computing with Special Emphasis on Time Factor. International Journal of Current Engineering and Technology, Vol. 5, 2015, No. 3, pp. 1898–1907.

[2] Jennings, B.—Stadler, R.: Resource Management in Clouds: Survey and Research Challenges. Journal of Network and Systems Management, Vol. 23, 2015, No. 3, pp. 567–619, doi: 10.1007/s10922-014-9307-7.

[3] Madni, S. H. H.—Abd Latiff, M. S.—Abdullahi, M.—Abdulhamid, S. M.—Usman, M. J.: Performance Comparison of Heuristic Algorithms for Task Scheduling in IaaS Cloud Computing Environment. PLoS One, Vol. 12, 2017, No. 5, pp. 1–26, doi: 10.1371/journal.pone.0176321.

[4] Mohialdeen, I. A.: Comparative Study of Scheduling Algorithms in Cloud Computing Environment. Journal of Computer Science, Vol. 9, 2013, No. 2, pp. 252–263, doi: 10.3844/jcssp.2013.252.263.

[5] Elzeki, O. M.—Rashad, M. Z.—Elsoud, M. A.: Overview of Scheduling Tasks in Distributed Computing Systems. International Journal of Soft Computing and Engineering, Vol. 2, 2012, No. 3, pp. 470–475.

[6] Maheswaran, M.—Ali, S.—Siegel, H. J.—Hensgen, D.—Freund, R. F.: Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. Journal of Parallel and Distributed Computing, Vol. 59, 1999, No. 2, pp. 107–131, doi: 10.1006/jpdc.1999.1581.

[7] Lenzini, L.—Mingozzi, E.—Stea, G.: Tradeoffs Between Low Complexity, Low Latency, and Fairness with Deficit Round-Robin Schedulers. IEEE/ACM Transactions on Networking, Vol. 12, 2004, No. 4, pp. 681–693, doi: 10.1109/tnet.2004.833131.

[8] Maipan-uku, J. Y.—Muhammed, A.—Abdullah, A.—Hussin, M.: Max-Average: An Extended Max-Min Scheduling Algorithm for Grid Computing Environment. Journal of Telecommunication, Electronic and Computer Engineering, Vol. 8, 2016, No. 6, pp. 43–47.

[9] Biradar, S.—Pawar, D.: A Review Paper of Improving Task Division Assignment Using Heuristics. International Journal of Science and Research, Vol. 4, 2015, No. 1, pp. 609–613.

[10] Mathew, T.—Sekaran, K. C.—Jose, J.: Study and Analysis of Various Task Scheduling Algorithms in the Cloud Computing Environment. 2014 IEEE International Conference on Advances in Computing, Communications and Informatics (ICACCI), 2014, pp. 658–664, doi: 10.1109/icacci.2014.6968517.

[11] Chen, H.—Wang, F.—Helian, N.—Akanmu, G.: User-Priority Guided Min-Min Scheduling Algorithm for Load Balancing in Cloud Computing. 2013 National Conference on Parallel Computing Technologies (PARCOMPTECH 2013), 2013, pp. 1–8, doi: 10.1109/parcomptech.2013.6621389.

[12] Parsa, S.—Entezari-Maleki, R.: RASA: A New Grid Task Scheduling Algorithm. International Journal of Digital Content Technology and Its Applications, Vol. 3, 2009, No. 4, pp. 152–160, doi: 10.4156/jdcta.vol3.issue4.10.

[13] Braun, T. D.—Siegel, H. J.—Beck, N.—Bölöni, L. L.—Maheswaran, M.—Reuther, A. I.—Robertson, J. P.—Theys, M. D.—Yao, B.—Hensgen, D.—Freund, R. F.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, Vol. 61, 2001, No. 6, pp. 810–837, doi: 10.1006/jpdc.2000.1714.

[14] Tabak, E. K.—Cambazoglu, B. B.—Aykanat, C.: Improving the Performance of Independent Task Assignment Heuristics MinMin, MaxMin and Sufferage. IEEE Transactions on Parallel and Distributed Systems, Vol. 25, 2014, No. 5, pp. 1244–1256, doi: 10.1109/tpds.2013.107.

[15] Li, B.—Pei, Y.—Wu, H.—Shen, B.: Heuristics to Allocate High-Performance Cloudlets for Computation Offloading in Mobile Ad Hoc Clouds. Journal of Supercomputing, Vol. 71, 2015, No. 8, pp. 3009–3036, doi: 10.1007/s11227-015-1425-9.

[16] Dehkordi, S. T.—Bardsiri, V. K.: TASA: A New Task Scheduling Algorithm in Cloud Computing. Journal of Advances in Computer Engineering and Technology, Vol. 1, 2015, No. 4, pp. 25–32.

[17] Alaei, N.—Safi-Esfahani, F.: RePro-Active: A Reactive – Proactive Scheduling Method Based on Simulation in Cloud Computing. Journal of Supercomputing, Vol. 74, 2018, No. 2, pp. 801–829, doi: 10.1007/s11227-017-2161-0.

[18] Tchernykh, A.—Lozano, L.—Schwiegelshohn, U.—Bouvry, P.—Pecero, J. E.—Nesmachnow, S.—Drozdov, A. Y.: Online Bi-Objective Scheduling for IaaS Clouds Ensuring Quality of Service. Journal of Grid Computing, Vol. 14, 2016, No. 1, pp. 5–22, doi: 10.1007/s10723-015-9340-0.

[19] Elzeki, O. M.—Reshad, M. Z.—Elsoud, M. A.: Improved Max-Min Algorithm in Cloud Computing. International Journal of Computer Applications, Vol. 50, 2012, No. 12, pp. 22–27, doi: 10.5120/7823-1009.

[20] Soltani, N.—Neysiani, B. S.—Barekatain, B.: Heuristic Algorithms for Task Scheduling in Cloud Computing: A Survey. International Journal of Computer Network and Information Security, Vol. 9, 2017, No. 8, pp. 16–22, doi: 10.5815/ijcnis.2017.08.03.

[21] Mao, Y.—Chen, X.—Li, X.: Max-Min Task Scheduling Algorithm for Load Balance in Cloud Computing. In: Patnaik, S., Li, X. (Eds.): Proceedings of International Conference on Computer Science and Information Technology. Springer, New Delhi, Advances in Intelligent Systems and Computing, Vol. 255, 2014, pp. 457–465, doi: 10.1007/978-81-322-1759-6_53.

[22] Sharma, G.—Banga, P.: Task Aware Switcher Scheduling for Batch Mode Mapping in Computational Grid Environment. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 3, 2013, No. 6, pp. 1292–1299.

[23] Li, H.—Wang, J.—Peng, J.—Wang, J.—Liu, T.: Energy-Aware Scheduling Scheme Using Workload-Aware Consolidation Technique in Cloud Data Centres. China Communications, Vol. 10, 2013, No. 12, pp. 114–124, doi: 10.1109/cc.2013.6723884.

[24] De Falco, I.—Scafuri, U.—Tarantino, E.: Two New Fast Heuristics for Mapping Parallel Applications on Cloud Computing. Future Generation Computer Systems, Vol. 37, 2014, pp. 1–13, doi: 10.1016/j.future.2014.02.019.

[25] Panda, S. K.—Agrawal, P.—Khilar, P. M.—Mohapatra, D. P.: Skewness-Based Min-Min Max-Min Heuristic for Grid Task Scheduling. Proceedings of the 2014 Fourth International Conference on Advanced Computing and Communication Technologies (ACCT '14), 2014, pp. 282–289.

[26] Yu, X.—Yu, X.: A New Grid Computation-Based Min-Min Algorithm. 2009 Sixth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2009), 2009, pp. 43–45, doi: 10.1109/fskd.2009.81.

[27] Thaman, J.—Singh, M.: Green Cloud Environment by Using Robust Planning Algorithm. Egyptian Informatics Journal, Vol. 18, 2017, No. 3, pp. 205–214, doi: 10.1016/j.eij.2017.02.001.

[28] Mehdi, N. A.—Mamat, A.—Ibrahim, H.—Subramaniam, S. K.: Impatient Task Mapping in Elastic Cloud Using Genetic Algorithm. Journal of Computer Science, Vol. 7, 2011, No. 6, pp. 877–883, doi: 10.3844/jcssp.2011.877.883.

[29] Patel, R.—Chandel, M.: Analysis of Various Task Scheduling Algorithms in Cloud Computing. International Research Journal of Engineering and Technology, Vol. 3, 2016, No. 3, pp. 493–496.

[30] Gupta, K.—Katiyar, V.: Survey of Resource Provisioning Heuristics in Cloud and Their Parameters. International Journal of Computational Intelligence Research, Vol. 13, 2017, No. 5, pp. 1283–1300.

[31] Calheiros, R. N.—Ranjan, R.—Beloglazov, A.—De Rose, C. A. F.—Buyya, R.: CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms. Software: Practice and Experience, Vol. 41, 2011, No. 1, pp. 23–50, doi: 10.1002/spe.995.

[32] Liu, Z.—Cho, S.: Characterizing Machines and Workloads on a Google Cluster. 2012 41st International Conference on Parallel Processing Workshops, 2012, pp. 397–403, doi: 10.1109/icppw.2012.57.

[33] Moreno, I. S.—Garraghan, P.—Townend, P.—Xu, J.: An Approach for Characterizing Workloads in Google Cloud to Derive Realistic Resource Utilization Models. Proceedings of the 2013 IEEE Seventh International Symposium on Service-Oriented System Engineering, 2013, pp. 49–60, doi: 10.1109/sose.2013.24.

[34] Reiss, C.—Tumanov, A.—Ganger, G. R.—Katz, R. H.—Kozuch, M. A.: Towards Understanding Heterogeneous Clouds at Scale: Google Trace Analysis. Intel Science and Technology Center for Cloud Computing, Technical Report ISTC–CC–TR–12–101, 2012.

[35] Chen, Y.—Ganapathi, A. S.—Griffith, R.—Katz, R. H.: Analysis and Lessons from a Publicly Available Google Cluster Trace. EECS Department, University of California, Berkeley, Technical Report No. UCB/EECS–2010–95, 2010.

[36] Kavulya, S.—Tan, J.—Gandhi, R.—Narasimhan, P.: An Analysis of Traces from a Production MapReduce Cluster. 2010 11th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid), 2010, pp. 94–103, doi: 10.1109/ccgrid.2010.112.

[37] Hussain, A.—Aleem, M.: GoCJ: Google Cloud Jobs Dataset for Distributed and Cloud Computing Infrastructures. MDPI Data, Vol. 3, 2018, No. 4, pp. 1–12, doi: 10.3390/data3040038.

[38] Mehdi, N. A.—Mamat, A.—Ibrahim, H.—Syrmabn, S. K.: Virtual Machines Cooperation for Impatient Jobs under Cloud Paradigm. International Journal of Computer and Information Engineering, Vol. 5, 2011, No. 3, pp. 300–306.

[39] Behzad, S.—Fotohi, R.—Effatparvar, M.: Queue Based Job Scheduling Algorithm for Cloud Computing. International Research Journal of Applied and Basic Sciences, Vol. 4, 2013, No. 11, pp. 3785–3790.

[40] Liu, L.—Mei, H.—Xie, B.: Towards a Multi-QoS Human-Centric Cloud Computing Load Balance Resource Allocation Method. The Journal of Supercomputing, Vol. 72, 2016, No. 7, pp. 2488–2501, doi: 10.1007/s11227-015-1472-2.

Altaf Hussain received his M.S. degree in computer software engineering from National University of Science and Technology (NUST), Islamabad, Pakistan in 2010. He received his B.S. in computer science with distinction from NWFP AUP, Pakistan. His research interests include software testing, data mining, sentiment analysis and distributing computing comprising scheduling, performance analysis and Cloud computing. He is currently a Ph.D. scholar and the member of Parallel Computing Network (PCN) Research Group at Capital University of Science and Technology, Islamabad, Pakistan.

Muhammad Aleem received his Ph.D. degree in computer science from the Leopold-Franzens Universität Innsbruck, Austria in 2012. His research interests include parallel and distributed computing comprising programming environments, multi-/many-core computing, performance analysis, cloud computing, and big-data processing. He is currently working as Assistant Professor at Capital University of Science and Technology, Islamabad, Pakistan.

Muhammad Azhar Iqbal is Assistant Professor at the Capital University of Science and Technology, Islamabad, Pakistan. He received his Ph.D. degree in communication and information systems from the Huazhong University of Science and Technology, Wuhan, P.R. China in 2012. His research interests include coding-aware routing in vehicular ad hoc networks, energy-efficient MAC for wireless body area networks, largescale simulation modeling and analysis of computer networks in Cloud.

Muhammad Arshad Islam completed his doctorate from University of Konstanz, Germany in 2011. His dissertation is related to routing issues in opportunistic network. His current research interests are related to MANETs, DTNs, social-aware routing and graph algorithms. He is currently working as Assistant Professor at Capital University of Science and Technology, Islamabad, Pakistan.

Computing and Informatics, Vol. 38, 2019, 555–574, doi: 10.31577/cai_2019_3_555

SUFFIX ARRAYS WITH A TWIST

Tomasz M. Kowalski, Szymon Grabowski

Łódź University of Technology
Institute of Applied Computer Science
Al. Politechniki 11, 90–924 Łódź, Poland
e-mail: {tkowals, sgrabow}@kis.p.lodz.pl

Kimmo Fredriksson

School of Computing, University of Eastern Finland
P.O.B. 1627, FI-70211 Kuopio, Finland
e-mail: kimmo.k.k.fredriksson@gmail.com

Abstract. The suffix array is a classic full-text index, combining effectiveness with simplicity. We discuss three approaches aiming to improve its efficiency even more: changes to the navigation, data layout and adding extra data. In short, we show that i) the way how we search for the right interval boundary impacts significantly the overall search speed, ii) a B-tree data layout easily wins over the standard one, iii) the well-known idea of a lookup table for the prefixes of the suffixes can be refined with using compression, iv) caching prefixes of the suffixes in a helper array can pose another practical space-time tradeoff.

Keywords: Suffix array, data structures, text indexes, hashing

Mathematics Subject Classification 2010: 68W32

1 INTRODUCTION

Everybody knows the suffix array (SA) [1], a simple full-text index data structure capable of finding the *occ* occurrences of a pattern P of length m in $O(m \log n + occ)$

time, where n is the length of the indexed text. The search mechanism consists in two binary searches, for the left and the right boundary of the interval of text suffixes starting with P, in the array of suffix offsets arranged in the lexicographical order of their text content. The performance of the suffix array can serve as a measuring stick for more advanced (e.g., compressed) text indexes [2] and at least for this reason it is important to know how to implement it efficiently and what space-time tradeoffs are possible.

The suffix array can be perceived as a simplification of the suffix tree (ST) [3], a tree whose string collection is the set of all the suffixes of a given text, with an additional requirement that all non-branching paths of edges are converted into single edges. Indeed, a suffix array can be obtained from a suffix tree by visiting its leaves in order (from left to right, obtained by depth-first traversal of the ST). Depending on the implementation, the pattern search over ST takes either $O(m \log \sigma + occ)$ or $O(m + occ)$ time (where the latter variant involves perfect hashing). ST can be built in linear time for integer alphabets [4]; a result that directly translates to linear-time SA construction (albeit more direct and economical linear-time SA construction algorithms were found later). The practical performance of ST and SA is rather comparable, yet implementation details may be important; for example, if the number of matches is large (which is typical for short patterns), the suffix array may be even by an order of magnitude faster than the suffix tree [5].

A number of attempts have been made to improve the time complexities of full-text indexes. For example, the suffix tray by Cole et al. [6], which can be seen as a cross of the suffix tree and the suffix array, allows to achieve $O(m + \log \sigma)$ search time, with $O(n)$ worst-case time construction and $O(n \log n)$ bits of space. Later, Fischer and Gawrychowski [7] reduced the search time to (deterministic) $O(m + \log \log \sigma)$, with preserved construction cost complexities. Even better time complexity, $O(m + \log \log_w \sigma)$ (where $w \geq \log n$ is the machine word size), for a compressed (sic!) index and deterministic linear time construction was recently achieved by Munro et al. [8]. Bille et al. [9] showed how to search for a *packed* pattern in a (standard) suffix array in $O(m/\alpha + \log n)$ time, where α is the number of characters one can pack in a machine word. In the same work, they presented a more involved construction allowing to search for a packed pattern in $O(m/\alpha + \log m + \log \log \sigma)$ time; the index size is still $O(n)$ words (or $O(n \log n)$ bits). We are not aware of any implementations of the algorithms mentioned in this paragraph, which means that they remain theoretical achievements so far.

The body of research on engineering the suffix array is surprisingly scarce. Although the basic SA idea can be easily grasped even by high-school students, many design choices from the implementor's point of view are not obvious. Let us pose a few questions:

1. Can the binary search strategy be replaced with a faster one, e.g., based on interpolation search?
2. As *occ* is usually small, what is practically the best way to find the right interval boundary once the left boundary is known?

3. Can we change the data layout of suffixes in order to obtain more local memory accesses?
4. How can we augment the suffix array with a moderate amount of extra data, to initially reduce the search interval and/or speed up string comparisons?

The answer to some of them is known, yet in this work we are dealing with the mentioned issues in a more systematic way.

The contributions of our paper are as follows. We show that the idea of k-ary heap layout of a sorted array, known from the earlier works [12, 13], makes practical sense also for the suffix array, due to increased data access locality. We discuss the impact of the technique for finding the right interval boundary in the suffix array on the overall performance. It is also noticed that augmenting the suffix array with extra data boosts the performance; novel techniques presented in this work include caching prefixes of the suffixes in a helper array and using a lookup table with Huffman-compressed keys.

2 IDEAS AND INCARNATIONS

The considered ideas are divided into three groups and each of them is described in a separate subsection. First we discuss non-standard SA traversal strategies. Later we advocate for alternative data layouts, beneficial for the search speed. In the last subsection some ways to augment the suffix array with extra data, to make the pattern search even faster, are proposed.

2.1 Navigating over the Suffix Array

A textbook alternative to binary search is interpolation search, which performs a number of "guesses" concerning the query's location based on the query value and the assumed distribution of keys. It is well-known that interpolation search over constant-size keys achieves $O(\log \log n)$ expected time not only for the simplest case, i.e., uniformly random distribution [10], yet we are not aware of any published experiments regarding text suffixes. Unfortunately, a straightforward interpretation of string prefixes (which have a lot of duplicates) as integers and standard linear interpolation yielded rather disappointing results.

Another question concerning the navigation over the SA is how the right interval boundary should be found. We examine two methods: a naive one performs the binary search over the range $\mathit{left} \dots n$ of suffixes, where left is the position of the least suffix greater or equal to the pattern, and the doubling (galloping) algorithm, which peeks the locations $SA[\mathit{left} + 2^i]$, $i = 0, 1, 2, \dots$, until it reaches too far and the search continues in the binary manner over the last considered interval. Note that the time complexity of the right interval boundary search improves in this way from $O(m \log n)$ to $O(m \log \mathit{occ})$.

We should also mention here using non-standard CPU instructions for binary search. Wide registers together with single-instruction multiple-data (SIMD) instruction sets are a popular extension of modern CPUs, including Intel's Pentium 4, Core 2, Nehalem and more recent architectures, Intel's Xeon, AMD's Phenom, Bulldozer and Ryzen, and ARM Cortex-A mobile processors. Zhou and Ross [11] proposed a SIMD-ized version of binary search (and other database operations) that is geared towards small datasets, up to a few hundred keys. Significant speedups were obtained as a result of the elimination of branch misprediction effects.

searchTree(*pat*, n, N, *step*)

```
$node \leftarrow 0$; $beg \leftarrow n$
while $node < N$ do  /* search down the tree from the top */
    $(c, beg) \leftarrow searchNode(pat, beg, node, 0, step)$
    $node \leftarrow childNode(node, c)$
if $isMaxPattern(pat)$ then  /* pattern is lexicographically the greatest */
    return $(beg, n)$
$pat \leftarrow incPattern(pat)$
$end \leftarrow beg$; $i \leftarrow end$; $endNode \leftarrow node(end)$
while $true$ do  /* search up the tree from the current node */
    if $pat < T[karySA[end]]$ then break
    $i \leftarrow end + 1$; $node \leftarrow endNode$
    while $true$ do  /* search for previous $beg$ value */
        if $endNode = 0$ then
            $end \leftarrow n$; break 2
        $(endNode, c) \leftarrow parent(endNode)$
        if $c < k - 1$ then break
    $end \leftarrow index(endNode) + c$
if $end = beg$ then  /* pattern not found */
    return $(beg, beg)$
$c \leftarrow elemOffset(i)$
$(c, end) \leftarrow searchNode(pat, end, node, c, step)$
$node \leftarrow childNode(node, c)$
while $node < N$ do  /* search down the tree from the current node */
    $(c, end) \leftarrow searchNode(pat, end, node, 0, step)$
    $node \leftarrow childNode(node, c)$
return $(beg, end)$
```

Figure 1. The $searchTree(pat, n, N, step)$ function, returning the first and the last index in the search tree corresponding to the range of suffixes of the indexed text starting with the string *pat*. The parameters n and N ($N \leq n$) refer to the number of suffixes and the number of nodes in the tree, respectively. The parameter *step* is passed to the searchNode function.

2.2 Linearized k-ary Tree Data Layout

Binary search over a sorted array is equivalent to walking down a path in a complete binary search tree. Schleger et al. [12] noticed that changing the tree layout from binary to k-ary ($k > 2$), together with linearization of the search tree, may be more cache-friendly and also convenient for SIMD processing. In their experiments (Intel Core i7) it achieved a speedup of as much as 3 up to 4.5 for 32-bit numbers and 2 to 2.5 for 64-bit numbers, compared to a plain binary search. This data organization can also be called an (implicit) B-tree layout [13], where the case of $B = 1$ (a complete binary tree with the root going first, then followed by its both children, etc.) is called the Eytzinger layout (dating back to old history) or the heap-order layout, as this method was proposed by Williams for an implementation of binary heaps [14]. We apply the presented idea to the suffix array, which, to our knowledge, has not been tried before. Note that setting the B-tree layout for a suffix array cannot be comparably successful as for, e.g., integers, as the accesses to the text are still at "random" areas.

The pseudocodes of algorithms on the non-standard layout are presented in Figures 1–3. The used notation and primitives (i.e., helper functions) need to be explained beforehand. The term "index" will refer to the position in a linearized k-ary tree, while "offset" to the position relative to the beginning of the node (i.e., the index relative to the beginning of the node). We use the following symbols and helper function names:

- n is the number of SA elements, i.e., the text length,
- N is the number of nodes in the tree, i.e., $N = \lceil n/B \rceil$,
- $\mathit{index}(\mathit{node})$ returns the index of the first element in the given node,
- $\mathit{node}(\mathit{index})$ returns the number of the node containing the given index,
- $\mathit{childNode}(\mathit{node}, c)$ returns the number of the c^{th} child node of the node,
- $\mathit{childNum}(\mathit{node})$ returns the number of the node among its parent's children,
- $\mathit{elemOffset}(\mathit{idx})$ returns the offset of the element,
- $\mathit{parent}(\mathit{node})$ returns a pair (p, off), where p is the parent node number and off is the smallest offset of an element in the parent node referring to a suffix not smaller than suffixes in node,
- $\mathit{incPattern}(\mathit{pat})$ returns the next pattern of the same length in lexicographical order. In the (very rare) case when pat is the lexicographically greatest pattern, the lexicographically smallest pattern of the same length is returned (however, such cases do not occur in our code),
- $isMaxPattern(\mathit{pat})$ tests if pattern is lexicographically the greatest.

Figure 1 presents a pseudocode of the function $\mathit{searchTree}(\mathit{pat}, n, N, \mathit{step})$, traversing an n-element B-tree structure comprised of N nodes, in order to return the pair $(\mathit{beg}, \mathit{end})$. The value of beg (resp. end) is the index of an element in the tree corresponding to the lexicographically smallest suffix not smaller (resp. suffix greater) than the pattern *pat*. As the tree element IDs have values in $\{0, 1, \ldots, n-1\}$, the special case of *end* set to n means that there are no suffixes lexicographically greater than *pat*.

$searchNode(pat, idx, node, startOff, step)$

```
(01) $c \leftarrow startOff + (step - 1)$; $j \leftarrow index(node) + c$
(02) while $c < k - 1$ do
(03)     if $pat < T[karySA[j]]$ then
(04)         $idx \leftarrow j$; break
(05)     $c \leftarrow c + step$; $j \leftarrow j + step$
(06) $guard \leftarrow c$
(07) if $guard > k - 1$ then $guard \leftarrow k - 1$
(08) $c \leftarrow c - (step - 1)$; $j \leftarrow j - (step - 1)$
(09) while $c < guard$ do
(10)     if $pattern < T[karySA[j]]$ then
(11)         $idx \leftarrow j$; break
(12)     $c \leftarrow c + 1$; $j \leftarrow j + 1$
(13) return $(c, idx)$
```

Figure 2. The $searchNode(pat, idx, node, startOff, step)$ function for locating the smallest element in the passed node (the third parameter) referring to a suffix lexicographically not smaller than the pattern *pat*

This function makes use of $searchNode(pat, idx, node, startOff, step)$ (Figure 2), which returns the smallest index of an element in the passed node (the third parameter) referring to a suffix not smaller than the pattern *pat*. The current index, *idx*, is updated only if a better candidate is found in the node. The parameter *startOff* stores the number of node elements which are skipped (as being lexicographically smaller than *pat*). The presented code for searchNode refers to the case of large nodes ($B > 8$), when a two-pass node lookup (the first pass with the step given as the last parameter of the function) is used.

Finally, the function count(*beg*, *end*) (Figure 3) returns the number of pattern occurrences in the range determined by the *beg* and *end* indexes. For simplicity, the presented code deals only with the case of $beg < end$, and $end < n$, i.e., when the *beg* index is located in a tree layer not lower than the layer of the index *end*. In the following paragraph we comment the main phases of this code.

In lines 01–02 we initialize the key variables, where *res* is the count to be eventually returned. The loop in lines 03–07 traverses down the tree until the layer just below the layer of the *end* index is reached. After the loop, the helper array *bOff* stores the left boundaries of the intervals from all the layers in which the elements from *beg* to *end* (inclusively) belong to. Lines 08–17 add the number of elements in the bottom layers of the tree, i.e., in the layers below the one to which *end* be-

count(*beg*, *end*)

```
$res \leftarrow 0$;   $bOff[0] \leftarrow beg$;   $b \leftarrow beg$;   $l \leftarrow 1$
$bNode \leftarrow childNode(node(beg), elemOffset(beg))$
**while** $bNode < N$ **do**
    $b \leftarrow index(bNode) + k - 1$
    **if** $b > end$ **then break**
    $bOff[l] \leftarrow b$;   $l \leftarrow l + 1$
    $bNode \leftarrow childNode(node(beg), elemOffset(beg))$
/* adding interval widths in tree layers below end index */
$e \leftarrow end$
$eNode \leftarrow childNode(node(end), elemOffset(end))$
**while** $eNode < N$ **do**
    $e \leftarrow index(eNode) + k - 1$
    **if** $e > n$ **then break**
    $res \leftarrow res + e - b$
    $eNode \leftarrow childNode(eNode, k - 1)$
    $bNode \leftarrow childNode(bNode, k - 1)$
    $b \leftarrow index(bNode) + k - 1$
**if** $bNode \leq N$ & $b \leq n$ **then** $res \leftarrow res + n - b$
/* adding interval widths in tree layers between beg and end indexes */
$e \leftarrow end$; $eNode \leftarrow node(end)$
**while** $l > 0$ **do**
    $l \leftarrow l - 1$
    $res \leftarrow res + e - bOff[l]$
    $eChild \leftarrow childNum(eNode)$;   $eNode \leftarrow parent(eNode)$
    $e \leftarrow index(eNode) + eChild$
$bNode \leftarrow node(bOff[0])$
/* adding interval widths in tree layers above beg index */
**if** $bNode = 0$ **then return** $res$
**while** $true$ **do**
    $bChild \leftarrow childNum(bNode)$;   $bNode \leftarrow parent(bNode)$
    $b \leftarrow index(bNode) + bChild$
    $res \leftarrow res + e - b$
    **if** $bNode = eNode$ **then return** $res$
    $eChild \leftarrow childNum(eNode)$;   $eNode \leftarrow parent(eNode)$
    $e \leftarrow index(eNode) + eChild$
```

Figure 3. The function *count*(*beg*, *end*), which returns the number of pattern occurrences in the range determined by the *beg* and *end* indexes of the tree structure. It is assumed in the presented code that $beg < end$ and $end < n$ (handling the other cases is similar, but would make the pseudocode much longer).

longs. The variable b (resp. e) is set to the first element from (resp. beyond) the considered interval in the current layer. Special care must be taken not to exceed the last stored element (line 17). In lines 18–24 we handle the layers between the last considered layer (containing *end*) and the first considered layer (containing *beg*). Finally, an analogous procedure continues up to the top, terminating when b and e are in the same node (lines 30).

We have also a locate function, which is very similar to count, only instead of incrementing the counter of matching suffixes it adds them to a returned list.

2.3 Augmenting the Suffix Array

Manber and Myers in their seminal paper [1] presented a nice trick saving several first steps in the binary search: if we know the SA intervals for all the possible first k symbols of the pattern, we can immediately start the binary search in a corresponding interval. We can set k to $\log_\sigma n$, where σ is the alphabet size, with $O(n \log n)$ extra bits of space and constant expected size of the interval. Unfortunately, real texts are far from random, hence in practice, we can use k up to 3 (assuming that text symbols are bytes), which offers a limited (yet, non-negligible) benefit. This idea will be referred in our experiments as using a lookup table, and more specifically we will denote the lookup table on pairs (resp. triples) of symbols with LUT2 (resp. LUT3).

In the same spirit, Grabowski and Raniszewski [15, 16] use a hash table to store the intervals for all k-symbol strings *occurring in the text.* This can significantly reduce the initial interval for real texts with relatively little extra space.

In this work we first propose a lookup table with keys being concatenations of Huffman codewords for the starting symbols of the text suffixes (Table 1), truncated to a specified length of b bits. Pattern search translates to finding the first b bits of Huffman encoding of the pattern, which is the LUT key, and then following with binary search over a range of suffixes read from the LUT. A look onto the rows, e.g., LUT-Huff-23b and LUT3, reveals that the resulting search intervals to go into are much narrower on average with the Huffman-based LUT, using the same amount of extra memory. A correct implementation of this idea, in combination with the B-tree SA layout, requires a reordering of the suffixes in the SA, to avoid nested LUT ranges (other options, like replacing Huffman with Hu-Tucker coding, are also possible, but we have not tried them out). Note also that the Huffman-based LUT entries store twice more data (both boundaries of the interval) than in the standard LUTs.

We also propose mixing the LUT or hash table interval narrowing with the B-tree layout, and also augmenting the search tree with prefixes of the suffixes in several top levels of the B-tree. Copying these text snippets into a helper array is beneficial due to more local memory accesses.

The last novelty is varying the parameter k, the length of the hashed strings. Using a fixed k results in having some intervals too wide (which deteriorates binary search) while some others are (too) narrow, which does not already help much.

	space (MiB)	dna200	english200	proteins200	xml200
LUT-Huff-15b	0.25	14.45 ± 1.21	15.53 ± 2.20	12.86 ± 0.81	17.05 ± 2.84
LUT-Huff-19b	4.00	11.07 ± 1.52	13.46 ± 2.63	9.10 ± 1.31	15.91 ± 3.64
LUT-Huff-23b	64.00	7.79 ± 1.98	11.63 ± 2.93	5.63 ± 2.17	14.99 ± 4.28
LUT2	0.25	23.74 ± 0.46	19.50 ± 2.18	19.27 ± 0.94	18.91 ± 2.40
LUT3	64.00	21.82 ± 0.61	16.55 ± 2.78	15.11 ± 1.18	16.74 ± 3.49

Table 1. Average binary logarithms (with their standard deviations) of the search interval widths for different LUT variants (first three rows: order-0 Huffman encoding with 15–23 bits, next two rows: standard 2-/3-byte LUTs). The averages are taken over all suffixes of the text. Without a LUT the corresponding binary logarithms would be $\log_2(200 \cdot 2^{20}) = 27.64$ in all cases.

Varying k is expected to have more balanced interval widths, which in turn may translate into more preferable space-time tradeoffs.

To this end, we use three parameters, $k_0 < k_1 < k_2$, corresponding to suffixes' prefix lengths, and the parameter r as an interval width threshold. The first of the three parameters, k_0, is used in a standard lookup table and was fixed to 2 throughout the experiments. If a pair of successive symbols, c_1c_2, does not occur in the text more than r times (i.e., there are at most r suffixes starting with this prefix), the suffixes starting with c_1c_2 are not inserted into any other data structure; if the pattern matches such a prefix, one access to the mentioned LUT reveals that the range of suffixes to search is of size at most r and the binary search follows.

Those suffixes which do not fall into a narrow range according to their first two symbols, are then divided into two groups, based on whether their k_1-symbol prefix occurs at most r times in the text. Those for which the answer is positive are inserted into a hash table in the manner of the SA-hash index (see [16] for more details). If a k_1-long prefix occurs more than r times though, we extend its occurrences to length k_2 and insert such (distinct) strings into the same hash table. Additionally, a bit array V is maintained, initialized with zeros. For each distinct k_1-symbol string from the text its computed hash value tells the position in V to set a bit if its count is at most r. Collisions are not handled here, which means that several different strings may overwrite the same bit in V. Yet, V happens to be (relatively) very small for real texts, which allows for small load factors (e.g., LF $= 0.1$) and in turn translates into rather few collisions.

Now, given a pattern to search, we first check its k_1-symbol prefix in V. If the accessed bit is 1, we assume that the prefix, although relatively short, is specific enough. We look for it in the hash table and the associated data is the range of suffixes in which we continue with binary search. If the bit accessed in V is 0 though, we look for the longer prefix, of length k_2, in the hash table and continue in the same manner.

Let us yet justify the presented idea using a small example. For a given (fixed) k, we (locally) obtain three adjacent intervals of width 900, 60 and 40, respectively. The numbers of binary search steps are: 10, 6 and 6, respectively. Yet, the average is not 7.333; it is rather $900/1\,000 \cdot 10 + 60/1\,000 \cdot 6 + 40/1\,000 \cdot 6 = 9.6$. This is because entering the widest interval is more likely than any of the remaining two. Now, we introduce k_1 and k_2, and it may happen that our considered intervals are split into three different intervals, of widths: 400, 500, 100, respectively. The corresponding numbers of binary search steps are now: 9, 9, 7, respectively. The (weighted) average is thus: $400/1\,000 \cdot 9 + 500/1\,000 \cdot 9 + 100/1\,000 \cdot 7 = 8.8$, i.e., yields some improvement.

3 EXPERIMENTAL RESULTS

All experiments were run on a machine equipped with a 4-core Intel i7 4790 3.6 GHz CPU and 32 GB of 1 600 MHz DDR3 RAM (9-9-9-24), hosting Windows Server 2012 R2. One CPU core was used for the computations. All codes (`https://bitbucket.org/kowallus/sa-search-dev/`) were written in C++ and compiled with 64-bit gcc 4.9.3 with `-O3`.

For each experiments, we took 500K patterns sampled from the text in a uniformly random manner, calculated the average time per pattern, repeated this procedure 11 times and presented the median. The searches were performed over 200 MB datasets from the well-known Pizza & Chili corpus (`http://pizzachili.dcc.uchile.cl/`).

In the first experiment we show how the count times are affected by two things: using lookup tables, including the introduced Huffman-based ones (on 15 or 23 bits) (Figure 5) and choosing a proper interval's right boundary search (Figure 4). HT denotes the idea of combining the suffix array with a hash table [15, 16]; it involves the parameter k, which is the length of suffixes' prefixes inserted in the hash table. In our experiments we (arbitrarily) choose the smallest k for which the overall size of the index, including the text, exceeds $5.5n$.

The doubling trick reduces the times usually by 20–30 % for the standard and LUT2-boosted suffix array, yet the effect is smaller for short patterns (i.e., small m), especially for DNA (where short patterns tend to have thousands of occurrences). This can be explained by the relatively small difference between $\log n$ and $\log occ$ in those cases. The Huffman-based LUTs are more efficient than their traditional counterparts (when about the same amount of memory is sacrificed).

For `xml200` (Figure 4) one can observe a different trend than for other datasets: the time decreases for smaller m. There are at least two factors specific to this dataset that cause such an effect. The first one is the extremely large (average) width of resulting intervals. For `english200`, the average interval width for $m = 9$ is about 10^3, while for `xml200` it is as much as 10^5. The difference in `xml200` is due to repetitiveness in the data (long XML tag names, etc.). This dataset is sensitive to m as long as the character access is involved; in datasets like `english200`, the average matching prefix length during the comparisons is not very sensitive to m (grows only slightly with growing m). The second factor is only a slight reduction of the resulting interval width with growing m. Note that most methods work faster on narrow

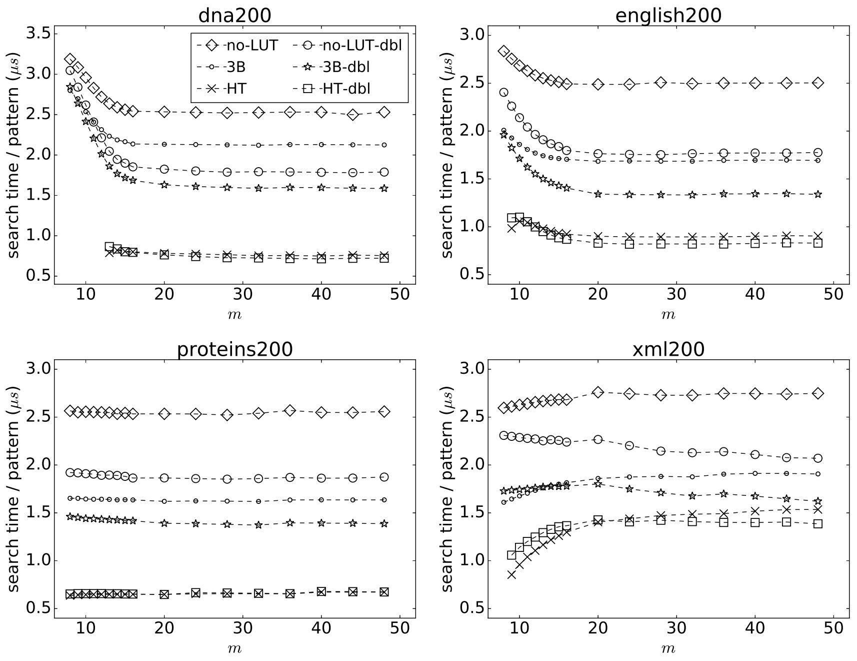


Figure 4. Count times for the standard suffix array and the SA augmented with a lookup table on triples of bytes and a hash table over k symbols ($k = 12$ for `dna200`, $k = 6$ for `proteins200` and $k = 8$ for the other datasets), using the standard and the doubling search for finding the right interval boundary. The standard SA size is $5n$ (bytes), the size with LUT3 is $5.321n$ and the size with a HT is $5.584n$, $5.882n$, $6.549n$ and $5.532n$, respectively, for `dna200`, `english200`, `proteins200` and `xml200`.

intervals, due to reduced time for finding the right boundary. Consider extending the pattern length from $m = 9$ to 10. For `english200`, the average interval length gets reduced to 56 %, while in `xml200` case to 83 % (in other words, the average interval for `xml200` shrinks by one sixth only, a really mild improvement). This effect occurs also in `proteins200`. Those two factors, taken together, may explain why for `xml200` using a smaller m may yield a shorter overall time, as opposed to other datasets. A similar reasoning also works for Figure 6.

Figure 5 shows how spending more space (from 2^{15} to 2^{25} array slots) for Huffman-based LUTs improves the count times. Apart from order-0 (i.e., context-free) Huffman coding also order-1 and order-2 Huffman models were used, to show that increasing the context order helps on compressible datasets (`english200`, `xml200`), but not on `dna200` and `proteins200` (comparable compression and more space used). Using, e.g., order-2 encoding means that the first input symbol is order-0 encoded, the second order-1 encoded and all the following ones are order-2 encoded. For `dna200`, biologically meaningful search patterns do not contain the

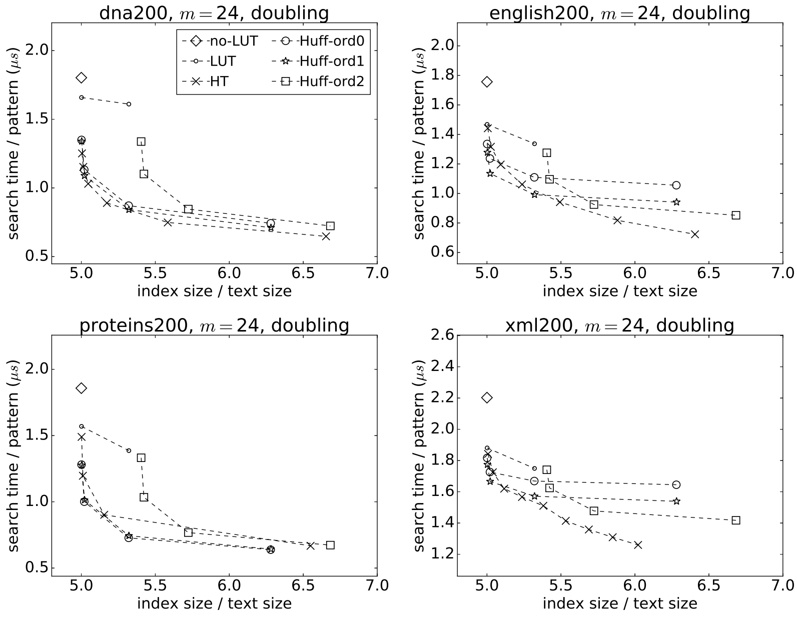


Figure 5. Count times for the standard suffix array, the SA augmented with a lookup table on pairs or triples of bytes (LUT), and on 15, 19, 23 or 25 bits of Huffman codewords (a series for a different coding order), and the hash table (HT) on varying k-grams from the text. The right interval boundary is found with the doubling technique.

N symbol and these should also be avoided in the searches with order-1 or order-2 Huffman-based LUT. The reason is that the N symbols tend to cluster, and N preceded by one or two Ns has high probability, which results in a one-bit Huffman codeword. In consequence, one of the inputs of our Huffman-based LUT can be a relatively long run of N symbols, which also means that the minimum pattern length should be relatively large (e.g., over 20), which is obviously undesirable.

Figure 6 shows the impact of the node size B in the B-tree layout on the count times, with varying pattern length. Even $B = 1$ results in a much faster search than with a standard SA (by a factor of 1.7–2.0; cf. also Figure 4) and growing B helps more, up to $B = 32$ (on all the datasets, $B = 64$ is slightly slower). Still, the speed gap between $B = 1$ and $B = 32$ rarely exceeds 10 %. This small improvement may seem disappointing, but is understandable. Using $B > 1$ improves access locality, concerning the suffix offsets, yet the accesses to the text are inevitable and those are generally not cached. This effects flattens all results. Moreover, the top levels of the tree (disregarding the choice of B) are cached across multiple patterns in the test collection.

The relative times of count and locate per found item, as a ratio of the respective results of the SA variant with the B-tree layout and the plain SA, are shown in

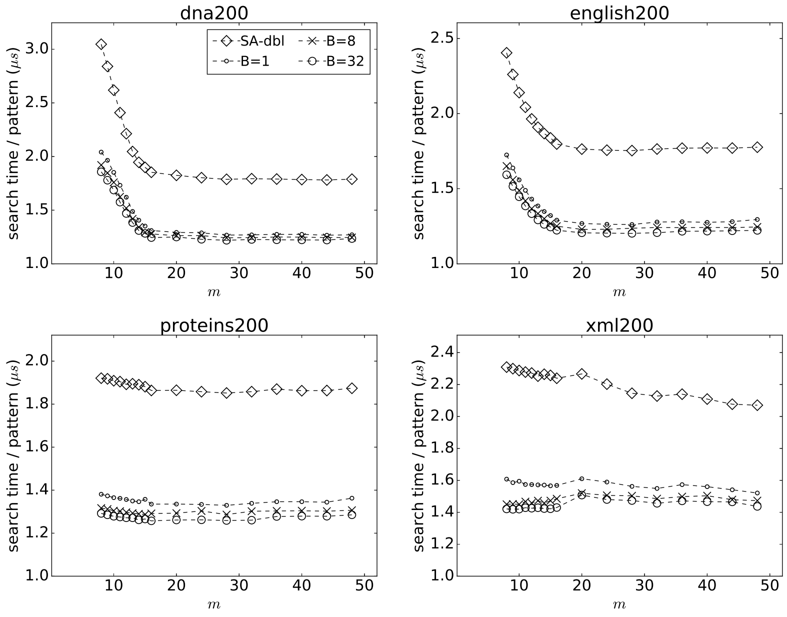


Figure 6. Count times for the standard SA and the SA with the B-tree layout, for selected node sizes B. The right interval boundary in the standard SA variant is found with the doubling technique (-dbl).

Figure 7. The locate operations in the B-tree SA variant suffer from the need to traverse over many layers of the tree, which is relatively more costly when the number of pattern occurrences tends to be larger (i.e., for small m). When m grows and *occ* is often a few (or even one), this overhead is relatively smaller and the cache-friendliness of the layout more than compensates the extra operations. The count operation, after the boundary suffixes are found, is also more costly in the B-tree layout, but the extra cost (logarithmic in N, as opposed to constant for the standard SA) is computational, without extra accesses (and thus resulting cache misses) to data.

In Figure 8 we show how augmenting the SA with various structures reducing the initial search interval affects the query times and the used space. The hash table (HT), based on the xxhash function (`https://github.com/Cyan4973/xxHash`), stores 8-grams from the text and was tried with two load factors (LF); the solution is called the SA-hash index in the original works [15, 16]. LUT2 gives a significant boost in a tiny space, yet it is the hash table (LF = 0.9) that excels here, speeding up the baseline variant by a factor of 1.5–2.

As our SAs with the B-tree layout can be augmented with prefixes of the suffixes visited in the first steps of the traversal of the tree (i.e., in the top levels), we test the impact of the prefix length on the performance and space of the resulting index

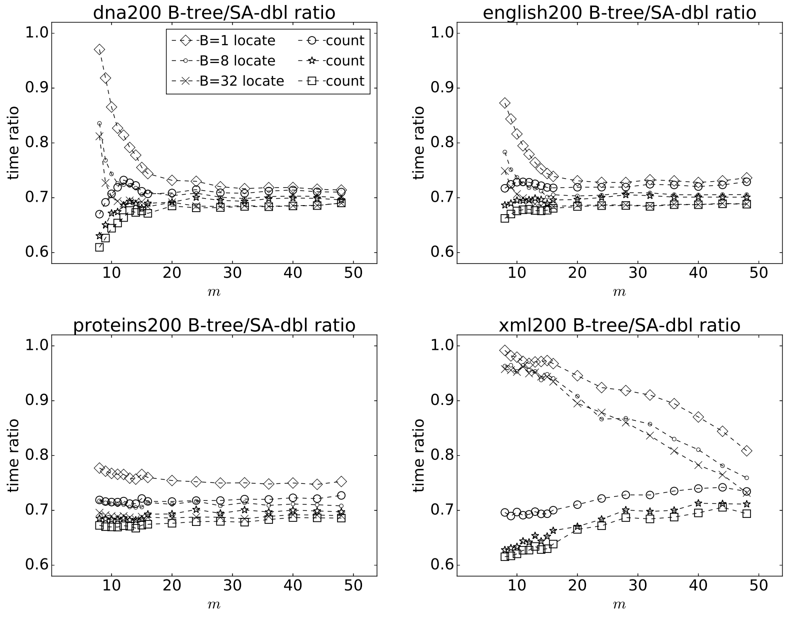


Figure 7. Count and locate relative times for the SA variant with the B-tree layout and the standard SA, for selected node sizes B

(Figure 9). Adding the prefixes gives a noticeable speedup even if they are limited in length to 8 characters and are attached to a few tree levels only, while the space overhead is rather small. Longer prefixes and more levels help less for a much bigger space penalty. Combinations of all ideas presented in the paper are shown in Figure 10, where the best option is to combine the B-tree layout with LUT/HT (adding prefixes on top of it has a negligible effect). In total, the speed of the standard SA with the standard right interval boundary search was usually improved by a factor exceeding 3 (from 2.6 for `xml200` to 3.9 for `dna200`, for $m = 24$).

In the last experiment we used the SA-hash index in a variant with varying the length of the hashed prefix. Different lines (space-time tradeoffs) are obtained with varying the parameters k_1 and k_2 (Figure 11). The key lines to compare are denoted as "SA-dbl & HT" (which is SA-hash with the doubling technique for finding the right interval boundary) and "SA-dbl & HT-var-k" (which is the new variant).

Table 2 presents some details, grouped in pairs of rows. The value of k (prefix length) used in the SA-hash algorithm is presented in the top rows. The parameters k_1 and k_2, and r, the interval width threshold, found in a learning procedure, are presented in the bottom rows. Clearly, r for the given dataset and the parameters k, k_1, k_2, is such to make the sizes of the compared structures possibly equal, as the next column demonstrates. More precisely, we find r in the following way. Let s_1

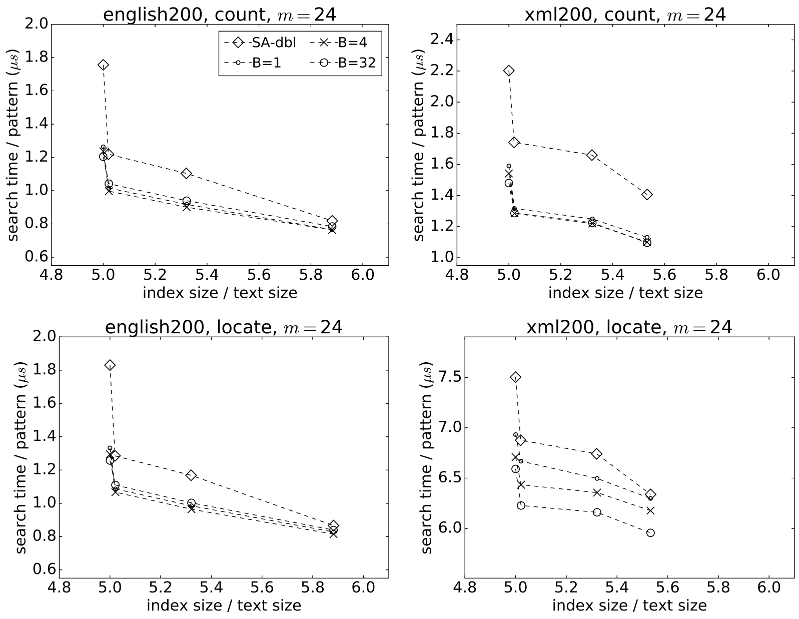


Figure 8. Augmenting the suffix array. The right interval boundary in the standard SA variant is found with the doubling technique (-dbl). The four points in each series correspond to: no extra data, LUT on 19 bit and 23 bits, respectively, using order-1 Huffman coding, and HT with LF = 0.9 and $k = 8$.

be the size of the SA-hash index for a given k. For fixed (arbitrarily chosen) values of k_1 and k_2, such that $k_1 < k$ and $k_2 > k$, we binary search for r in a way to obtain the size of the HT-var-k index not greater than s_1, but possibly close to it.

However, we see in Figure 11 that the net result of our efforts is mixed. In many cases the new variant is able to achieve a slight improvement in speed using

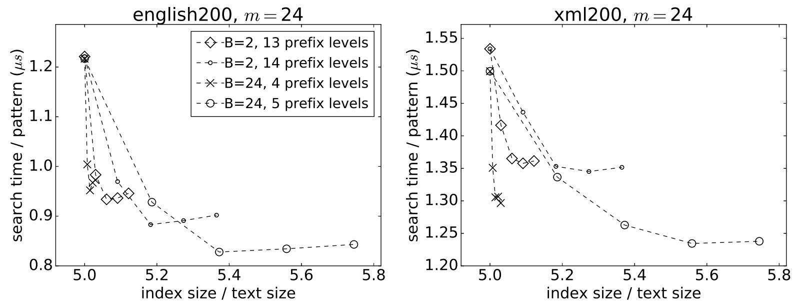


Figure 9. Index sizes and count times for the SA with the B-tree layout, when several top levels of the tree store the corresponding suffixes' prefixes of length $\{0, 4, 8, 12, 16\}$

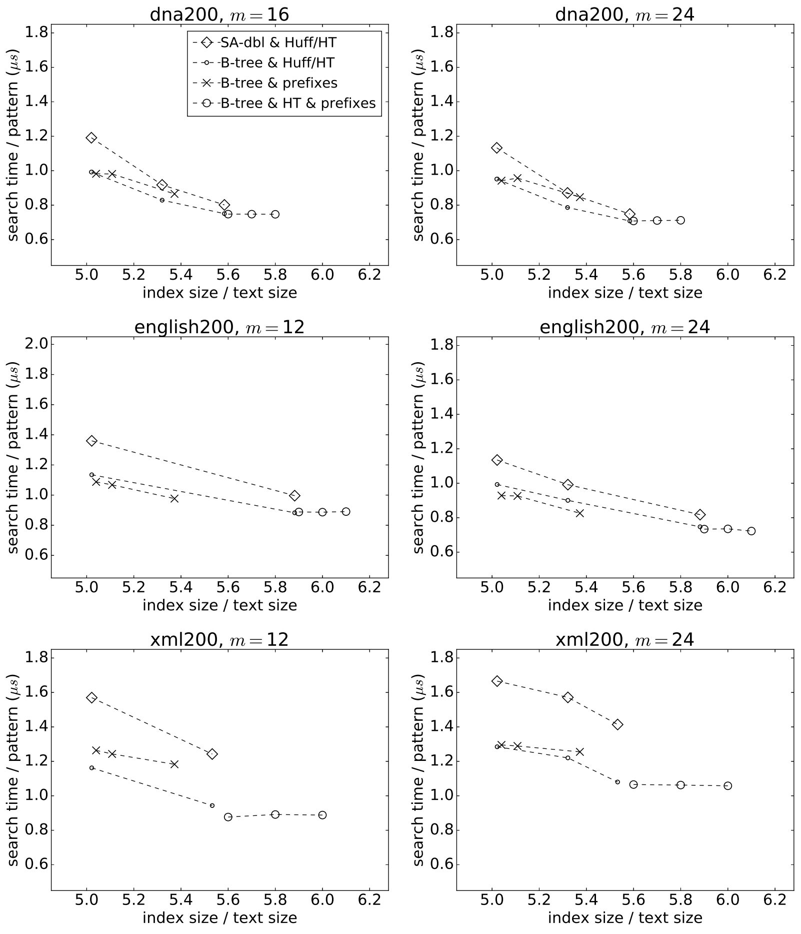


Figure 10. Index sizes and count times for several SA variants with different layouts and extra data. Successive points in the series are obtained by changing the LUT or hash table component and/or using the prefix copies on varying number of levels in the tree.

the same space, with (sometimes) queries faster by more than 10 % for the `xml200` dataset. On the `proteins200` dataset, however, using k_1 and k_2 instead of a single value of k makes the queries slower by more than 10 %.

4 CONCLUSION

Algorithm engineering not once revitalizes old ideas and data structures. In this work, we attempted to improve the performance of the classic full-text index, the suffix array. Our work focused on the navigation over the index, changes to its layout

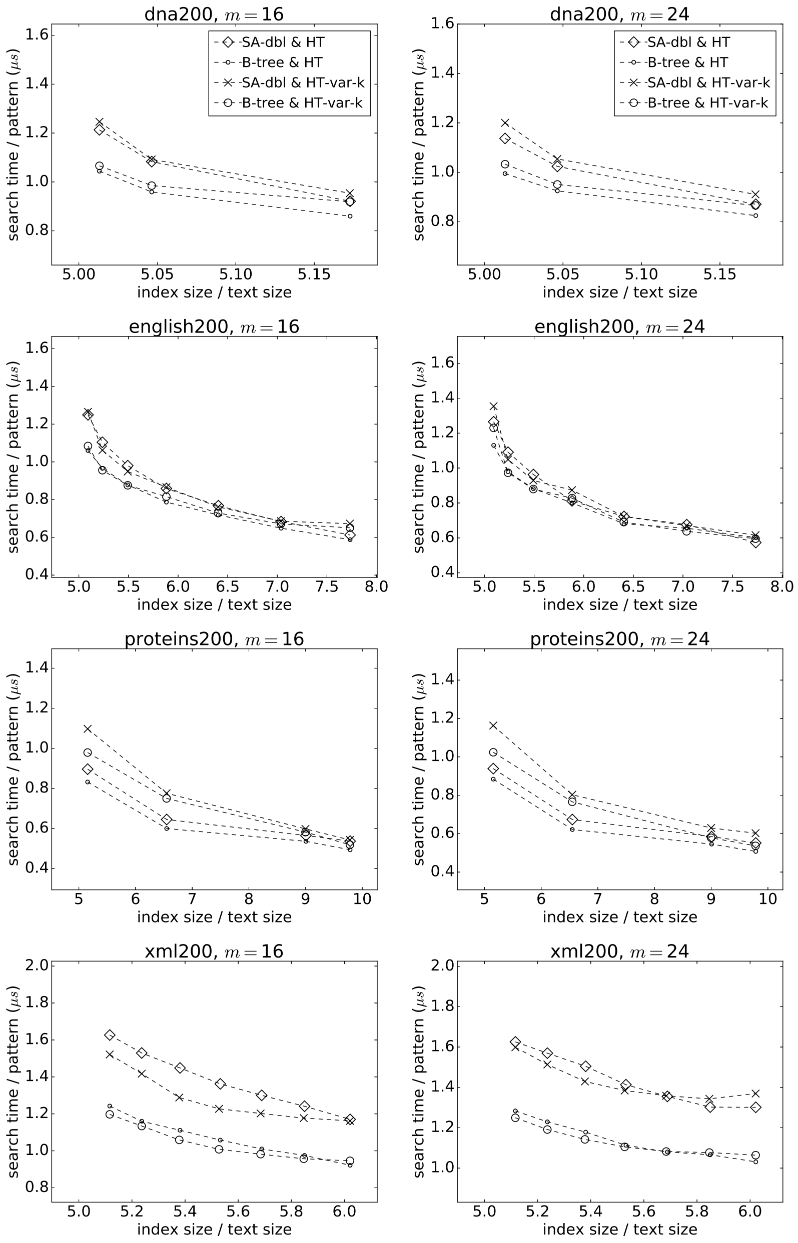


Figure 11. Index sizes and count times for the SA-hash index with two hashed prefix lengths (k_1 and k_2)

Dataset	k	k_1	k_2	r	Total Size	Filter Size	avg $\log_2$(ival_width)
dna200	12	–	–	–	5.584	–	5.256 ± 2.333
	–	10	16	964	5.584	0.006	7.601 ± 2.528
english200	8	–	–	–	5.882	–	7.045 ± 3.797
		6	12	3726	5.882	0.030	7.115 ± 3.695
proteins200	6	–	–	–	6.549	–	4.056 ± 2.801
		4	10	2989	6.549	0.001	7.802 ± 4.277
xml200	8	–	–	–	5.532	–	12.102 ± 5.942
		6	12	867	5.528	0.032	10.925 ± 5.539

Table 2. Index sizes for four 200 MB Pizza & Chili datasets. The used parameters are: k for the standard SA-hash index and k_1, k_2 for the SA-hash index with variable prefix lengths. The parameter r, the interval width threshold, was set in a way to have the two index sizes possibly close to each other. The rightmost column presents the average binary logarithms (with their standard deviations) of the search interval widths.

and augmenting the index with extra data. The experiments show that generally the best option is to combine a B-tree layout of the suffix with a lookup table or a hash table (reducing the interval of suffixes for further binary or k-ary search), with a speedup over the standard SA configuration by a factor usually exceeding 3, for a relatively small penalty in the space.

Acknowledgement

The work was supported by the Polish National Science Centre under the project DEC-2013/09/B/ST6/03117 (the first two authors).

REFERENCES

[1] Manber, U.—Myers, G.: Suffix Arrays: A New Method for On-Line String Searches. SIAM Journal on Computing, Vol. 22, 1993, No. 5, pp. 935–948, doi: 10.1137/0222058.

[2] Ferragina, P.—González, R.—Navarro, G.—Venturini, R.: Compressed Text Indexes: From Theory to Practice. Journal of Experimental Algorithmics, Vol. 13, 2009, Art. No. 12, doi: 10.1145/1412228.1455268.

[3] Weiner, P.: Linear Pattern Matching Algorithms. Proceedings of the 14th Annual IEEE Symposium on Switching and Automata Theory (swat 1973), Washington, DC, 1973, pp. 1–11, doi: 10.1109/swat.1973.13.

[4] Farach, M.: Optimal Suffix Tree Construction with Large Alphabets. Proceedings of the 38th IEEE Annual Symposium on Foundations of Computer Science (FOCS '97), 1997, pp. 137–143, doi: 10.1109/sfcs.1997.646102.

[5] Grimsmo, N.: On Performance and Cache Effects in Substring Indexes. Technical Report IDI-TR-2007-04, Norwegian University of Science and Technology, Trondheim, Norway, 2007.

[6] Cole, R.—Kopelowitz, T.—Lewenstein, M.: Suffix Trays and Suffix Trists: Structures for Faster Text Indexing. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (Eds.): Automata, Languages and Programming (ICALP 2006). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4051, 2006, pp. 358–369, doi: 10.1007/11786986_32.

[7] Fischer, J.—Gawrychowski, P.: Alphabet-Dependent String Searching with Wexponential Search Trees. In: Cicalese, F., Porat, E., Vaccaro, U. (Eds.): Combinatorial Pattern Matching (CPM 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9133, 2015, pp. 160–171, doi: 10.1007/978-3-319-19929-0_14.

[8] Munro, J. I.—Navarro, G.—Nekrich, Y.: Fast Compressed Self-Indexes with Deterministic Linear-Time Construction. Proceedings of the 28th International Symposium on Algorithms and Computation (ISAAC 2017), Leibniz International Proceedings in Informatics (LIPIcs), Vol. 92, 2017, Art. No. 57, 12 pp., doi: 10.4230/LIPIcs.ISAAC.2017.57.

[9] Bille, P.—Gørtz, I. L.—Skjoldjensen, F. R.: Deterministic Indexing for Packed Strings. Proceedings of the 28th Annual Symposium on Combinatorial Pattern Matching (CPM 2017), Leibniz International Proceedings in Informatics (LIPIcs), Vol. 78, 2017, Art. No. 6, 11 pp., doi: 10.4230/LIPIcs.CPM.2017.6.

[10] Willard, D. E.: Searching Unindexed and Nonuniformly Generated Files in $\log\log N$ Time. SIAM Journal on Computing, Vol. 14, 1985, No. 4, pp. 1013–1029, doi: 10.1137/0214071.

[11] Zhou, J.–Ross, K. A.: Implementing Database Operations Using SIMD Instructions. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data (SIGMOD '02), ACM, 2002, pp. 145–156, doi: 10.1145/564691.564709.

[12] Schlegel, B.—Gemulla, R.—Lehner, W.: K-ary Search on Modern Processors. Proceedings of the Fifth International Workshop on Data Management on New Hardware (DaMoN '09), 2009, pp. 52–60, doi: 10.1145/1565694.1565705.

[13] Khuong, P.-V.—Morin, P.: Array Layouts for Comparison-Based Searching. Journal of Experimental Algorithmics, ACM, Vol. 22, 2017, No. 1, Art. No. 1.3, doi: 10.1145/3053370.

[14] Williams, J.: Algorithm 232: Heapsort. Communications of the ACM, Vol. 7, 1964, No. 6, pp. 347–348.

[15] Grabowski, S.—Raniszewski, M.: Two Simple Full-Text Indexes Based on the Suffix Array. In: Holub, J., Žďárek, J. (Eds.): Proceedings of the Prague Stringology Conference (PSC), 2014, pp. 179–191.

[16] Grabowski, S.—Raniszewski, M.: Compact and Hash Based Variants of the Suffix Array. Bulletin of the Polish Academy of Sciences – Technical Sciences, Vol. 65, 2017, No. 4, pp. 407–418, doi: 10.1515/bpasts-2017-0046.

Tomasz Marek Kowalski received his computer science M.Sc. and Ph.D. degrees from Łódź University of Technology in 2004 and 2008, respectively. His research interests involve object systems, computer languages and indexing techniques for various searching problems (string matching, bioinformatics, etc.). He has published over 60 papers in journals and conferences. He is currently Assistant Professor at the Institute of Applied Computer Science of the Łódź University of Technology.

Szymon Grabowski received his M.Sc. degree from the University of Łódź in 1996, Ph.D. degree from the AGH University of Science and Technology in Cracow in 2003, and the habilitation degree at the Systems Research Institute of the Polish Academy of Sciences in Warsaw in 2011. His former research, including Ph.D. dissertation, involved nearest neighbor classification methods in pattern recognition, also with applications in image processing. Currently, his main interests are focused on string matching and text indexing algorithms, and data compression. Some of his particular research topics include various approximate string matching problems, compressed text indexes, and XML compression. He has published over 120 papers in journals and conferences. He is currently Professor at the Institute of Applied Computer Science of Łódź University of Technology.

Kimmo Fredriksson received his computer science M.Sc. and Ph.D. degrees from University of Helsinki in 1997 and 2001, respectively. His research interests include wide variety of string matching problems as well as indexing techniques for searching in metric spaces. He has published about 60 papers on these topics in international conferences and journals. He has the position of Adjunct Professor at the University of Eastern Finland.

Computing and Informatics, Vol. 38, 2019, 575–596, doi: 10.31577/cai_2019_3_575

DIALOGUE MODEL USING ARGUMENTS FOR CONSENSUS DECISION MAKING THROUGH COMMON KNOWLEDGE FORMATION

Ayslan Trevizan Possebom

Federal Institute of Paraná – Paranavaí
1400, José Felipe Tequinha St.
87703-536 Paranavaí-PR, Brazil
e-mail: possebom@gmail.com

Mariela Morveli-Espinoza, Cesar Augusto Tacla

Federal University of Technology of Paraná
Graduate Program in Electrical and Computer Engineering
3165, 7 September Ave.
80230-901 Curitiba-PR, Brazil
e-mail: morveli.espinoza@gmail.com, tacla@utfpr.edu.br

Abstract. Argumentation plays an important role in reasoning and allows the justification of opinions, especially when applied to collaborative decision making. Reaching consensus is not a trivial task where arguments exchanged in a dialogue and common knowledge are important for consensus. This paper presents a model of argumentative dialogue to support the formation of common knowledge in a group of agents that communicate by sending arguments, and proposes a semantics for consensus decision making. The output of the model is a weighted argumentation graph in which semantics is used to decide the preference of the group.

Keywords: Structured logical arguments, consensus, dialogue, decision making, common knowledge, possible worlds

Mathematics Subject Classification 2010: 68T42

1 INTRODUCTION

Reaching consensus about an issue through discussion with a group of people is not a trivial task [15, 24]. When applied to a group of intelligent agents, this task becomes even more complex, since agents must reason about logically related instructions [1]. In order for consensus decision making in a group to take place, it is necessary to identify the consensus level (or level of acceptance) of the group with regard to the available decision alternatives. The stronger the justifications for supporting or rejecting a particular decision alternative, and the greater the consensus of the group on these justifications, the closer the situation will be to a decision by consensus [25]. The choice of a decision alternative by consensus does not reflect the optimal decision, but rather the one that is preferred by most of the agents.

Consensus is directly related to common knowledge. Agreement on a decision implies common knowledge, and this becomes a prerequisite when a group of agents try to make decisions together [12, 20]. Common knowledge occurs when all agents know an item of information and also know that the other agents in the group know that information [16].

Several methods for group decision making have been proposed in the literature [15, 24], including majority voting, auctions, Borda, Condorcet, and judgment aggregations, among others. These methods do not assume a dialogue between the group members, and each participant only votes on or gives a preference relation for the set of possible alternatives, expressing neither the reasons for these votes nor the conditions for opinion formation. Thus, dialogue becomes an important step before voting, in which all participants can express their opinions and arguments, defending or attacking the alternatives or the information in other arguments presented by other participants. Furthermore, through this dialogue, agents can change their way of thinking based on the arguments presented.

The use of argumentation in multi-agent systems has received a great deal of attention in the last decade. Building arguments allows the agents to reach a collective agreement that is consistent with their beliefs and goals [6, 22]. For a collective decision to be close to unanimity, we need to identify the consensus level of the group on the information in the arguments sent during the dialogue, analysing both the supporting and rejecting relations in each element of the inner structure of those arguments. The information in an argument that is supported (or accepted) by most agents should be consented to by the other agents in the group who do not know it or who reject it. Thus, the supported information becomes consensually accepted by the group in relation to the issue under discussion.

In this paper, we propose a model of dialogue that uses arguments in the messages sent by the agents. Through the information in these arguments, common knowledge is formed based on the majority knowledge of the group. The innovations and contributions of this work are as follows:

1. development of a process of common knowledge identification;

2. identification of the relation between common knowledge and consensus; and
3. development of semantics for consensus decision making.

Thus, the paper has two goals:

1. To present a dialogue model that can be applied to multi-agent systems, where each argument presented needs to be evaluated by the group in an attempt to identify the consensus level on each piece of information presented in the arguments. As a result of this model, common knowledge is formed about the set of information that was accepted or rejected by the group of agents. This common knowledge formation can be used in several application domains where multi-agent systems are used, such as chatbots, sensor networks, ranking of the importance of web pages, identification of simultaneous actions, or any domain in which there is a need for the formation of group opinion.
2. To generate a weighted argumentation graph [2, 5] for each dialogue, so that the decision alternatives can be analysed based on the arguments presented, resulting in a preference relation for the group. The preference relation draws on computation of the strength of an argument, and uses a semantics that considers all the weights of the arguments to determine the preference level for each alternative.

This paper is structured as follows. Section 2 presents the preliminary concepts of possible worlds, common knowledge and structured arguments. Section 3 describes the proposed model of dialogue, covering the structures of the agents, the construction of arguments, the formation of common knowledge, and the consensus decision-making process, including the argument strength, the weighted argumentation graph and semantics for determining the preference relation among the decision alternatives. A practical example, a discussion of the results, and related work are given in Section 4. Finally, we present the conclusions and an outline of planned future work in Section 5.

2 PRELIMINARIES

In this section, we introduce the fundamental background of knowledge representation related to possible worlds and common knowledge, which we use to represent the possible decision alternatives (or issues) that are the subject of dialogue among a group of agents. We also describe the fundamental concepts of arguments and attack relations between arguments, which form the basic structure used by the agents to send messages to the group during the dialogue.

The model proposed in this work considers a virtual environment where each agent in a group, each one with its own knowledge base, is able to act sending messages to the group in a discussion (dialogue) about any issue using logically structured arguments. We model a dynamic process of dialogue that can be used by the agents for choosing the alternative that is consensually justified or for obtaining the order of preference over the available decision alternatives.

2.1 Possible Worlds and Knowledge Representation

The classical model of reasoning about knowledge, as used by a single agent, is known as the possible world model [17]. Possible worlds represent a possible state of affairs (that is, there may be situations in which a belief holds for one issue under discussion, but does not hold for another issue) [12]. Let $AG = \{ag_1, \dots, ag_n\}$ with $n > 0$ be a finite set of agents. An agent $ag_i \in AG$ believes f if f is necessarily true for ag_i, i.e. it is true in all possible worlds for that agent. The modal operator K_i represents the knowledge of agent ag_i. The formula $K_1 f$ is read as "agent ag_1 knows f", $K_1 K_2 f$ is read as "agent ag_1 knows that ag_2 knows f" and $\neg K_2 K_1 K_3 f$ is read as "agent ag_2 does not know that ag_1 knows that ag_3 knows f".

When the reasoning involves the knowledge of a set of agents, two modal operators can be defined [13]: E_{AG} (where $E_{AG} f$ represents the situation in which every agent in the group knows f) and C_{AG} (where $C_{AG} f$ represents the situation in which every agent in the group knows f and they all know that every agent in the group knows f, i.e. f is common knowledge in the group).

2.2 Structured Logical Arguments

The basis of the proposed dialogue model is the exchange of arguments among agents. When an agent sends a message to the group containing an argument, this argument represents the agent's opinion of, point of view on or justification for the issue under discussion. In this paper, Σ is a knowledge base with formulae (beliefs) in a propositional language, and the arguments are built based on these formulae [4]. In addition, $\vdash$ is the classical inference, $\equiv$ represents logical equivalence, $\perp$ represents contradiction, $\wedge$ conjunction, $\vee$ disjunction, $\neg$ negation, $\rightarrow$ implication, and $\leftrightarrow$ biconditionality. An argument [3, 4, 21] is formed by a pair $\langle \Phi, \alpha \rangle$ where Φ represents the support (premises) and α the claim of the argument, such that

1. α is a formula;
2. $\Phi \subseteq \Sigma$;
3. $\Phi \nvdash \perp$;
4. $\Phi \vdash \alpha$; and
5. $\nexists \Phi' \subseteq \Phi$ such that $\Phi' \vdash \alpha$.

Arguments are created to justify a position against the decision alternative or other arguments. The most common attack relations between arguments are undercut and rebuttal [4, 21]. Let $arg_1 = \langle \Phi_1, \alpha_1 \rangle$ and $arg_2 = \langle \Phi_2, \alpha_2 \rangle$ be two distinct arguments: arg_1 undercuts arg_2 iff $\exists \varphi \in \Phi_2$ such that $\alpha_1 \equiv \neg\varphi$, and arg_1 rebuts arg_2 iff $\alpha_1 \equiv \neg\alpha_2$.

Example 1. Let $\Sigma = \{a, \neg b, a \rightarrow \neg b, d \rightarrow b, a \rightarrow d\}$ be a knowledge base of an agent. Let us consider the following three arguments $arg_1 = \langle \{a, a \rightarrow \neg b\}, \neg b \rangle$, $arg_2 = \langle \{\neg b, d \rightarrow b, a \rightarrow d\}, \neg a \rangle$, and $arg_3 = \langle \{a, a \rightarrow d, d \rightarrow b\}, b \rangle$. We have that arg_2 undercuts arg_1 and arg_3 rebuts arg_1. Figure 1 illustrates these attack relations.

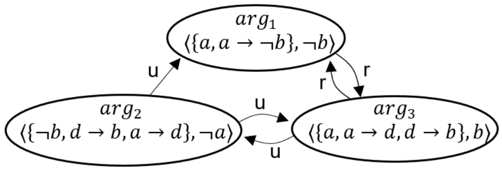

arg_1
$\langle\{a, a \to \neg b\}, \neg b\rangle$
arg_2
$\langle\{\neg b, d \to b, a \to d\}, \neg a\rangle$
arg_3
$\langle\{a, a \to d, d \to b\}, b\rangle$
u
u
u
r
r

Figure 1. Types of attack relation: undercut (u) and rebuttal (r)

3 COMMON KNOWLEDGE AND CONSENSUS DECISION MAKING

The proposed dialogue model using arguments for common knowledge formation is constructed using agents that play two roles: argumentation and mediation. Argumentative agents are responsible for building arguments and voting, using the beliefs in their respective knowledge bases. Each belief consists of a formula in propositional language. Argumentative agents are also responsible for giving their opinions by voting to support or reject the formulae in arguments sent by the other argumentative agents during the dialogue. The mediator agent is responsible for several other aspects: controlling the message exchange among the argumentative agents, that is, controlling the course of the dialogue; calculating the consensus level on each formula within an argument; informing the group of argumentative agents of which formulae should be accepted during the dialogue; computing the strength of each argument; and informing the group of the outcome of the decision.

For the argumentative and mediator agents, we define a model for common knowledge formation (CKF).

Definition 1. A model for common knowledge formation is a tuple $CKF = \langle AG, EX, ISS, med, t, \sigma\rangle$ where:

- $AG = \{ag_1, \ldots, ag_n\}$ with $n > 1$ is the finite set of argumentative agents;
- $EX = \{ex_1, \ldots, ex_n\}$ with $ex_i \in [0, 1]$ and $\sum_{i=1}^{n} ex_i = 1$ is the set of expertise values for the argumentative agents, such that ag_i has expertise ex_i;
- $ISS = \{iss_1, \ldots, iss_m\}$ with $m > 1$ is the finite set of issues (or decision alternatives) to be discussed;
- med is the mediator agent;
- t is the waiting time used by the mediator to coordinate the message exchange; and
- σ is a threshold value determining when a formula is common knowledge.

3.1 Argumentative Agent

Argumentative agents are responsible for building arguments, and for supporting or rejecting any information used in an argument.

Definition 2. Let AG be the set of argumentative agents. An argumentative agent $ag_i \in AG$ is a tuple $\langle \Sigma_i, S_i, A_i \rangle$, where:

- $\Sigma_i = K_i \cup KO_i$ is the knowledge base, with K_i representing the beliefs that the agent has about the environment, and KO_i representing the beliefs acquired through communication with other agents;
- S_i is the argument base, which is used to store the set of arguments to be sent to the group; and
- A_i is the current argument that is being discussed by the group, which is used to look for counterarguments.

A formula $f \in \Sigma_i$ may be followed by a label, such as $f[iss_i(b), \ldots]$ where $iss_i \in ISS$ is an issue and $b \in [0,1]$ is the group consensus level on f related to the issue iss_i. Whenever $b > 0$, most of the agents in AG believe f, and this formula is accepted as common knowledge. For $b = 0$, formula f is not accepted by the group as common knowledge since it is rejected or because most agents do not know f.

Example 2. [adapted from [3]] Consider a set of argumentative agents AG deciding whether or not a patient should undergo surgery. Let Bob, $ag_{bob} \in AG$, be an argumentative agent representing a doctor. The formulae in the knowledge base represent the following information: the patient should undergo a surgery (sg), the patient has colonic polyps (a), the patient is at risk of loss of life (b), the patient is experiencing side-effects (c), the patient has cancer (d). Its structure is defined by:

$$\begin{aligned}
K_{bob} &= \{a[sg(1)], \neg c \rightarrow \neg sg[sg(0)], c \rightarrow sg, d \rightarrow b, a \rightarrow d, d \wedge \neg b \rightarrow sg\},\\
KO_{bob} &= \{\neg b[sg(0.7)], \neg c[sg(0.4)]\},\\
S_{bob} &= \{\langle \{\neg c, \neg c \rightarrow \neg sg\}, \neg sg \rangle, \langle \{a, a \rightarrow d, d \rightarrow b\}, b \rangle\},\\
A_{bob} &= \{\langle \{a, a \rightarrow \neg b\}, \neg b \rangle\}.
\end{aligned}$$

The formulae $\neg b$ and $\neg c$ were accepted by the group, and therefore, were considered by ag_{bob} and used to update its KO_{bob} base. Formula a was accepted unanimously, while $\neg c \rightarrow \neg sg$ was not accepted by the group in the dialogue about sg. When ag_{bob} has the opportunity to send arguments, both arguments in S_{bob} will be sent to the group. A_{bob} stores the current argument in the discussion, and this is used as a reference to look for other counterarguments, storing them in S_{bob} when requested.

Let ARG_i be the set of all arguments that can be built from Σ_i and $arg \in ARG_i$ be an argument with $arg = \langle \Phi, \alpha \rangle$. The function $premise(arg)$ returns Φ (a set of formulae in support of arg) and $claim(arg)$ returns α (the formula in the claim of arg). Let F be the set of all formulae in arg obtained from the function $split(arg)$ ($premise(arg) \cup claim(arg)$). Each formula $f \in F$ has a set of atoms obtained from the function $atoms(f)$. These functions are used by the argumentative agents to express support for or rejection of each formula in the argument that is presented to the group in the dialogue.

A formula f in any argument is supported by an argumentative agent when it knows that formula. The argumentative agent rejects f when it knows other formulae with the same atoms, but with no equivalent meaning.

Definition 3. A formula f in an argument arg_1 sent by an agent ag_i is supported by ag_j, with $i \neq j$, iff (i) $\exists arg_2 \in ARG_j | claim(arg_2) \leftrightarrow f$ is a tautology or (ii) $\exists g \in \Sigma_j | atoms(g) = atoms(f)$ and $\mu \leftrightarrow f$ is a tautology. Formula f is rejected iff (i) $\exists arg_2 \in ARG_j | claim(arg_2) \leftrightarrow \neg f$ is a tautology or (ii) $\exists g \in \Sigma_j | atoms(g) = atoms(f)$ and $g \leftrightarrow f$ is not a tautology.

From Definition 3, we can observe that the agent ag_j supports a formula f of an argument sent by ag_i if ag_j has an argument for f (i.e. $\langle\{\Phi\}, f\rangle \in ARG_j$), or if ag_j knows f (i.e. $f \in \Sigma_j$). A rejection occurs when ag_j has an argument for $\neg f$ (i.e. $\langle\{\Phi\}, \neg f\rangle \in ARG_j$), or if ag_j knows a formula g that has the same atoms as f, but g and f are not logically equivalent.

Example 3. [cont. 2] Consider $arg = \langle\{a, a \rightarrow \neg d\}, \neg d\rangle$, which will be analyzed to identify the consensus level on its formulae. Agent ag_{bob} supports a because it believes this formula. Formula $a \rightarrow \neg d$ is rejected because ag_{bob} believes $a \rightarrow d$ and $(a \rightarrow \neg d) \leftrightarrow (a \rightarrow d)$ is not a tautology. Formula $\neg d$ is supported by ag_{bob} because it has an argument for $\neg d$: $\langle\{\neg b, d \rightarrow b\}, \neg d\rangle$ and rejected with argument $\langle\{a, a \rightarrow d\}, d\rangle$.

The possible actions available to all the argumentative agents $ag_i \in AG$ in an argumentative dialogue for a decision by consensus are as follows:

- *discArg*(arg, y): the current argument arg presented in the dialogue is stored in A_i of the argumentative agents along with a number y that denotes its position in the sequence in which the argument was sent to the group;
- *askSpeak*$()$: when $S_i \neq \emptyset$, ag_i informs med that it has some arguments to be sent;
- *propose*$()$: when ag_i is requested to send its arguments, it sends all the arguments in S_i to med and then S_i is emptied;
- *attack*(t): ag_i looks for arguments attacking the argument in A_i, storing them in S_i, in time t;
- *voteSupport*(f, t): ag_i votes to support formula f at time t;
- *voteRejection*(f, t): ag_i votes to reject formula f at time t;
- *learn*(f, b, iss): ag_i updates Σ_i with formula f and the label containing the consensus level b for issue iss. If $f \in \Sigma_i$, then the action $inform(f, b, iss)$ is executed; otherwise, formula $f[iss(b)]$ should be inserted in KO_i;
- *inform*(f, b, iss): agents with $f \in \Sigma_i$ should update f with the label $f[iss(b)]$ only when issue iss is not yet annotated in f;

- $query(ag_j, at)$: agent ag_i can ask ag_j with $i \neq j$ for formulae containing the atom at. This action does not involve the *med* agent and can be performed at any time by the argumentative agents;
- $answer(ag_i, F)$: when an agent ag_j is queried, it replies to ag_i, returning the set of formulae F containing the atom at.

3.2 Mediator Agent

There is a dialogue for each decision alternative and the mediator agent maintains a dialogue table for each dialogue. This table is used to store the sequence of arguments received during the dialogue, the attacks on the arguments and information about the consensus regarding each argument.

Definition 4. The mediator agent *med* is a tuple $\langle WB, AGENDA, DT, \delta\rangle$, where:

- WB is an ordered list of argumentative agents;
- $AGENDA$ is a list that stores all the arguments sent by one agent when requested;
- $DT = \{dt_1, \ldots, dt_m\}$ is the set of dialogue tables where each $dt_i \in DT$ has the arguments sent during the dialogue on the issue iss_i;
- δ is the knowledge base that stores all the formulae of the arguments within a dialogue, with their respective annotations.

WB is used by *med* as a coordination resource that emulates a face-to-face meeting. When an argumentative agent has arguments to send to the group, it asks to speak (action *askSpeak*()) to *med* and waits for a request to send the arguments. This resource ensures that only agents in WB are granted the right to speak. Only the agent at the top of the list at any given moment is able to send its arguments when requested.

When *med* requests the arguments of an argumentative agent, all the arguments received (action *propose*()) are stored in $AGENDA$. Each argument needs to be checked; that is, *med* checks whether the arguments are admitted and whether an argument has already been presented in the current dialogue.

Definition 5. An argument is admitted iff its formulae in Φ are accepted for the current dialogue. A formula f is accepted for the issue iss_i if it does not mention another issue: $\forall atom(f) \notin ISS \setminus \{iss_i\}$. Furthermore, it must satisfy one of the following conditions:

1. it has not been presented in any other arguments in the current dialogue (formula without a label for iss_i); or
2. there is a consensus on it (label $iss_i(b)$ with $b > 0$).

Example 4. [cont. 2] In a dialogue about sg, the argument $\langle\{a[sg(1)], a \rightarrow d\}, d\rangle$ is admitted. There is consensus on a, and $a \rightarrow d$ has not been presented earlier in any argument about the issue sg. On the other hand, the argument $\langle\{\neg c[sg(0.4)], \neg c \rightarrow \neg sg[sg(0)]\}, \neg sg\rangle$ is not admitted. Although there is consensus on $\neg c$, there is no consensus on $\neg c \rightarrow \neg sg$.

Each admitted argument within the $AGENDA$ occupies a row in the current dialogue table. The dialogue table has the following fields: y (a sequential number indicating the sequence in which the arguments are presented), the issuer agent, the admitted argument, the argument being attacked, the sets of supporting and rejecting agents for each formula of the argument, the set of consensus levels for each formula of the argument, and the intrinsic strength of the argument.

After the support and rejection steps (actions *voteSupport* and *voteRejection*), agent *med* computes the consensus level of f and stores it with the corresponding label in δ, informing the group of argumentative agents of whether or not f should be accepted as common knowledge.

Algorithm 1 shows the dialogue model for CKF executed by *med*. Firstly, the structure for a new dialogue is created (line 2, function *newDialogue*(*iss*)) involving the creation of *WB*, $AGENDA$, and the start point for the dialogue $arg = \langle\{T\}, iss\rangle$ is returned (a structure with only the claim to be discussed). The dialogue table for the issue *iss* is initialized and the current line in the table is returned by *updateDT*(*med*, *arg*) (line 3). Then, *med* sends *arg* to the group (line 4) and asks for support and rejection of formula *iss* at time t (lines 5–6). The dialogue table is updated, including the list of agents that supported and rejected formula *iss*, the consensus level and the intrinsic strength for the start point (lines 7–8, functions buf(*iss*) and $is(arg)$). The argumentative agents look for counterarguments at time t (line 9). When *med* requests agents with arguments to send, the responses are stored in *agents* (line 10) and *WB* is updated (line 11, function *updateWB*(*agents*)). For each agent in *WB*, the agent in the first position of the queue sends its arguments (line 13, function *requestArgs*(ag_i)), and all arguments received are stored in the $AGENDA$ (line 14, function *updateAGENDA*(*argsList*)). Each argument is checked (line 16, function *check*(arg_k)). Only the admitted arguments are stored in *dt* for the current dialogue (line 17), and the group is informed (line 18). Each formula of the argument undergoes a voting process considering a time t (lines 20–21) and receives a consensus level (line 22); the current dialogue table is updated with the agreement and rejection lists and the consensus level for the formula under analysis (line 23); the knowledge base of *med* is updated with the formula and its related label (line 24); *med* informs the group of whether or not the formula should be accepted (lines 25–28); the intrinsic strength is updated in the dialogue table (line 29); and *med* asks the group to look for counterarguments, waiting a time t before asking which agents have arguments to send (lines 30–32). The current dialogue ends when *WB* and $AGENDA$ are empty.

The *buf*(f) function acts as a belief update function. It is responsible for computing the consensus level of the group on formula f in an argument, determining which formulae should be accepted as common knowledge. Equation (1) shows how this function is obtained. We refer to ex_i as the expertise value of the agent that

Algorithm 1 Dialogue model for common knowledge formation

```
    Input: the issue to be discussed (decision alternative)
    Output: the dialogue table for the issue
 1: procedure DIALOGUE($iss$)
 2:     $arg \leftarrow newDialogue(iss)$
 3:     $y \leftarrow updateDT(med, arg)$
 4:     $action(discArg(arg, y))$
 5:     $support \leftarrow action(voteSupport(iss, t))$
 6:     $reject \leftarrow action(voteRejection(iss, t))$
 7:     $b \leftarrow buf(iss)$
 8:     $updateDT(y, iss, support, reject, b, is(arg))$
 9:     $action(attack(t))$
10:     $agents \leftarrow action(askSpeak())$
11:     $updateWB(agents)$
12:     for all $ag_i \in WB$ do
13:         $argsList \leftarrow requestArgs(ag_i)$
14:         $updateAGENDA(argsList)$
15:         for all $arg_k \in AGENDA$ do
16:             if $check(arg_k)$ then
17:                 $y \leftarrow updateDT(ag_i, arg_k)$
18:                 $action(discArg(arg_k, y))$
19:                 for all $f \in arg_k$ do
20:                     $support \leftarrow action(voteSupport(f, t))$
21:                     $reject \leftarrow action(voteRejection(f, t))$
22:                     $b \leftarrow buf(f)$
23:                     $updateDT(y, f, support, reject, b)$
24:                     if $b \geq \sigma$ then
25:                         $action(learn(f, b, iss))$
26:                     else
27:                         $action(inform(f, 0, iss))$
28:                 $updateDT(y, is(arg_k))$
29:                 $action(attack(t))$
30:                 $agents \leftarrow action(askSpeak())$
31:                 $updateWB(agents)$
```

sent the current argument, and $Support[f]$ and $Reject[f]$ as the set of agents that voted to support or reject formula f, respectively.

$$buf(f) = ex_i + \sum_{ag_j \in Support[f]} ex_j - \sum_{ag_j \in Reject[f]} ex_j. \tag{1}$$

With $buf(f)$ representing the consensus level on f, $iss_k \in ISS$ the issue under discussion, and $ag_i \in AG$ an argumentative agent:

- $buf(f) \geq \sigma$: formula $f[iss_k(buf(f))]$ should be accepted and considered common knowledge;
- $buf(f) < \sigma$: formula $f[iss_k(0)]$ should be updated only when $f \in \Sigma_i$.

3.3 Consensus Decision Making

Each dialogue table contains a finite set of arguments $ARG = \{arg_1, \ldots, arg_z\}$ with $z > 0$ related to an issue. This set is then mapped to an abstract argumentation framework such as the one proposed by Dung [11], formed of a pair $AF = \langle ARG, R\rangle$ where ARG is the set of arguments and R is a binary relation representing attacks between arguments with $R \subseteq ARG \times ARG$. The notation $R(arg_i, arg_j)$ represents the situation in which arg_i attacks arg_j. During the mapping, an undercut is a single attack relation from the attacking to the attacked (i.e. $R(arg_i, arg_j)$), while a rebuttal is a symmetric relation (i.e. $R(arg_i, arg_j)$ and $R(arg_j, arg_i)$). The abstract argumentation framework is represented as a graph in which the arguments are nodes and the attack relations are edges. During mapping, each line of the dialogue table represents an argument (column *arg*). The attack relation (column *att*) is used to link arguments. Other attacks between arguments can exist and these additional edges need to be identified.

The starting point arg_1 in the dialogue table is the main node in an argumentation graph representing the decision alternative. This node is special since it receives only an undercut representing the arguments against the decision alternative. We refer to $ARGS = ARG \setminus \{arg_1\}$ as the set of all arguments removing the main node.

The consensus decision-making process uses two additional phases to find the decision alternative that is most preferred by the group: computation of the strength of the arguments and determination of the extent to which one alternative is preferred to another.

3.3.1 Computing Argument Strength

Arguments have two types of strength: intrinsic and overall strength [8]. The intrinsic strength is a value obtained using the concept of group majority knowledge, which expresses the extent to which an argument is reliable based on its formulae. This type of strength considers the supporting and rejecting votes in each formula of the structured argument sent during the dialogue. The overall strength is a score representing the importance of the arguments when compared to other arguments in an argumentation graph. This type of strength considers the attack relations between arguments in an abstract argumentation graph.

Equations (2) and (3) show how to compute the intrinsic and overall strengths, respectively. Let $length : ARG \to \mathbb{N}$ be a function that returns the number of formulae of an argument $arg_i \in ARG$ and $attack : ARG \to ATT$ with $ATT \subseteq ARG$ be a function that returns the set of arguments that attack arg_i, that is,

$\{arg_j \in ARG | R(arg_j, arg_i)\}$.

$$is(arg_i) = \left(\frac{\sum_{f \in split(arg_i)} buf(f)}{length(arg_i)} + 1 \right) * 0.5, \tag{2}$$

$$os(arg_i) = \frac{is(arg_i)}{1 + \sum_{arg_j \in attack(arg_i)} os(arg_j)}. \tag{3}$$

To solve the entire system of argumentation, we use an iterative method [10] on the set $ARGS$. Let $time_0$ be the initial overall strength calculation for each argument, and $time_s$ the overall strength calculation obtained after the s^{th} iteration. An iteration at $time_s$ computes a new overall strength at $time_{s+1}$ for all arguments in $ARGS$. We refer to $os(arg_i)^s$ as the process of computing the overall strength of arg_i in iteration s. The iteration terminates when $os(arg_i)^s = os(arg_i)^{s+1}$ for all arguments. The result is independent of the processing order of the arguments.

3.3.2 Computing Preference Relations

To determine the preference relations among the decision alternatives, we propose an adaptation to the argument labelling in which the arguments receive a label of "in", "out" or "undec" according to their iterations with other arguments in the argumentation graph [7]. These labels are used to specify the arguments that are accepted (in) or rejected (out), and those that are neither accepted nor rejected (undec) [23].

In this adaptation, the labelling implies that arguments with greater overall strengths (labelled as "in") are acceptable (or partially acceptable) and undermine those arguments with lower overall strengths (labelled as "out") that are attacked by them. Arguments labelled as "undec" are those with identical overall strengths. In this case, we use the intrinsic strength that represents the consensus level to determine the argument most preferred by the group ("in" or "out"). The arguments are designated as "undec" only when both the overall and intrinsic strengths are the same and there is no attacking argument labelled "in".

We define the following functions to obtain the neighbours of an argument $arg_i \in ARGS$: $getAllNeighbors : ARGS \to NB$ where NB is the set of all neighbours of arg_i (we consider neighbours to be the attackers and attacked arguments with $NB = \{arg_b \in ARGS | R(arg_b, arg_i) \cup R(arg_i, arg_b)\}$); $getLabeledNeighbors : NB \to LNB$ where LNB is the set of neighbours with an associated label ("in", "out", or "undec"); and $ACC : AF \to INARGS$ is the set of acceptable arguments of an argumentation graph with the label "in". The argument labelling is a function that assigns a label to each argument in the graph. Argument labelling uses two sets P and Q representing the set of all labelled neighbours with maximum overall strength and the set of all labelled neighbours with equal overall strength, respectively, where $P = \{arg_j \in LNB | os(arg_j) > os(arg_i)\}$ and $Q = \{arg_j \in LNB | os(arg_j) = os(arg_i)\}$.

Definition 6. Let $arg_i \in ARGS$ be an argument and P and Q be the sets of labelled neighbours with maximum and equal overall strengths, respectively. An argument labelling for an argumentation graph is a total function $L : ARGS \to \{in, out, undec\}$ such that:

1. if $P = \emptyset$ and $Q = \emptyset$, then $L(arg_i) = in$;
2. if $P = \emptyset$ and $(\exists arg_j \in Q)(is(arg_i) = is(arg_j))$, then $L(arg_i) = undec$;
3. if $P = \emptyset$ and $(\exists arg_j \in Q)(is(arg_i) > is(arg_j))$, then $L(arg_i) = in$;
4. if $P = \emptyset$ and $(\exists arg_j \in Q)(is(arg_i) < is(arg_j))$, then $L(arg_i) = out$;
5. if $(\exists arg_j \in P)(L(arg_j) = in)$, then $L(arg_i) = out$;
6. if $(\forall arg_j \in P)(L(arg_j) \in \{out, undec\})$, then $L(arg_i) = in$.

The labelling of all arguments uses an iterative method to calculate the overall strength of the arguments, since the arguments have links between them and when an argument is labelled, its attacks and attackers need to be reviewed. We refer to $L(arg_i)^w$ as the process of labelling argument arg_i in an iteration w. The last iteration occurs when $L(arg_i)^w = L(arg_i)^{w+1}$ for all arguments in $ARGS$.

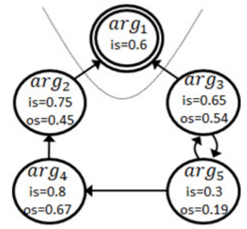
Let $POS : ARGS \to \{-1, 1\}$ be a function that returns 1 if arg_i is a supporting argument, and -1 otherwise. Supporting arguments are those with even distance over a simple shorter path to the main node in the argumentation graph, while rejecting arguments are those with odd distance over a simple shorter path to the main node in the argumentation graph. To compute the preference level for the decision alternative iss, we need to compute the position of the accepted arguments according to Equation (4). The preferred order relation of two decision alternatives is denoted by the symbols $\succ$ and $\sim$ (preferred or equally preferred, respectively). Whenever $pref(iss_1) > pref(iss_2)$ we have $iss_1 \succ iss_2$; for $pref(iss_1) < pref(iss_2)$ we have $iss_2 \succ iss_1$; and for $pref(iss_1) = pref(iss_2)$ we have $iss_1 \sim iss_2$. In this case, the choice of the preferred decision alternative is random. In the special case when there is no argument in the dialogue, that is, if $ARGS = \emptyset$, then the preference level for the decision alternative is $pref(iss) = is(arg_1)$.

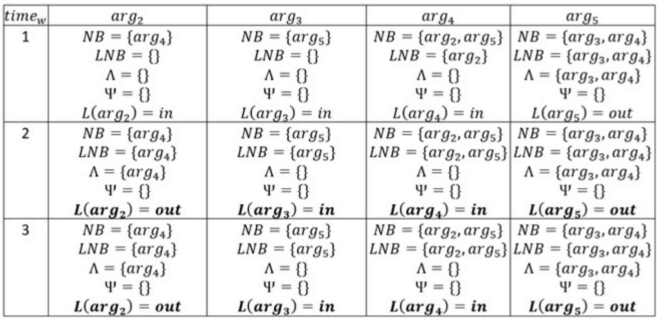
$$pref(iss) = \sum_{arg_y \in INARGS} POS(arg_y) * os(arg_y). \tag{4}$$

Example 5. Consider the argumentation graph and the iterations for labelling in Figure 2. We have $L(arg_2) = out$, $L(arg_3) = in$, $L(arg_4) = in$, $L(arg_5) = out$, $ACC = \{arg_3, arg_4\}$, $POS(arg_3) = -1$, and $POS(arg_4) = 1$. The preference level for the issue is $pref(iss_i) = 0.13$

4 PRACTICAL EXAMPLE

Our example consists of a discussion among three agents that are trying to decide whether a robot should rescue a human being in a disaster situation. The robot has





$time_w$	arg_2	arg_3	arg_4	arg_5
1	$NB = \{arg_4\}$ $LNB = \{\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $L(arg_2) = in$	$NB = \{arg_5\}$ $LNB = \{\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $L(arg_3) = in$	$NB = \{arg_2, arg_5\}$ $LNB = \{arg_2\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $L(arg_4) = in$	$NB = \{arg_3, arg_4\}$ $LNB = \{arg_3, arg_4\}$ $\Lambda = \{arg_3, arg_4\}$ $\Psi' = \{\}$ $L(arg_5) = out$
2	$NB = \{arg_4\}$ $LNB = \{arg_4\}$ $\Lambda = \{arg_4\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_2) = out}$	$NB = \{arg_5\}$ $LNB = \{arg_5\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_3) = in}$	$NB = \{arg_2, arg_5\}$ $LNB = \{arg_2, arg_5\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_4) = in}$	$NB = \{arg_3, arg_4\}$ $LNB = \{arg_3, arg_4\}$ $\Lambda = \{arg_3, arg_4\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_5) = out}$
3	$NB = \{arg_4\}$ $LNB = \{arg_4\}$ $\Lambda = \{arg_4\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_2) = out}$	$NB = \{arg_5\}$ $LNB = \{arg_5\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_3) = in}$	$NB = \{arg_2, arg_5\}$ $LNB = \{arg_2, arg_5\}$ $\Lambda = \{\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_4) = in}$	$NB = \{arg_3, arg_4\}$ $LNB = \{arg_3, arg_4\}$ $\Lambda = \{arg_3, arg_4\}$ $\Psi' = \{\}$ $\boldsymbol{L(arg_5) = out}$

Figure 2. Labelling of arguments for issue iss_i. The double line represents the main node.

a stretcher that can carry only one person at a time. There are two possible decision alternatives for the robot: recharge its battery, x, or rescue the individual and take him/her to the hospital, y. Let $CKF = \langle\{ag_1, ag_2, ag_3\}, \{0.4, 0.3, 0.3\}, \{x, y\}, med, 10, 0.4\rangle$ and the atoms in the formulae represent the sentences: $a =$ the battery has less than 70 % charge; $b =$ the person is very far from the robot; $c =$ the risk of death is 9 ($[0, 10]$ where 10 means the person is dead); and $d =$ the person is alive. The initial knowledge is:

$$ag_1 = \{\{a, b, \neg c, c \to d, a \wedge b \to x, a \wedge b \to \neg y\}, \{\}\},$$

$$ag_2 = \{\{a, \neg b, c, c \to d, a \wedge b \to x, a \wedge b \to y, a \wedge b \to \neg d\}, \{\}\},$$

$$ag_3 = \{\{b, c, c \to d, d \to \neg x, a \wedge b \to x, a \wedge b \to \neg y, d \to y\}, \{\}\}.$$

Agent med creates the starting point $arg_1 = \langle\{T\}, x\rangle$ and informs the group (arg_1 is stored in the A_i base of all argumentative agents). For voting, we have $Support[x] = \{ag_1\}$ where ag_1 has the argument $\langle\{a, b, a \wedge b \to x\}, x\rangle$ supporting it, and $Reject[x] = \{ag_3\}$ where ag_3 has the argument $\langle\{c, c \to d, d \to \neg x\}, \neg x\rangle$, with $buf(x) = 0.1$. Agent ag_3 has $S_3 = \{\langle\{c, c \to d, d \to \neg x\}, \neg x\rangle\}$ and sends this to med when requested. After being checked by the mediator, this argument is inserted into the dt_x as arg_2. Thus, for voting, we have $Support[c] = \{ag_2\}$, $Reject[c] = \{ag_1\}$, $Support[c \to d] = \{ag_1, ag_2\}$, $Reject[\neg x] = \{ag_1\}$, $buf(c) = 0.2$, $buf(c \to d) = 1$, $buf(d \to \neg x) = 0.3$, and $buf(\neg x) = -0.1$. The group is informed of all formulae and these are updated with the corresponding label.

Agent ag_1 has two arguments arg_3 and arg_4 in $S_1 = \{\langle\{\neg c\}, \neg c\rangle, \langle\{a, b, a \wedge b \to x\}, x\rangle\}$. When requested, ag_1 sends the arguments, med checks them and informs the group. For voting, we have: $Reject[\neg c] = \{ag_2, ag_3\}$ with $buf(\neg c) = -0.2$ (agents ag_2 and ag_3 have the counterargument $\langle\{c\}, c\rangle$ but formula $c[x(0)]$ was presented in a previous argument and was not accepted, meaning that this argument is not admitted); $Support[a] = \{ag_2\}$ and $Reject[a] = \{ag_3\}$ with $buf(a) = 0.4$, $Support[b] = \{ag_3\}$ and $Reject[b] = \{ag_2\}$ with $buf(b) = 0.4$, $Support[a \wedge b \to x] = \{ag_2, ag_3\}$ with $buf(a \wedge b \to x) = 1$ (all agents know this formula), $Reject[x] = \{ag_3\}$

with $\mathit{buf}(x) = 0.1$. Formulae $a[x(0.4)]$, $b[x(0.4)]$, and $a \wedge b \to x[x(1)]$ are then accepted by the group, becoming common knowledge.

Agent ag_2 has the argument $\mathit{arg}_5 = \langle\{\neg b\}, \neg b\rangle$ in S_2 with $\mathit{Support}[\neg b] = \{ag_3\}$, $\mathit{Reject}[\neg b] = \{ag_1, ag_3\}$ and $\mathit{buf}(\neg b) = -0.1$. Agent ag_3 has the argument $\langle\{b\}, b\rangle$ with $\mathit{Support}[b] = \{ag_1, ag_2\}$, $\mathit{Reject}[b] = \{ag_2\}$ and $\mathit{buf}(b) = 0.7$. Since WB and AGENDA are empty, the dialogue for x is complete. It is important to note that the argumentative agents have other counterarguments during the dialogue, but these are not admitted. Agent *med* starts the dialogue again for the next decision alternative. After all dialogues, the knowledge of the agents is:

$$\begin{aligned} ag_1 = \{&\{a[x(0.4), y(0.4)], b[x(0.4), y(0.4)], \neg c[x(0), y(0.7)], c \to d[x(1), y(1)],\\ &a \wedge b \to x[x(1)], a \wedge b \to \neg y[y(0.4)]\}, \{\}\},\end{aligned}$$

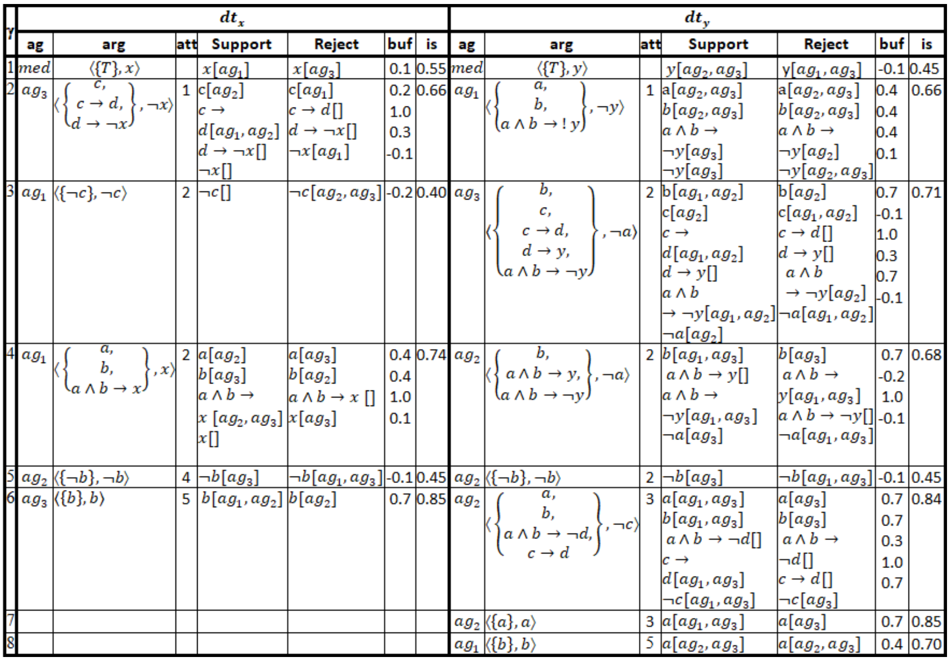
$$\begin{aligned} ag_2 = \{&\{a[x(0.4), y(0.4)], \neg b[x(0), y(0)], c[x(0), y(0)], c \to d[x(1), y(1)],\\ &a \wedge b \to x[x(1)], a \wedge b \to y[y(0)], a \wedge b \to \neg d[y(0)]\}, \{b[x(0.4), y(0.4)],\\ &a \wedge b \to \neg y[y(0.4)], \neg c[y(0.7)]\}\},\end{aligned}$$

$$\begin{aligned} ag_3 = \{&\{b[x(0.4), y(0.4)], c[x(0), y(0)], c \to d[x(1), y(1)], d \to \neg x[x(0)],\\ &a \wedge b \to x[x(1)], a \wedge b \to \neg y[y(0.4)], d \to y[y(0)]\}, \{a[x(0.4), y(0.4)],\\ &\neg c[y(0.7)]\}\}.\end{aligned}$$

Table 1 shows the dialogues for x and y. The corresponding argumentation graphs are shown in Figure 3 with the arguments and their overall strengths. The steps used in labelling the arguments are presented in Table 2. For a dialogue about x, we have: $L(\mathit{arg}_2) = \mathit{out}$, $L(\mathit{arg}_3) = \mathit{in}$, $L(\mathit{arg}_4) = \mathit{in}$, $L(\mathit{arg}_5) = \mathit{out}$, $L(\mathit{arg}_6) = \mathit{in}$, $\mathit{ACC} = \{\mathit{arg}_3, \mathit{arg}_4, \mathit{arg}_6\}$, $\mathit{POS}(\mathit{arg}_3) = 1$, $\mathit{POS}(\mathit{arg}_4 = 1)$, $\mathit{POS}(\mathit{arg}_6) = 1$ and $\mathit{pref}(x) = 1.52$. For a dialogue about y, we have: $L(\mathit{arg}_2) = \mathit{in}$, $L(\mathit{arg}_3) = \mathit{out}$, $L(\mathit{arg}_4) = \mathit{out}$, $L(\mathit{arg}_5) = \mathit{out}$, $L(\mathit{arg}_6) = \mathit{in}$, $L(\mathit{arg}_7) = \mathit{in}$, $L(\mathit{arg}_8) = \mathit{in}$, $\mathit{ACC} = \{\mathit{arg}_2, \mathit{arg}_6, \mathit{arg}_7, \mathit{arg}_8\}$, $\mathit{POS}(\mathit{arg}_2) = -1$, $\mathit{POS}(\mathit{arg}_6) = -1$, $\mathit{POS}(\mathit{arg}_7) = -1$, $\mathit{POS}(\mathit{arg}_8) = -1$, and $\mathit{pref}(y) = -1.83$. As a result of this dialogue, we have $x \succ y$, where x is the preferred decision alternative for the group.

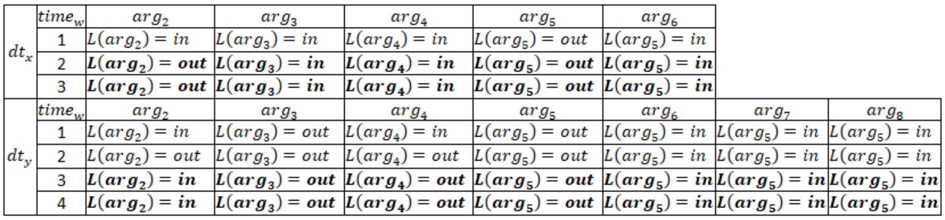
4.1 Results and Discussion

From the practical example given above, it is possible to observe the relation between common knowledge and consensus about information that is accepted by the group of agents. In this work, we refer to each decision alternative as a possible world. One approach to formalising these possible worlds is the Kripke structure [12]. A Kripke Structure KS for n agents over a set of primitive propositions Γ is a tuple $(P, \pi, \Theta_1, \ldots, \Theta_n)$ where P is a non-empty set of possible worlds; $\pi : (p) \to \{\mathit{true}, \mathit{false}\}$ is a function that assigns a truth value to the propositions in Γ for each possible world $p \in P$; and Θ_i is a binary accessibility relation between the possible worlds in P. We can represent the practical example in a Kripke Structure using the system S5 [12, 14, 18] for knowledge representation in each agent



γ	dt_x							dt_y						
	ag	**arg**	**att**	**Support**	**Reject**	**buf**	**is**	**ag**	**arg**	**att**	**Support**	**Reject**	**buf**	**is**
1	med	$\langle\{T\}, x\rangle$		$x[ag_1]$	$x[ag_3]$	0.1	0.55	med	$\langle\{T\}, y\rangle$		$y[ag_2, ag_3]$	$y[ag_1, ag_3]$	-0.1	0.45
2	ag_3	$\langle\left\{\begin{matrix} c, \\ c \to d, \\ d \to \neg x \end{matrix}\right\}, \neg x\rangle$	1	$c[ag_2]$ $c \to$ $d[ag_1, ag_2]$ $d \to \neg x[]$ $\neg x[]$	$c[ag_1]$ $c \to d[]$ $d \to \neg x[]$ $\neg x[ag_1]$	0.2 1.0 0.3 -0.1	0.66	ag_1	$\langle\left\{\begin{matrix} a, \\ b, \\ a \wedge b \to !\, y \end{matrix}\right\}, \neg y\rangle$	1	$a[ag_2, ag_3]$ $b[ag_2, ag_3]$ $a \wedge b \to$ $\neg y[ag_3]$ $\neg y[ag_3]$	$a[ag_2, ag_3]$ $b[ag_2, ag_3]$ $a \wedge b \to$ $\neg y[ag_2]$ $\neg y[ag_2, ag_3]$	0.4 0.4 0.4 0.1	0.66
3	ag_1	$\langle\{\neg c\}, \neg c\rangle$	2	$\neg c[]$	$\neg c[ag_2, ag_3]$	-0.2	0.40	ag_3	$\langle\left\{\begin{matrix} b, \\ c, \\ c \to d, \\ d \to y, \\ a \wedge b \to \neg y \end{matrix}\right\}, \neg a\rangle$	2	$b[ag_1, ag_2]$ $c[ag_2]$ $c \to$ $d[ag_1, ag_2]$ $d \to y[]$ $a \wedge b$ $\to \neg y[ag_1, ag_2]$ $\neg a[ag_2]$	$b[ag_2]$ $c[ag_1, ag_2]$ $c \to d[]$ $d \to y[]$ $a \wedge b$ $\to \neg y[ag_2]$ $\neg a[ag_1, ag_2]$	0.7 -0.1 1.0 0.3 0.7 -0.1	0.71
4	ag_1	$\langle\left\{\begin{matrix} a, \\ b, \\ a \wedge b \to x \end{matrix}\right\}, x\rangle$	2	$a[ag_2]$ $b[ag_3]$ $a \wedge b \to$ $x\ [ag_2, ag_3]$ $x[]$	$a[ag_3]$ $b[ag_2]$ $a \wedge b \to x\ []$ $x[ag_3]$	0.4 0.4 1.0 0.1	0.74	ag_2	$\langle\left\{\begin{matrix} b, \\ a \wedge b \to y, \\ a \wedge b \to \neg y \end{matrix}\right\}, \neg a\rangle$	2	$b[ag_1, ag_3]$ $a \wedge b \to y[]$ $a \wedge b \to$ $\neg y[ag_1, ag_3]$ $\neg a[ag_3]$	$b[ag_3]$ $a \wedge b \to$ $y[ag_1, ag_3]$ $a \wedge b \to \neg y[]$ $\neg a[ag_1, ag_3]$	0.7 -0.2 1.0 -0.1	0.68
5	ag_2	$\langle\{\neg b\}, \neg b\rangle$	4	$\neg b[ag_3]$	$\neg b[ag_1, ag_3]$	-0.1	0.45	ag_2	$\langle\{\neg b\}, \neg b\rangle$	2	$\neg b[ag_3]$	$\neg b[ag_1, ag_3]$	-0.1	0.45
6	ag_3	$\langle\{b\}, b\rangle$	5	$b[ag_1, ag_2]$	$b[ag_2]$	0.7	0.85	ag_2	$\langle\left\{\begin{matrix} a, \\ b, \\ a \wedge b \to \neg d, \\ c \to d \end{matrix}\right\}, \neg c\rangle$	3	$a[ag_1, ag_3]$ $b[ag_1, ag_3]$ $a \wedge b \to \neg d[]$ $c \to$ $d[ag_1, ag_3]$ $\neg c[ag_1, ag_3]$	$a[ag_3]$ $b[ag_3]$ $a \wedge b \to$ $\neg d[]$ $c \to d[]$ $\neg c[ag_3]$	0.7 0.7 0.3 1.0 0.7	0.84
7								ag_2	$\langle\{a\}, a\rangle$	3	$a[ag_1, ag_3]$	$a[ag_3]$	0.7	0.85
8								ag_1	$\langle\{b\}, b\rangle$	5	$a[ag_2, ag_3]$	$a[ag_2, ag_3]$	0.4	0.70

Table 1. Arguments, supporting and rejecting votes and intrinsic strengths for x and y



	$time_w$	arg_2	arg_3	arg_4	arg_5	arg_6		
dt_x	1	$L(arg_2) = in$	$L(arg_3) = in$	$L(arg_4) = in$	$L(arg_5) = out$	$L(arg_5) = in$		
	2	$\boldsymbol{L(arg_2) = out}$	$\boldsymbol{L(arg_3) = in}$	$\boldsymbol{L(arg_4) = in}$	$\boldsymbol{L(arg_5) = out}$	$\boldsymbol{L(arg_5) = in}$		
	3	$\boldsymbol{L(arg_2) = out}$	$\boldsymbol{L(arg_3) = in}$	$\boldsymbol{L(arg_4) = in}$	$\boldsymbol{L(arg_5) = out}$	$\boldsymbol{L(arg_5) = in}$		
	$time_w$	arg_2	arg_3	arg_4	arg_5	arg_6	arg_7	arg_8
dt_y	1	$L(arg_2) = in$	$L(arg_3) = out$	$L(arg_4) = in$	$L(arg_5) = out$	$L(arg_5) = in$	$L(arg_5) = in$	$L(arg_5) = in$
	2	$L(arg_2) = out$	$L(arg_3) = out$	$L(arg_4) = out$	$L(arg_5) = out$	$L(arg_5) = in$	$L(arg_5) = in$	$L(arg_5) = in$
	3	$\boldsymbol{L(arg_2) = in}$	$\boldsymbol{L(arg_3) = out}$	$\boldsymbol{L(arg_4) = out}$	$\boldsymbol{L(arg_5) = out}$	$\boldsymbol{L(arg_5) = in}$	$\boldsymbol{L(arg_5) = in}$	$\boldsymbol{L(arg_5) = in}$
	4	$\boldsymbol{L(arg_2) = in}$	$\boldsymbol{L(arg_3) = out}$	$\boldsymbol{L(arg_4) = out}$	$\boldsymbol{L(arg_5) = out}$	$\boldsymbol{L(arg_5) = in}$	$\boldsymbol{L(arg_5) = in}$	$\boldsymbol{L(arg_5) = in}$

Table 2. Labelling arguments from dt_x and dt_y

where the possible worlds are symmetric, transitive and reflexive. Let the sequence $[x, y]$ represent the decision alternatives, with values T for true and F for false.

Before the dialogue, agent ag_1 has one argument asserting x ($\langle\{a, b, a \wedge b \to x\}, x\rangle$) and one argument asserting $\neg y$ ($\langle\{a, b, a \wedge b \to \neg y\}, \neg y\rangle$). For this agent, there is only one possible world: $w_1 = [T, F]$. Agent ag_2 does not have an argument for either x or y. In this case, both decision alternatives are accepted with four possible worlds: $w_1 = [T, F]$, $w_2 = [F, T]$, $w_3 = [T, T]$, $w_4 = [F, F]$. Agent ag_3 has one argument for $\neg x$ ($\langle\{c, c \to d, d \to \neg x\}, \neg x\rangle$) and one for y ($\langle\{c, c \to d, d \to$

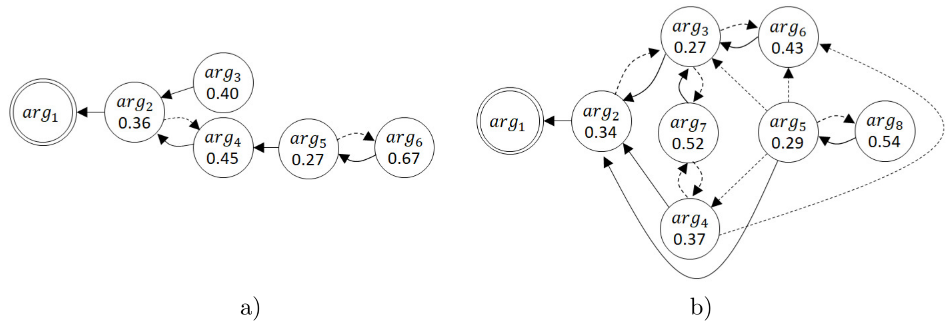


Figure 3. Argumentation graphs mapped from a) dt_x and b) dt_y. Double lines represent the main node, and dotted lines are additional attack relations detected in the mapping.

$y\}, y\rangle)$ with only one possible world: $w_2[F, T]$. In this case, there is no consensus on x or y, as shown in Figure 4 a).

After the dialogue, $ag1$, ag_2 and ag_3 accept only world $w_1 = [T, F]$ (arguments: $\langle\{a, b, a \wedge b \rightarrow x\}, x\rangle$ and $\langle\{a, b, a \wedge b \rightarrow \neg y\}, \neg y\rangle$). Agent ag_2 still has an argument for y ($\langle\{a, b, a \wedge b \rightarrow y\}, y\rangle$) and ag_3 has arguments for $\neg x$ and y ($\langle\{c, c \rightarrow d, d \rightarrow \neg x\}, \neg x\rangle$ and $\langle\{c, c \rightarrow d, d \rightarrow y\}, y\rangle$), but these arguments are not admitted and therefore do not establish a position against $\neg x$ or in favour of y. After the dialogue, we can observe that there is a consensus on world $w_1 = [T, F]$, as shown in Figure 4 b).

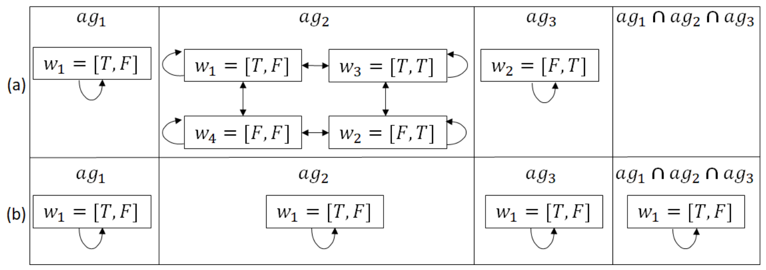


Figure 4. Possible worlds and consensus for agents ag_1, ag_2, and ag_3: a) before the dialogue; and b) after the dialogue, with common knowledge formation

We can also use the modal operators K, E, and C to represent the knowledge of the agents:

- $K_i\varphi$ is equivalent to $\varphi \in \Sigma_i$ (e.g. $K_1\neg c$ implies that $\neg c \in \Sigma_1$). If an agent knows a piece of information, that information is in the agent's knowledge base or can be inferred.
- $\neg K_i K_j \varphi$ (e.g. $K_2 d$ and $\neg K_1 d$ or $\neg K_1 K_2 d$). If an agent does not know about a piece of information, it can query other agents.

- $K_i\varphi$ implies that $K_iK_i\varphi$. If an information is in the agent's knowledge base, the agent knows that information and can use it to build new arguments and vote.
- $\neg K_i\varphi$ implies that $K_i\neg K_i\varphi$. The unknown information cannot be used to build new arguments and vote.
- $E_{AG}\varphi$ is equivalent to $\forall ag_i \in AG : \varphi \in \Sigma_i$. Before the dialogue in the practical example, agents know $c \to d$.
- $C_{AG}\varphi$ is equivalent to $\forall\Sigma_i, \exists\varphi : \varphi[iss(b)]$ for $iss \in ISS$ and $b > 0$. A formula is common knowledge when related to an issue if it has a label with a consensus level greater than zero.

Other characteristics of the model are as follows:

- It is able to represent the different belief states for each formula. Formulae φ and $\neg\varphi$ may be accepted at the same time for the same issue. The agent may not have a well-defined position for that information.
- There are two ways in which an agent can decrease the strength of an argument: voting for rejection of its formulae or sending counterarguments.
- Maximal acceptance of an argument arg results in $is(arg) = 1$, while maximal rejection of the argument results in $is(arg) = ex_i$ (the expertise value of the proponent).
- A small number of strong attacks may be equivalent to or more rigorous than several weak attacks.

4.2 Related Works

Dung [11] proposed some semantics to determine the admissibility of the arguments, that is, a formal framework to identify conflict outcomes, such as preferred or grounded semantics. The idea is to specify sets of acceptable arguments or extensions. An extension is a set of arguments that can be accepted together. These semantics are used to select arguments without considering support for or rejection of a decision alternative or group decision.

Coste-Marquis et al. [9] used an argumentation graph with weights in the attack relations, and applied Dung's semantics in determining the last attacked or best defended extensions. Our proposal deals with strengths in arguments represented as numerical values, applied when a group of agents intends to select the preferred alternative by considering the opinions of all the agents.

Da Costa Pereira et al. [10] use a belief revision based on argumentation that assigns fuzzy labelling to each argument, permitting the agent to change its mind without removing the previous information forever, and allowing for recovery if this information turns out to be wrong. In our work, the evaluation is carried out based on the set of formulae of the argument, all arguments are evaluated, and the agents store all the information that is acceptable to every possible world in their knowledge bases.

The work closest to our approach is probably that of Leite and Martins [19], who extend Dung's framework by applying it to online debate systems, allowing people to vote to support or reject arguments or to send arguments that are not logically structured. They defined a semantics for application to online debating systems (democracy, universality, etc.) and to rank the arguments from the strongest to the weakest, suggesting a preference for the group, although not a definite one. In our work, arguments are logically structured; they are sent by agents within a restricted group; the strengths of the arguments use quantitative (votes) as well as qualitative (attacks) values; there are different dialogues, one for each issue; and the framework still creates the preference order of the possible decision alternatives as a result.

5 CONCLUSION

This work presents an argumentative dialogue model for CKF in a group of agents. This model is generic, and can be applied to a discussion about any issue where there is a need for group opinions. We use propositional logic to represent the information in the knowledge bases, and to build arguments and a voting model for support and rejection, although other logical languages may be used.

The model has four main characteristics:

1. it allows agents to take part in a dialogue by exchanging arguments through attack relations, while supporting or rejecting the arguments by voting on their formulae;
2. based on the expertise of the agents, we can evaluate the arguments in numerical form, representing the extent to which each formula (or argument as a whole) is accepted by the group of agents;
3. the results obtained after the dialogue allow for an approximation of opinions, meaning that the group can apply the model in consensus decision-making problems; and
4. the model presents a direct relation between common knowledge and consensus.

When a piece of information is taken as common knowledge, there is a consensus of the group accepting that information. The output of the model is not the optimal decision, but rather the decision preferred by the group.

There are several possible ways to extend this work. Some of these future directions involve the application of the model to decision making in which blocking is possible, and the use of a reputation system to assign expertise to the agents, where each value is related to the type of information presented in the argument.

REFERENCES

[1] Amgoud, L.—Belabbès, S.—Prade, H.: A Formal General Setting for Dialogue Protocols. In: Euzenat, J., Domingue, J. (Eds.): Artificial Intelligence: Methodology,

Systems, and Applications (AIMSA 2016). International Workshop on Argumentation in Multi-Agent Systems, Varna, Bulgaria, September 2006. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4183, 2006, pp. 13–23, doi: 10.1007/11861461_4.

[2] Amgoud, L.—Ben-Naim, J.—Doder, D.—Vesic, S.: Acceptability Semantics for Weighted Argumentation Frameworks. Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Vol. 1, 2017, pp. 56–62, doi: 10.24963/ijcai.2017/9.

[3] Amgoud, L.—Prade, H.: Using Arguments for Making and Explaining Decisions. Journal of Artificial Intelligence, Vol. 173, 2009, No. 3, pp. 413–436, doi: 10.1016/j.artint.2008.11.006.

[4] Besnard, P.—Hunter, A.: Constructing Argument Graphs with Deductive Arguments: A Tutorial. Argument and Computation, Vol. 5, 2014, No. 1, pp. 5–30, doi: 10.1080/19462166.2013.869765.

[5] Bistarelli, S.—Pirolandi, D.—Santini, F.: Solving Weighted Argumentation Frameworks with Soft Constraints. In: Larrosa, J., O'Sullivan, B. (Eds.): Recent Advances in Constraints (CSCLP 2009). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6384, 2011, pp. 1–18, doi: 10.1007/978-3-642-19486-3_1.

[6] Bodanza, G.—Tohmé, F.—Auday, M.: Collective Argumentation: A Survey of Aggregation Issues Around Argumentation Frameworks. Argument and Computation, Vol. 8, 2017, No. 1, pp. 1–34, doi: 10.3233/aac-160014.

[7] Caminada, M.: On the Issue of Reinstatement in Argumentation. In: Fisher, M., van der Hoek, W., Konev, B., Lisitsa, A. (Eds.): Logics in Artificial Intelligence (JELIA 2006). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4160, 2006, pp. 111–123, doi: 10.1007/11853886_11.

[8] Cayrol, C.—Lagasquie-Schiex, M. C.: Graduality in Argumentation. Journal of Artificial Intelligence Research, Vol. 23, 2005, pp. 245–297, doi: 10.1613/jair.1411.

[9] Coste-Marquis, S.—Konieczny, S.—Marquis, P.—Ouali, M. A.: Selecting Extensions in Weighted Argumentation Frameworks. In: Verheij, B., Szeider, S., Woltran, S. (Eds.): Computational Models of Argument. IOS Press, Frontiers in Artificial Intelligence and Applications, Vol. 245, 2012, pp. 342–349, doi: 10.3233/978-1-61499-111-3-342.

[10] Da Costa Pereira, C.—Tettamanzi, A. G. B.—Villata, S.: Changing One's Mind: Erase or Rewind? Possibilistic Belief Revision with Fuzzy Argumentation Based on Trust. Proceedings of the 22$^{\text{nd}}$ International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, 2011, pp. 164–171.

[11] Dung, P. M.: On the Acceptability of Arguments and Its Fundamental Role in Nonmonotonic Reasoning, Logic Programming and n-Person Games. Artificial Intelligence, Vol. 77, 1995, No. 2, pp. 321–357, doi: 10.1016/0004-3702(94)00041-x.

[12] Fagin, R.—Halpern, J. Y.—Moses, Y.—Vardi, M. Y.: Reasoning about Knowledge. MIT Press, London, England, 2004.

[13] Fagin, R.—Halpern, J. Y.: Reasoning about Knowledge and Probability. Journal of the ACM, Vol. 41, 1994, No. 2, pp. 340–367, doi: 10.1145/174652.174658.

[14] Girle, R.: Possible Worlds. Acumen Publishing, Durham, 2013, doi: 10.1017/upo9781844653454.

[15] Grossi, D.—Pigozzi, G.: Judgment Aggregation: A Primer. Synthesis Lectures on Artificial Intelligence and Machine Learning, Morgan and Claypool, Vol. 8, 2014, No. 2, pp. 1–151, doi: 10.2200/s00559ed1v01y201312aim027.

[16] Halpern, J. Y.—Moses, Y.: Knowledge and Common Knowledge in a Distributed Environment. Journal of the ACM, Vol. 37, 1990, No. 3, pp. 549–587, doi: 10.1145/79147.79161.

[17] Halpern, J. Y.: Reasoning about Knowledge: An Overview. Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning about Knowledge, Monterey, California, 1986, pp. 1–17, doi: 10.1016/b978-0-934613-04-0.50004-1.

[18] van der Hoek, W.: Systems for Knowledge and Belief. Journal of Logic and Computation, Vol. 3, 1993, No. 2, pp. 173–195, doi: 10.1093/logcom/3.2.173.

[19] Leite, J.—Martins, J.: Social Abstract Argumentation. Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence (IJCAI '11), Barcelona, Spain, Vol. 3, 2011, pp. 2287–2292, doi: 10.5591/978-1-57735-516-8/IJCAI11-381.

[20] Ménager, L.: Consensus and Common Knowledge of an Aggregate of Decisions. Games and Economic Behaviour, Vol. 62, 2008, No. 2, pp. 722–731, doi: 10.1016/j.geb.2007.04.007.

[21] Parsons, S.—McBurney, P.: Argumentation-Based Dialogues for Agent Co-Ordination. Group Decision and Negotiation, Vol. 12, 2003, No. 5, pp. 415–439, doi: 10.1023/b:grup.0000003742.50038.d3.

[22] Rahwan, I.—Simari, G. R. (Eds.): Argumentation in Artificial Intelligence. Springer US, 2009, doi: 10.1007/978-0-387-98197-0.

[23] Rahwan, I.—Tohmé, F.: Collective Argument Evaluation as Judgement Aggregation. Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS '10), Toronto, Canada, Vol. 1, 2010, pp. 417–424.

[24] Rome, A. G. L.: Unanimity Rule and Organizational Decision Making: A Simulation Model. Organization Science, Vol. 15, 2004, No. 6, pp. 704–718, doi: 10.1287/orsc.1040.0090.

[25] Singh, R.—Benyoucef, L.: A Consensus Based Group Decision Making Methodology for Strategic Selection Problems of Supply Chain Coordination. Engineering Applications of Artificial Intelligence, Vol 26, 2013, No. 1, pp. 122–134, doi: 10.1016/j.engappai.2012.03.013.

Ayslan Trevizan Possebom received his B.Sc. in information systems in 2002 from Paranaense University and his M.Sc. in computer science from the State University of Maringa. He is Full Professor of Information Technology at the Federal Institute of Paraná, and a Ph.D. candidate in the electrical and computer engineering program at the Federal University of Technology of Parana. His research interests include multi-agent systems, argumentation and decision making.

Mariela Morveli-Espinoza received her B.Sc. in systems engineering in 2001 from the San Agustin National University of Arequipa and her M.Sc. in systems engineering and computer science from the Federal University of Rio de Janeiro. She is currently a Ph.D. candidate in the electrical and computer engineering program at the Federal University of Technology of Parana. Her research interests include intelligent agents, multi-agent systems and argumentation.

Cesar Augusto Tacla received his B.Sc. in software engineering in 1989 and his M.Sc. in electrical and computer engineering in 1993 from the Federal University of Technology of Parana. In 2003, he was awarded his Ph.D. degree in informatics from the University of Technology of Compiègne. Currently, he is Full Professor within the electrical and computer engineering program at the Federal University of Technology of Parana. His research interests include multi-agent systems, argumentation, ontologies and cooperative systems.

Computing and Informatics, Vol. 38, 2019, 597–620, doi: 10.31577/cai_2019_3_597

FUZZY SIDE INFORMATION CLUSTERING-BASED FRAMEWORK FOR EFFECTIVE RECOMMENDATIONS

Mohammed Wasid

Department of Computer Science & Engineering
Government Engineering College, Bharatpur-321303, India
&
Department of Computer Engineering
Aligarh Muslim University, Aligarh-202001, India
e-mail: erwasid@gmail.com

Rashid Ali

Department of Computer Engineering
Aligarh Muslim University, Aligarh-202001, India
e-mail: rashidaliamu@rediffmail.com

Abstract. Collaborative filtering (CF) is the most successful and widely implemented algorithm in the area of recommender systems (RSs). It generates recommendations using a set of user-product ratings by matching similarity between the profiles of different users. Computing similarity among user profiles efficiently in case of sparse data is the most crucial component of the CF technique. Data sparsity and accuracy are the two major issues associated with the classical CF approach. In this paper, we try to solve these issues using a novel approach based on the side information (user-product background content) and the Mahalanobis distance measure. The side information has been incorporated into RSs to further improve their performance, especially in the case of data sparsity. However, incorporation of side information into traditional two-dimensional recommender systems would increase the dimensionality and complexity of the system. Therefore, to alleviate the problem of dimensionality, we cluster users based on their side information using k-means clustering algorithm and each user's similarity is computed using the Mahalanobis distance method. Additionally, we use fuzzy sets to represent the side information more efficiently. Results of the experimentation with two benchmark datasets show that our framework improves the recommendations quality and predictive accuracy of both traditional and clustering-based collaborative recommendations.

Keywords: Recommender systems, collaborative filtering, Mahalanobis distance, k-means clustering, multi-criteria, demographic recommender

1 INTRODUCTION

Recommender System (RSs) is the information filtering software tool which gives suggestions to internet users for the products which are more likely to be preferred by them or relevant to their choice [1]. Here, a suggestion can be from any domain, such as which movie to watch, which song to listen, what products to buy, or which online news to read. Collaborative filtering (CF) is the most used and popular technology implemented in both industry and academia due to its simplicity and accurate enough recommendations ability [2]. The CF technique is further classified into product-based and user-based methods. The core idea of the product-based method is to provide product suggestions to users based on the other similar products, while the user-based method generates recommendation to a user (target user) by finding a set of users who have high correlations with this user. In both ways, finding similar users (products) to target user (product) is the crucial step for CF technique [3]. Currently, most of the CF similarity measures are based on commonly rated products. Although these CF recommendation methods are popular and widely used, they still suffer a number of inadequacies, including [4]:

- Data sparsity: This is a very usual problem in collaborative recommenders where users give ratings to a small set of products from a broad set of products available in the system. The actual problem occurs in the neighborhood set generation phase, where a very few or no common product ratings are available for similarity computation between users which leads to invalid neighborhood set formation.
- Cold-start: This problem arises in a scenario similar to that of data sparsity. In this problem, it is tough to produce recommendations for users who are newly introduced into the system or have not rated a single product yet.
- Multidimensionality: Traditional RSs works fine in case of two dimensions, i.e., users and products, but tends to fail when a recommendation is needed for a system having more than two dimensions. Therefore, the curse of dimensionality is one of the major issues in classical recommender systems.

All of the above-listed problems are approximately dependent on each other; for instance, while handling data sparsity issue through some user-product features in the system, the multidimensionality issue comes into the picture [5]. Therefore, to tackle these problems, a compact model is required where the fusion of user-product side information into traditional recommender system does not affect its dimensions.

Although researchers are working towards the improvement in the accuracy of recommender systems using the overall user-product single-criterion ratings [3, 6], it has been seen that the user's demographic data, products description, contextual and multi-criteria rating factors significantly influence the utility of recommendations. In this work, we incorporate

1. user demographic information (user age) and
2. multi-criteria ratings as side information for two different recommenders.

A collaborative filtering technique with demographic information is known as demographic recommender systems. We have used user age as the third dimension to alleviate data sparsity issue. Let us suppose a user x who is new in the system and has no past ratings (user cold-start), then, in such case, other users can only be matched with him/her by his/her age. Hence the young age group users will be closer to each other rather than the old age people. Whereas, a multi-criteria recommender system (MCRS) can provide more effective and accurate suggestions to users as compared to the classical RSs. In MCRS the preference of a user is represented based on several aspects of the products [7]. Instead of having a single overall rating for a product, MCRS represents products using multiple components to attain preference of a user in depth [7, 8]. For example, hotels can be evaluated more effectively based on their different components (cleanliness, rooms, cuisines, and price) instead of evaluating on a single overall rating. Now, our job is to find a technique which can combine these side information with traditional recommendation systems without dimension expansion and further improve their accuracy. After connecting the side information into the system, we have to identify a method to compute effective similarity between user models with side information. There is a need for selecting an efficient similarity measure for a top-N neighborhood set generation.

Therefore, in this work, we proposed a framework which has achieved our goals. In our framework, we treated the side information (user age or multi-criteria ratings) as a clustering parameter for k-means clustering. The users are clustered using their age (or multi-criteria ratings) and this reduces the user search space while forming neighborhood set of the user. Each user is assigned to his/her most similar side information cluster. Further, we use Mahalanobis distance measure to compute the distance between users within the group [35]. Our work follows a simple and effective method to generate more expert recommendations to the users. The contribution of the paper can be outlined as follows:

- We propose a recommendation method that improves the accuracy of collaborative filtering and is based on side information, Mahalanobis distance, and k-means clustering algorithm.

- We present a novel framework to deal with the curse of dimensionality in collaborative filtering recommender systems. No method in the literature deals with the multi-criteria and demographic multi-dimensionality problem using a single framework.
- A novel user profile is built on user-product side information, where fuzzy logic is used to handle the sparsity and uncertainty and reduces the complexity of the system.
- Modified Mahalanobis distance measure is proposed for users matching.
- We perform extensive experiments on two real datasets, namely Yahoo! Movies for multi-criteria ratings and MovieLens for user age feature. The results show that our proposed framework can deal with the multi-dimensionality and data sparsity issues effectively and accurately as compared to other traditional techniques.

This paper is organized as follows: Section 2 introduces the background and related work. The proposed recommendation framework is introduced in detail in Section 3. In Section 4, we evaluate the proposed method using the MovieLens 100 K and Yahoo! Movies datasets and compare it with the existing methods. We conclude the paper in Section 5.

2 BACKGROUND

2.1 Recommendation Techniques

Recommender systems employ different information filtering techniques to product recommendations based on the type of application. There are many techniques implemented in literature, but the Content-based (CB), Collaborative filtering (CF), and hybrid techniques are the major recommendation techniques [4]. We will discuss each of them in the following subsections.

2.1.1 Content-Based Technique

This technique suggests products similar to the ones user selected in the past. The core mechanism of this technique depends upon the content or feature of the past preferred products [1]. These contents or features are further used to build a user profile for each user. Thereafter, similarity between user profiles and other products are computed to obtain similar products.

2.1.2 Collaborative Filtering Technique

This is the most widely implemented technique in literature. It produces recommendations based on preferences of other like-minded users in the system. CF works in three steps. First, similarity is computed between users on the basis of their historical ratings. Secondly, the neighborhood set is formed by obtaining most similar users to the target user. Thirdly, predictions and recommendations are made through neighborhood users' collective ratings [2]. We can observe from above mentioned steps that the similarity computation is a critical step for CF technique and the performance of the system considerably depends on the quality of neighborhood set selection. Therefore, there is a need for a good mechanism to find neighbors of target user which can facilitate better recommendations to the users. We will discuss some of the well-known similarity measures in Section 2.2.

2.1.3 Hybrid Filtering

Here, more than one filtering technique is combined to improve the effectiveness of the recommender system. Hybrid filtering is used to remove drawbacks of each technique separately [19]. There are multiple ways to implement a hybrid recommender system; it can be implemented by combining separate recommender techniques or by adding content-based characteristics to collaborative model and vice-versa.

2.2 Similarity Measures

Similarity computation is an intermediate and primary step in collaborative filtering which is used for neighborhood set formation [10, 19]. Here user-product rating matrix is used for similarity computation. Most of the similarity measures fail in case of sparse data. In our work, we will perform experiments on the following similarity measures to compute the similarity/distances between users.

2.2.1 Pearson Correlation Coefficient (PC)

This is the most popular similarity computation method usually applied to memory-based CF [33]. In this method, the similarity between two users is based only on the ratings both users have given to products in the past. The PC is calculated as follows:

$$PC(x,y) = \frac{\sum_{n \in N_{xy}} (r_{x,n} - \bar{r_x})(r_{y,n} - \bar{r_y})}{\sqrt{\sum_{n \in N_{xy}} (r_{x,n} - \bar{r_x})^2} \sqrt{\sum_{n \in N_{xy}} (r_{y,n} - \bar{r_y})^2}} \tag{1}$$

where $r_{x,n}$ is the rating of user x on product n and $\bar{r_x}$ is the mean of the total ratings given by the user x. N_{xy} is the set of products commonly rated by both user x and y.

2.2.2 Cosine-Based Similarity (CS)

This method uses the concept of angle to compute the similarity among different users. This similarity is based on the cosine of the angle between two users x and y.

It is calculated using:

$$CS(x,y) = \frac{\sum_{n \in N_{x,y}} r_{x,n} \cdot r_{y,n}}{\sqrt{\sum_{n \in N_{x,y}} r_{x,n}^2} \sqrt{\sum_{n \in N_{x,y}} r_{y,n}^2}}. \tag{2}$$

2.2.3 Extended Jaccard Coefficient (JC)

This method can be used for continuous or discrete non-negative features and gets reduced to the Jaccard coefficient in case of the binary attributes as input. This coefficient, which is represented as JC, is defined by the following equation:

$$JC(x,y) = \frac{\sum_{n \in N_{x,y}} r_{x,n} r_{y,n}}{\sum_{n \in N_{x,y}} r_{x,n}^2 + \sum_{n \in N_{x,y}} r_{y,n}^2 - \sum_{n \in N_{x,y}} r_{x,n} r_{y,n}}. \tag{3}$$

2.2.4 The Mahalanobis Distance (MD)

This is a well-known distance measuring formula which is calculated using the inverse of the variance-covariance matrix of the dataset of interest [14, 15]. The MD for a single user x is computed similar to the concept of Euclidean distance method.

$$MD = \sqrt{(x - \bar{x}) vc_z^{-1} (x - \bar{x})^T} \tag{4}$$

where

$$vc_z^{-1} = \begin{bmatrix} \sigma_2^2/det(vc_z) & -\rho_{12}\sigma_1\sigma_2/det(vc_z) \\ -\rho_{12}\sigma_1\sigma_2/det(vc_z) & \sigma_1^2/det(vc_z) \end{bmatrix}$$

where σ_1^2 and σ_2^2 are the variances of the values of the first and second users respectively. $\rho_{12}\sigma_1\sigma_2$ is the covariance between the two users and det(vc_z) is the determinant of the variance-covariance matrix (vc_z), which is computed as follows:

$$vc_z = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}. \tag{5}$$

Since Equation (4) is used only for a single user x, we will discuss MD for multiple users in detail in Section 3.2.

2.3 Related Work

In our work, a user-based clustering has been applied to identify users with similar preferences based on user age or multi-criteria ratings. For example, in a restaurant, one may like the food, but not the service, or vice-versa. So, in the real-life scenario, the side information dramatically affects the overall preference of a user. There has been a lot of research done in the area of recommender systems using clustering, like a user-based clustering has been proposed using user-user similarity computations and resulting clusters are used for neighborhood set generation [10].

Ungar and Dean [21] clustered both users and products separately using k-means and Gibbs sampling. Here, users can be un-clustered using the number of products in each product cluster they have rated and vice-versa. Tsai and Chihli [22] proposed cluster ensembles method for collaborative filtering recommendation where they have used the self-organizing map and k-means clustering and three different ensemble methods. A novel clustering-based CF approach was developed where user groups are formed using a proposed method to reduce the impact of the data sparsity [20, 32]. After cluster formation, nearest neighbors are found from each user group to produce the useful recommendation to the users. Similarly, Liu [9] proposed an improved clustering-based collaborative filtering recommender method. In this approach the authors applied k-means clustering to cluster the users and then an enhanced similarity method was developed to generate nearest neighbors in the cluster for the target user. The problem of computational time has also been addressed using k-means clustering by clustering user-product ratings and generate nearest neighbors [23].

Liu et al. [11] presented a multi-criteria recommendation approach by clustering users based on their criteria preferences in a preference lattice. Authors showed that some set of criteria dominate the overall ratings and different users have their own different dominant set of criteria. A clustering and regression-based technique was proposed in [12, 13] to improve the predictive accuracy of multi-criteria CF: different clustering methods were used to detect similar customer segments and regression was used for learning important weights for the various quality factors. Nilashi et al. in [26] proposed a technique to solve scalability and sparsity problems of multi-criteria recommender systems using dimensionality reduction and Neuro-Fuzzy techniques. In their approach the Neuro-Fuzzy technique was used to solve sparsity problem, and scalability was handled using higher order singular value decomposition along with supervised learning (classification) methods. Similarly, in [24], a hybrid recommendation model was proposed to overcome the same issues by using ontology and dimensionality reduction techniques. The authors have used EM clustering to cluster user-product and Singular Value Decomposition (SVD) techniques for dimensionality reduction. Whereas, [25] shows a method which combines dimension reduction and user clustering in collaborative filtering in which the authors have used principal component analysis and SVD techniques for dimensionality reduction plus k-means and agglomerative hierarchical clustering techniques for user clustering. Authors in [30, 31] deal with the curse of dimensionality by handling data sparsity problem of CF technique. Xu et al. [27] used the clustering algorithm to cluster user profile and then combined it with product-based collaborative filtering to improve its performance. Furthermore, authors have incorporated fuzzy set theory to deal with different rating schemas and tackle the scalability issue of recommender systems.

Furthermore, many authors have chosen certain clustering parameters from the user-product profile features in literature. Frémal and Lecron in [16] presented a clustering based recommender system based on product's metadata: they have used movie genre as a clustering parameter. Since a single movie can have multiple genres, therefore, authors assigned a single product to multiple clusters and results from every cluster were combined using different weighting strategies. In the same way, [29] proposed a technique to cluster products based on the contents using k-means algorithms. Product-grain clustering [18] was introduced by choosing contexts as a clustering parameter using k-means algorithm. Further, these context clusters were incorporated into matrix factorization technique to overcome data sparsity, scalability and prediction quality issues. Wang et al. [28] proposed a new algorithm which clusters user attributes using k-means algorithm. Here, longitude and latitude of the user are considered as the clustering parameter and after cluster formation, the similarity of each user is calculated within the respective cluster.

The major problem with these works is that they all have used some additional techniques to alleviate the issue of dimensionality reduction which makes the system more complicated. A complex system will take more time to produce recommendations which may irritate the online user. Unlike the approaches mentioned above, our proposed framework is straightforward and requires no mathematical modelling for dimensionality reduction because we have treated the other dimensions as the feature of user cluster. Our work in this paper can be summarized twofold. First, we identify the clustering parameter for cluster segments and incorporate fuzzy sets to handle the uncertainty issue associated with them. After cluster formation, in the second fold, we applied different similarity measures and compared with our proposed Mahalanobis distance-based method. However, our goal is to study the impact of different similarity measures for clustering and non-clustering approaches.

3 THE PROPOSED FRAMEWORK

In this section, we give a description of our proposed framework which improves recommendation system's accuracy by identifying different users' belongingness into different clusters based on their side information. In the proposed technique, the actual neighbors of users are found based on their Mahalanobis distance within the user cluster. Unlike traditional similarity measures, the Mahalanobis distance not only considers the commonly rated products between the two users but also finds the variance and covariance between them which makes it more efficient to generate the more similar neighborhood set to the target user, thus, improving the effectiveness of the CF recommendation [15]. The framework is constructed according to the idea and description mentioned above. Figure 1 presents the architecture of the system. The proposed algorithm has three main components which are explained in the below subsections.

3.1 User Cluster Formation

In order to generate most relevant user clusters according to their side information, the first step is to identify the clustering parameter from the existing dataset. After

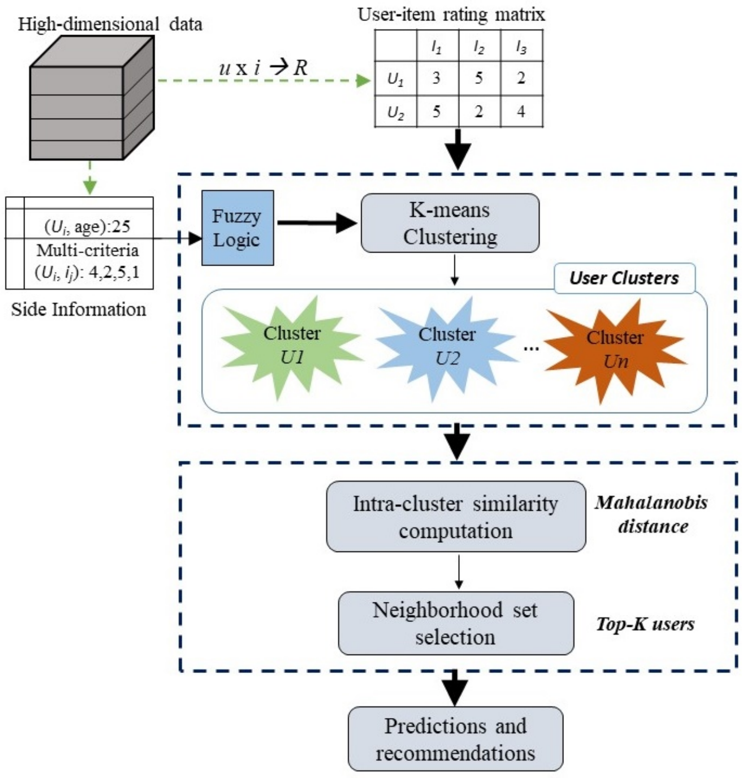


Figure 1. Clustering with fuzzy side information framework

which, the number of user cluster centers has to be defined. The next step is to cluster each user with any one of these pre-specified cluster centers based on their in-between distance (similarity). Those users are assigned to the cluster which has minimum distance (maximum similarity) from them, so that users with most common preferences are grouped in the same cluster. This process iterates until convergence criteria are met. The algorithm for creating user clusters is shown below.

3.1.1 Fuzzy Approach

Fuzzy set has been used to deal with the vague concepts, like ‘old’, ‘short’, ‘poor’, and so on. To incorporate the fuzzy sets into recommender system, proper fuzzification (designing of membership functions) will be required and an appropriate distance function will be needed to match the local and global similarities between different users.

```
Algorithm 1 Algorithm for user cluster formation
  Input: Dataset with Side Information, number of clusters K
  Output: Set of user clusters

  Step 1: [Initialization]
  Initialize randomly K clustering centers $\{c_1, c_2, c_3, .., c_k\}$ from the dataset.
  Step 2: [Assignment]
  a. Find the closest cluster center $c$ for each user $a$ using:
  (i) Equation (8) in case of fuzzified side information OR
  (ii) following Euclidean distance for non-fuzzified side information:
        $Edis_{c,a} = \sqrt{(M_{c,1} - M_{a,1})^2 + (M_{c,2} - M_{a,2})^2 + ... + (M_{c,n} - M_{a,n})^2}$
  where: $Edis_{c,a}$ - distance between the cluster center $c$ and the user $a$;
  $n$ – number of user rated products;
  $M_{c,n}$ – age or multi-criteria ratings of cluster center $c$ for product $n$
  (n=4 for multi-criteria ratings; n=1 for age side information);
  $M_{a,n}$ – age or multi-criteria ratings of user $a$ for product $n$.
  b. Arrange user $a$ by its distance from each clustering centers $\{c_1, c_2, c_3, .., c_k\}$.
  c. Assign the user $a$ with the nearest distance cluster center $c$.
  Step 3: [Convergence]
  a. Iteratively process cluster reassignment until convergence criteria is reached.
  b. If convergence criteria met, then the algorithm terminates and returns a set of
  user clusters $\{c_1, c_2, c_3, .., c_k\}$, otherwise go to step 2.
```

Side information fuzzification: We have used the following fuzzy sets to deal with the uncertainty associated with the user-item side information (multi-criteria ratings and user age).

In our approach, multi-criteria ratings from Yahoo! Movies dataset are classified into six fuzzy sets, namely very bad (VB), bad, average, good, very good, and excellent (Exl) [6], as shown in Figure 2, following are the membership functions for these fuzzy sets:

$$P_{VB}(m) = \begin{cases} 1-m, & m \le 1, \\ 0, & m \ge 1, \end{cases} \tag{6a}$$

$$P_{t(v)}(m) = \begin{cases} 0, & m \le v-2, m > v, \\ m-v+2, & v-2 < m \le v-1, \\ v-m, & v-1 < m \le v \end{cases} \tag{6b}$$

where *t(v)* represents the bad, average, good, and very good for each value of $v = 2$, 3, 4, and 5, respectively.

$$P_{Exl}(m) = \begin{cases} 0, & m \le 4, \\ m - 4, & 4 < m \le 5. \end{cases} \tag{6c}$$

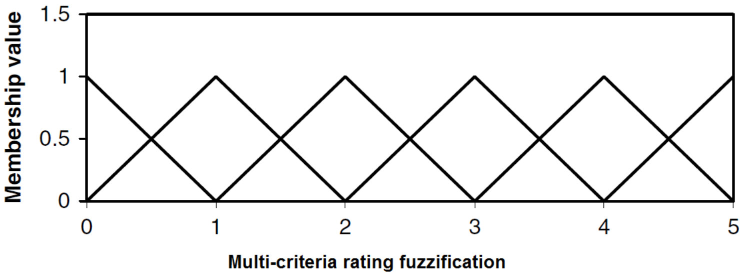


Figure 2. Fuzzy sets for multi-criteria rating side information

The 'user age' feature from MovieLens dataset is fuzzified into three fuzzy sets namely, young, middle-aged, and old [6], as shown in Figure 3, the membership functions of these fuzzy sets are shown below.

$$Q_{young}(m) = \begin{cases} 1, & m \le 20, \\ (35 - m)/15, & 20 < m \le 35, \\ 0, & m > 35, \end{cases} \tag{7a}$$

$$Q_{middle}(m) = \begin{cases} 0, & m \le 20, m > 60, \\ (m - 20)/15, & 20 < m \le 35, \\ 1, & 35 < m \le 45, \\ (60 - m)/15, & 45 < m \le 60, \end{cases} \tag{7b}$$

$$Q_{Old}(m) = \begin{cases} 0, & m \le 45, \\ (m - 45)/15, & 45 < m \le 60, \\ 1, & m > 60. \end{cases} \tag{7c}$$

Fuzzy distance function: After side information fuzzification process, to compute the distances between fuzzified features, we replace the Euclidean distance method in Algorithm 1 mentioned above in step *2(i)* with the following modified Euclidean fuzzy distance formula [17].

$$Gfd(X, Y) = \sqrt{\sum_{i=1}^{z} (Lfd(x_i, y_i))^2} \tag{8}$$

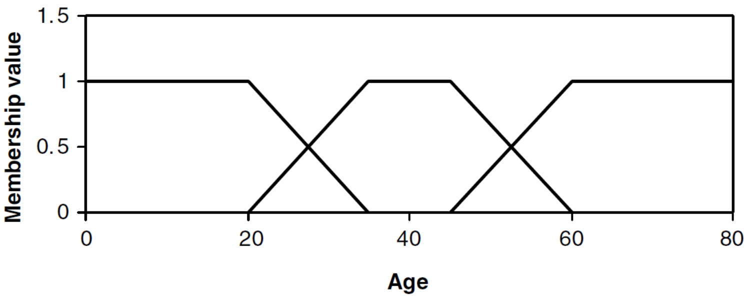


Figure 3. Fuzzy sets for age side information

where *Gfd* is the global fuzzy distance, z is the length of fuzzified vector and *Lfd* is the local fuzzy distance of users:

$$Lfd = dis(x_i, y_i) \times d(x_i, y_i). \tag{9}$$

Here, $d(x_i, y_i)$ computes the difference between vectors x and y of size m, and $dis(x_i, y_i)$ is an Euclidean distance function given shown below.

$$dis(x_j, y_j) = \sqrt{\sum_{i=1}^{m} (x_{i,j} - y_{i,j})^2} \tag{10}$$

where $x_{i,j}$ denotes the membership value of the i^{th} feature in the j^{th} fuzzy set.

3.2 Similarity Computation and Neighbors Selection

After creating groups, to generate most similar top-N neighbors of the target user, the next step is to compute similarities among different users within their respective clusters. The logic behind similarity computation is to extract those users who have provided similar ratings to the same products. As Mahalanobis distance method persists multiple benefits compared to the classical similarity methods, we choose this method to compute the distance between two users who have co-rated a product with a rating. Such as, the individual item's rating does not affect the distance as it only depends upon the variance and covariance of the total ratings given by user. Also, in multi-dimensional space, the MD works well by removing the scaling as well as the collinearity impact of the variables and then calculates the simple Euclidean distance between users. Moreover, there are very few research works published in the field of recommender systems which incorporates Mahalanobis distance for recommendations [15, 34]. The Mahalanobis distance between users can be calculated with the help of following equations.

Let us assume there are two users x and y and our task is to compute the distance between them, in such case, Equation (4) for two different users can be rewritten as [14]:

$$[(x-\bar{x})(y-\bar{y})]vc_z^{-1} = \begin{bmatrix} \frac{\sigma_2^2(x-\bar{x})-(y-\bar{y})\rho_{12}\sigma_1\sigma_2}{det(vc_z)} & \frac{\sigma_1^2(y-\bar{y})-(x-\bar{x})\rho_{12}\sigma_1\sigma_2}{det(vc_z)} \end{bmatrix}. \tag{11}$$

By multiplying $\begin{bmatrix}(x-\bar{x})\\(y-\bar{y})\end{bmatrix}$ in both side of the above Equation (11), we will get

$$\begin{aligned}[(x-\bar{x})(y-\bar{y})]vc_z^{-1}\begin{bmatrix}(x-\bar{x})\\(y-\bar{y})\end{bmatrix} &= \frac{\sigma_2^2(x-\bar{x})^2-(y-\bar{y})(x-\bar{x})\rho_{12}\sigma_1\sigma_2}{det(vc_z)} \\ &\quad + \frac{\sigma_1^2(y-\bar{y})^2-(x-\bar{x})(y-\bar{y})\rho_{12}\sigma_1\sigma_2}{det(vc_z)}\end{aligned} \tag{12}$$

$$= \frac{\sigma_2^2(x-\bar{x})^2(1-\rho_{12}^2)+\sigma_1^2(y-\bar{y}^2)-2(x-\bar{x})(y-\bar{y})\rho_{12}\sigma_1\sigma_2+\sigma_2^2(x-\bar{x})\rho_{12}^2}{\sigma_1^2\sigma_2^2(1-\rho_{12}^2)} \tag{13}$$

$$= \frac{(x-\bar{x})^2}{\sigma_1^2} + \frac{(y-\bar{y})^2}{\sigma_2^2(1-\rho_{12}^2)} - 2\frac{(x-\bar{x})(y-\bar{y})\rho_{12}}{\sigma_1\sigma_2(1-\rho_{12}^2)} + \frac{\rho_{12}^2(x-\bar{x})^2}{\sigma_1^2(1-\rho_{12}^2)} \tag{14}$$

$$= \frac{(x-\bar{x})^2}{\sigma_1^2} + \left[\frac{(y-\bar{y})}{\sigma_2\sqrt{1-\rho_{12}^2}} - \frac{\rho_{12}(x-\bar{x})}{\sigma_1\sqrt{1-\rho_{12}^2}}\right]^2. \tag{15}$$

After comparing Equation (15) with Equation (4), the Mahalanobis distance for users x and y will be

$$MD = \sqrt{\left(\frac{x-\bar{x}}{\sigma_1}\right)^2 + \left[\left\{\left(\frac{y-\bar{y}}{\sigma_2}\right) - \rho_{12}\left(\frac{x-\bar{x}}{\sigma_1}\right)\right\}\frac{1}{\sqrt{1-\rho_{12}^2}}\right]^2}. \tag{16}$$

The subtraction portion of the above formula is used to correct the correlation between the data. This equation will become a simple Euclidean distance method in case of uncorrelated variables. The above equation is limited to compute similarities between two users who have co-rated a single movie only. Therefore, we updated the formula so that it can calculate distances between multiple users and multiple products ($|items| \geq 1$).

$$MD(x,y) = \frac{\sum_{n\in N_{xy}}\sqrt{\left(\frac{r_{x,n}-\bar{x}}{\sigma_1}\right)^2 + \left[\left\{\left(\frac{r_{y,n}-\bar{y}}{\sigma_2}\right) - \rho_{12}\left(\frac{r_{x,n}-\bar{x}}{\sigma_1}\right)\right\}\frac{1}{\sqrt{1-\rho_{12}^2}}\right]^2}}{|N_{xy}|} \tag{17}$$

where N_{xy} is the set of co-rated products by both the users x and y. Rating of user x on product n is represented by $r_{x,n}$ whereas $\bar{x}$ represents the mean of the ratings given by the user x to all products. Finally, after similarity computation between users within the cluster, the top-N most similar users are selected for neighborhood formation.

3.3 Prediction and Recommendations

The collective ratings given by the users of the neighborhood set is used to predict the ratings of all unseen products for the target user [3, 6]. This method is used for all clusters to predict each user's unseen ratings. Finally, top predicted products can be recommended to the target user.

$$pre_{x,i} = \bar{r}_x + nf \sum_{x' \in B} dis(x, x') \times (r_{x',i} - \bar{r}_{x'}). \tag{18}$$

Here, $dis(x, x')$ is the distance between target user x and a neighborhood set member x', B is the neighborhood set of those users who have experienced product i earlier. The multiplier nf, a normalizing factor, is computed as:

$$nf = \frac{1}{\sum_{x' \in B} |dis(x, x')|}.$$

4 EXPERIMENTAL EVALUATION

4.1 Experiment Datasets

MovieLens 100 K dataset contains 943 users' ratings for 1 682 movies gathered by the GroupLens research laboratory at the University of Minnesota. The dataset consists of 100 000 ratings where each user has rated at least 20 movies. The ratings follow the 1-bad, 2-average, 3-good, 4-very good, and 5-excellent numerical scale. This dataset also contains the user-product background information like age, occupation, genre, etc. The sparsity level of the dataset is 93.69 %. Another dataset that we use is the Yahoo! Movies dataset, which contains 62156 ratings rated by 6 078 users on 780 movies. The dataset includes user ratings, movie criteria ratings, total number of movies rated by a user, and corresponding index of the movie which is rated. Additionally, each movie is associated with four different criteria namely story, acting, direction, and visuals, for which users have provided their ratings individually. This dataset follows rating scale from 1-bad to 13-excellent. Since MovieLens supports rating scale from 1-min to 5-max, therefore, we opt to normalize Yahoo! ratings in the same range to get similar range results. For normalization, we formed five rating groups $[\mathrm{Poor}(1,2,3), \mathrm{Fair}(4,5), \mathrm{Average}(6,7,8), \mathrm{Good}(9,10), \mathrm{Excellent}(11,12,13)]$ of these 1–13 ratings, such that, the average of each group $\{2, 4.5, 7, 9.5, 12\}$ have equal step size (i.e., 2.5 in our case). The sparsity level for this dataset is 98.69 %.

4.2 Experimental Settings

From the MovieLens dataset, we selected only those users who have rated at least 60 movies and discarded movies with zero ratings: 497 users and 1 682 movies satisfied this condition and contributed 84,596 ratings out of 100 000. Similarly, from Yahoo! Movies dataset, we extracted those users who have rated at least 20 movies. Where 484 users and 945 movies satisfied this condition and contributed 19 050 ratings out of 62 156. Furthermore, we divided each user's ratings randomly into training set and testing set in the percentage ratio of 70 % and 30 % respectively. The ratings in the training set are used for building the model and neighborhood set generation whereas the testing rating set is marked as unseen products of the target user. After calculating the similarity or distances among users' effectively, we selected top-30 users for the neighborhood set formation. The size of the neighborhood set is chosen experimentally. For k-means algorithm, we choose four cluster centers randomly as an initial point and repeated the cluster assignment process for 30 repetitions or until similar cluster appeared for three consecutive times. We have conducted multiple experiments on the following recommendation methods to demonstrate the effectiveness of the proposed scheme.

- Each User method: In this approach, each user's similarity is computed with every other user in the system. Therefore, we called this approach as the Each User method also known as the non-clustering method.
- Clustering with Side Information approach (Clust-SI): To build a cluster, the first thing to do is to identify the clustering parameters through which we can form similar user clusters. After multiple experiments, we choose 'User Age' feature as a clustering parameter from MovieLens and multi-criteria ratings from Yahoo! Movies datasets. We termed these selected parameters as the side information.
- Clustering with Fuzzy Side Information approach (Clust-FSI): This is the extended version of the Clust-SI approach where we apply fuzzy sets on the side information to deal with the uncertainty issue associated with them and obtain as close as possible neighborhood set for the target user.

4.3 Evaluation Matrices

In this paper, mean absolute error (MAE), root mean square error (RMSE), and coverage of the system evaluation matrices are used for evaluating the performance of the experimental methods. The motivation behind choosing these performance measures is their simplicity and vast use for measuring the effectiveness of RSs [3, 6, 17, 19]. The MAE calculates the average of the absolute differences between actual ($r_{k,j}$) and predicted user ratings ($pre_{k,j}$). The following formula gives the $MAE(k)$ for target user x_k:

$$MAE(k) = \frac{1}{p_k} \sum_{j=1}^{p_k} |pre_{k,j} - r_{k,j}| \tag{19}$$

where p_k is the cardinality of the test ratings set of user x_k. Whereas, RMSE method squares the error, therefore, its value grows faster than MAE when there is a big gap between actual and predicted values. Lower the MAE and RMSE infers more accurate predictions given by the system.

$$RMSE(k) = \sqrt{\frac{\sum_{j=1}^{p_k} (pre_{k,j} - r_{k,j})^2}{p_k}}. \tag{20}$$

The third evaluation metric which we used gives a total coverage of the system. Coverage evaluates a system by measuring the percentage of products for which a recommender system can provide predictions. Usually, a RSs may not be efficient to make predictions for every product in the system. Higher the coverage corresponds to better the prediction ability of the system.

$$Cov = \frac{\sum_{t=1}^{P_k} f_t}{\sum_{t=1}^{P_k} p_t} \tag{21}$$

where f_t is the total number of predicted products for target user x_t.

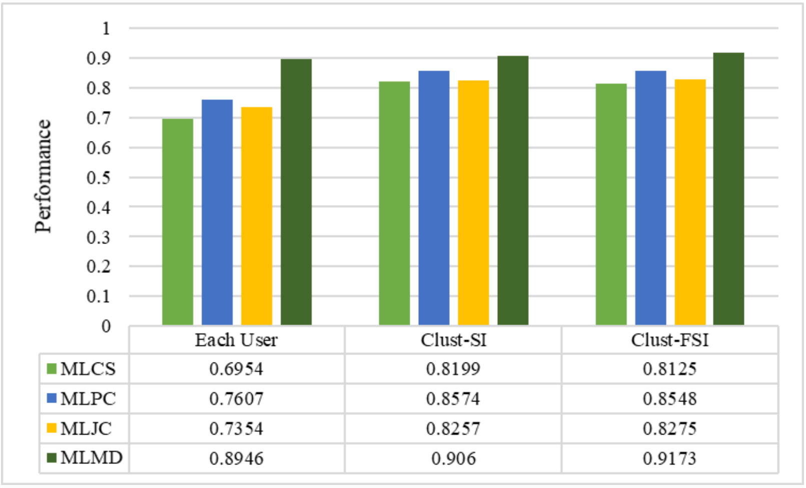
4.4 Experimental Results and Analysis

This section presents the results of the experiments conducted on two real datasets to the improvement of collaborative filtering recommender's performance. The goal of this experiment is to compare the results of the proposed method with other state-of-the-art CF methods under multiple evaluation metrics.

4.4.1 Comparing the Each-User and Clust-SI Methods

In this experiment, we run the Clust-SI approach and compare its results with the traditional Each-User approach. For Each-User, experiments are run for entire training users' database for each step. This type of computation is too time-consuming and sometimes it leads to overfitting problem. In our experiments, we follow the user partitioning technique to shorten the user space. We applied the k-means clustering algorithm to form user clusters by choosing side information as a clustering parameter. Further, the neighborhood set for users is obtained from their respective user clusters only. This agrees with our thinking of reducing the user search space and may reduce the chances of overfitting. Another benefit of the use of side information clustering is that it will alleviate the multi-dimensionality issue because we are not building a complex model by selecting the side information as a third dimension of the system. We picked the user age feature from the MovieLens dataset as the clustering parameter where users are grouped into multiple clusters using k-means clustering. Whereas, for Yahoo! Movies dataset, we selected the multi-criteria movie ratings as the clustering parameter where users with high (low) similar interests are grouped into same (different) groups.

Results summarized in following figures and tables show that Clust-SI outperforms Each-User for all the performance measures. Results of Clust-SI are much better than Each-User for every corresponding similarity approach. The comparing approaches are labelled with four letters (e.g., MLCS or YMCS), the first two letters represent the name of the dataset (i.e. ML for MovieLens and YM for Yahoo! Movies) and the remaining letters represent the name of the measure used (i.e. CS for Cosine-based similarity). The MAE and RMSE of Clust-SI are always smaller than the corresponding approaches of Each-User, as shown in Tables 1 and 2 for both MovieLens and Yahoo! Movies datasets, respectively. Whereas, the coverage is higher for all the comparing approaches.



	Each User	Clust-SI	Clust-FSI
🟩 MLCS	0.6954	0.8199	0.8125
🟦 MLPC	0.7607	0.8574	0.8548
🟨 MLJC	0.7354	0.8257	0.8275
🟩 MLMD	0.8946	0.906	0.9173

Figure 4. Comparison of coverage for different collaborative recommenders on ML dataset

4.4.2 Comparing the Clust-SI and Clust-FSI Methods

In this experiment, fuzzy logic is used to handle the uncertainty issue associated with the side information. The multi-criteria ratings and user age are fuzzified with the help of fuzzy function shown in Figures 4 and 5, respectively. The purpose of using fuzzy logic for side information is to get as close as possible to the set of users for the target user. Where a young user will be matched with other young users instead of the old users. It will generate more effective neighborhood set for the target user in comparison to the non-fuzzified methods. Since we clustered users based on their fuzzified side information, we call this approach as Clust-FSI

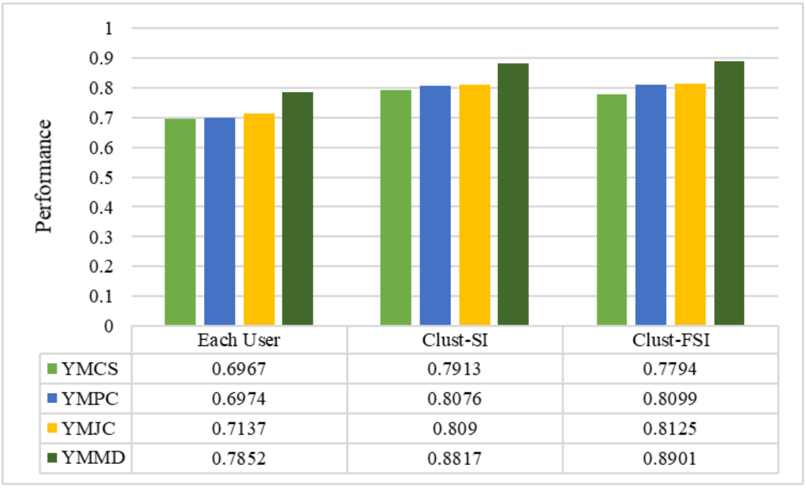
MAE	Non-Clustering	Clustering Based Approach	
	Each User	*Clust-SI*	*Clust-FSI*
MLCS	0.8498	0.8163	**0.8151**
MLPC	0.8544	**0.8196**	0.8199
MLJC	0.8304	**0.8143**	0.8146
MLMD	0.8142	0.8139	**0.8133**

a) Comparison of MAE for different collaborative recommenders

RMSE	Non-Clustering	Clustering Based Approach	
	Each User	*Clust-SI*	*Clust-FSI*
MLCS	1.0632	**1.0241**	1.0243
MLPC	1.07	**1.0286**	1.0325
MLJC	1.0413	**1.0215**	1.0237
MLMD	1.0194	1.0186	**1.0161**

b) Comparison of RMSE for different collaborative recommenders

Table 1. Performance on MovieLens Dataset



	Each User	Clust-SI	Clust-FSI
🟩 YMCS	0.6967	0.7913	0.7794
🟦 YMPC	0.6974	0.8076	0.8099
🟨 YMJC	0.7137	0.809	0.8125
🟩 YMMD	0.7852	0.8817	0.8901

Figure 5. Comparison of coverage for different collaborative recommenders on YM dataset

approach. The experimental results of this approach are compared with Each-User and Clust-SI approaches, as shown in above figures and tables. Results summarized in Tables 1 and 2 show that Clust-SI and Clust-FSI outperform Each-User method for both datasets. The MAE and RMSE for Each-User method are always higher than the clustering based approaches, therefore, we can infer that the use of the side information clustering has improved the performance of the system. Similarly, Figures 4 and 5 prove that the coverage of the clustering based approaches are higher than the non-clustering method where higher coverage shows the higher accuracy

MAE	Non-Clustering	Clustering based approach	
	Each user	*Clust-SI*	*Clust-FSI*
YMCS	0.9694	**0.89908**	0.9006
YMPC	0.9586	0.8938	**0.8833**
YMJC	0.9459	0.8957	**0.8803**
YMMD	0.8871	0.8822	**0.8675**

a) Comparison of MAE for different collaborative recommenders

RMSE	Non-Clustering	Clustering based approach	
	Each user	*Clust-SI*	*Clust-FSI*
YMCS	1.1923	1.109	**1.0207**
YMPC	1.1772	1.1094	**1.1006**
YMJC	1.1664	1.0957	**1.0866**
YMMD	1.0904	1.0839	**1.0735**

b) Comparison of RMSE for different collaborative recommenders

Table 2. Performance on Yahoo! Movies Dataset

of the system. Since non-clustering approach fails, as compared to the clustering based approaches, now we will analyze the performance of the clustering based approaches.

We compared the Clust-SI and Clust-FSI approaches based on four different similarity methods (CS, PC, JS, and MD) concerning MAE, RMSE, and Coverage of the system using two different datasets. Table 1 shows that the MAE and RMSE of MLPC and MLJC methods of Clust-SI are slightly better than the Clust-FSI on MovieLens dataset whereas the methods for Clust-FSI outperform the Clust-SI methods except the MAE of YMCS method for Yahoo! Movies dataset, as shown in Table 2. From Figures 4 and 5 we can see that the coverage of MLCS, MLPC and YMCS for Clust-SI approach is more accurate than the Clust-FSI methods. It means that these methods have greater ability to predict ratings as compared to the fuzzy-based clustering methods.

From the above-shown results we can infer that the clustering based technique has always improved the performance in comparison to the non-clustering methods. Furthermore, the Clust-SI approach shows more accurate results than the Clust-FSI approach for MovieLens dataset. In most of the cases, the Clust-FSI techniques outperform the non-fuzzified clustering approach for the Yahoo! Movies dataset. From these observations we can say that our proposed method can work more efficiently in case of sparse data (Yahoo! Movies dataset is sparser than MovieLens dataset). Moreover, the results prove that the Mahalanobis distance-based measure remains the best-performing method throughout the experiments for all performance measures on both datasets.

5 CONCLUSION

In this paper, we introduced a recommender system framework based on the side information clustering. We focus on the data sparsity and dimensionality issues using Mahalanobis distance and k-means clustering through

1. user's age and
2. multi-criteria movie ratings.

Our approach is based on the assumption that each user has a different opinion on different features. Therefore, to distinguish users, the prime concern of this work is to identify user segments with similar tastes. Fuzzy sets for side information have been applied to choose more accurate and reliable neighbors for the target user in each cluster. Moreover, we know that the traditional Cosine similarity and Pearson Correlation Coefficient methods have a shortage that they work only for commonly rated products and are not suitable for capturing user preferences at ground level. To overcome this problem, we used the well-known Mahalanobis distance method which considers user preferences from local to global level through the variance-covariance matrix. In experiments, we evaluated the effectiveness of our proposed algorithm on accuracy and recommendation performance improvement. The experimental results on two benchmark datasets with different side information demonstrated that our proposed fuzzy-based algorithm for Mahalanobis distance method (Clust-FSI – MD) has better performance compared with the classical recommendation algorithms. Additionally, our proposed framework can be seen as a cross-domain recommendation model because this compact model can be applied to different domains with different clustering parameters. For instance, the model which is used for multi-criteria movie recommendation can also be applied to context-aware music recommendation system. Therefore, in future work, we will try to apply our proposed framework to other domains where multi-dimensionality is the major issue.

REFERENCES

[1] 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.

[2] Su, X.—Khoshgoftaar, T. M.: A Survey of Collaborative Filtering Techniques. Advances in Artificial Intelligence, Vol. 2009, 2009, Art. No. 421425, 19 pp., doi: 10.1155/2009/421425.

[3] Wasid, M.—Kant, V.—Ali, R.: Frequency-Based Similarity Measure for Context-Aware Recommender Systems. 2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2016, pp. 627–632, doi: 10.1109/icacci.2016.7732116.

[4] Adomavicius, G.—Tuzhilin, A.: Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions. IEEE Transactions on Knowledge and Data Engineering, Vol. 17, 2005, No. 6, pp. 734–749, doi: 10.1109/tkde.2005.99.

[5] Domingues, M. A.—Jorge, A. M.—Soares, C.: Dimensions as Virtual Products: Improving the Predictive Ability of Top-N Recommender Systems. Information Processing and Management, Vol. 49, 2013, No. 3, pp. 698–720, doi: 10.1016/j.ipm.2012.07.009.

[6] Wasid, M.—Kant, V.: A Particle Swarm Approach to Collaborative Filtering Based Recommender Systems Through Fuzzy Features. Procedia Computer Science, Vol. 54, 2015, pp. 440–448, doi: 10.1016/j.procs.2015.06.051.

[7] Sahoo, N.—Krishnan, R.—Duncan, G.—Callan, J.: Research Note – The Halo Effect in Multicomponent Ratings and Its Implications for Recommender Systems: The Case of Yahoo! Movies. Information Systems Research, Vol. 23, 2012, No. 1, pp. 231–246, doi: 10.1287/isre.1100.0336.

[8] Li, Q.—Wang, C.—Geng, G.: Improving Personalized Services in Mobile Commerce by a Novel Multicriteria Rating Approach. Proceedings of the 17th International Conference on World Wide Web (WWW '08), 2008, pp. 1235–1236, doi: 10.1145/1367497.1367743.

[9] Liu, X.: An Improved Clustering-Based Collaborative Filtering Recommendation Algorithm. Cluster Computing, Vol. 20, 2017, No. 2, pp. 1281–1288, doi: 10.1007/s10586-017-0807-6.

[10] Sarwar, B. M.—Karypis, G.—Konstan, J.—Riedl, J.: Recommender Systems for Large-Scale E-Commerce: Scalable Neighborhood Formation Using Clustering. Proceedings of the Fifth International Conference on Computer and Information Technology (ICCIT), Vol. 1, 2002.

[11] Liu, L.—Mehandjiev, N.—Xu, D. L.: Multi-Criteria Service Recommendation Based on User Criteria Preferences. Proceedings of the Fifth ACM Conference on Recommender Systems, ACM, 2011, pp. 77–84, doi: 10.1145/2043932.2043950.

[12] Nilashi, M.—Jannach, D.—bin Ibrahim, O.—Ithnin, N.: Clustering- and Regression-Based Multi-Criteria Collaborative Filtering with Incremental Updates. Information Sciences, Vol. 293, 2015, pp. 235–250, doi: 10.1016/j.ins.2014.09.012.

[13] Nilashi, M.—bin Ibrahim, O.—Ithnin, N.—Sarmin, N. H.: A Multi-Criteria Collaborative Filtering Recommender System for the Tourism Domain Using Expectation Maximization (EM) and PCA–ANFIS. Electronic Commerce Research and Applications, Vol. 14, 2015, No. 6, pp. 542–562, doi: 10.1016/j.elerap.2015.08.004.

[14] De Maesschalck, R.—Jouan-Rimbaud, D.—Massart, D. L.: The Mahalanobis Distance. Chemometrics and Intelligent Laboratory Systems, Vol. 50, 2000, No. 1, pp. 1–18, doi: 10.1016/s0169-7439(99)00047-7.

[15] Komkhao, M.—Lu, J.—Li, Z.—Halang, W. A.: Incremental Collaborative Filtering Based on Mahalanobis Distance and Fuzzy Membership for Recommender Systems. International Journal of General Systems, Vol. 42, 2013, No. 1, pp. 41–66, doi: 10.1080/03081079.2012.710437.

[16] Frémal, S.—Lecron, F.: Weighting Strategies for a Recommender System Using Item Clustering Based on Genres. Expert Systems with Applications, Vol. 77, 2017, pp. 105–113, doi: 10.1016/j.eswa.2017.01.031.

[17] Al-Shamri, M. Y. H.—Bharadwaj, K. K.: Fuzzy-Genetic Approach to Recommender Systems Based on a Novel Hybrid User Model. Expert Systems with Applications, Vol. 35, 2008, No. 3, pp. 1386–1399, doi: 10.1016/j.eswa.2007.08.016.

[18] Shi, Y.—Lin, H.—Li, Y.: Context-Aware Recommender Systems Based on Item-Grain Context Clustering. In: Peng, W., Alahakoon, D., Li, X. (Eds.): AI 2017: Advances in Artificial Intelligence. Springer, Cham, Lecture Notes in Computer Science, Vol. 10400, 2017, pp. 3–13, doi: 10.1007/978-3-319-63004-5_1.

[19] Wasid, M.—Ali, R.—Kant, V.: Particle Swarm Optimisation-Based Contextual Recommender Systems. International Journal of Swarm Intelligence, Vol. 3, 2017, No. 2-3, pp. 170–191, doi: 10.1504/IJSI.2017.087874.

[20] Zhang, J.—Lin, Y.—Lin, M.—Liu, J.: An Effective Collaborative Filtering Algorithm Based on User Preference Clustering. Applied Intelligence, Vol. 45, 2016, No. 2, pp. 230–240, doi: 10.1007/s10489-015-0756-9.

[21] Ungar, L. H.—Foster, D. P.: Clustering Methods for Collaborative Filtering. In AAAI Workshop on Recommendation Systems, Vol. 1, 1998, pp. 114–129.

[22] Tsai, C.-F.—Hung, C.: Cluster Ensembles in Collaborative Filtering Recommendation. Applied Soft Computing, Vol. 12, 2012, No. 4, pp. 1417–1425, doi: 10.1016/j.asoc.2011.11.016.

[23] Wei, S.—Ye, N.—Zhang, S.—Huang, X.—Zhu, J.: Collaborative Filtering Recommendation Algorithm Based on Item Clustering and Global Similarity. 2012 Fifth International Conference on Business Intelligence and Financial Engineering (BIFE), 2012, pp. 69–72, doi: 10.1109/bife.2012.23.

[24] Nilashi, M.—Ibrahim, O.—Bagherifard, K.: A Recommender System Based on Collaborative Filtering Using Ontology and Dimensionality Reduction Techniques. Expert Systems with Applications, Vol. 92, 2018, pp. 507–520, doi: 10.1016/j.eswa.2017.09.058.

[25] Son, N. T.—Dat, D. H.—Trung, N. Q.—Anh, B. N.: Combination of Dimensionality Reduction and User Clustering for Collaborative-Filtering. Proceedings of the 2017 International Conference on Computer Science and Artificial Intelligence (CSAI 2017), ACM, 2017, pp. 125–130, doi: 10.1145/3168390.3168405.

[26] Nilashi, M.—Bin Ibrahim, O.—Ithnin, N.—Zakaria, R.: A Multi-Criteria Recommendation System Using Dimensionality Reduction and Neuro-Fuzzy Techniques. Soft Computing, Vol. 19, 2015, No. 11, pp. 3173–3207, doi: 10.1007/s00500-014-1475-6.

[27] Xu, S.—Watada, J.: A Method for Hybrid Personalized Recommender Based on Clustering of Fuzzy User Profiles. 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), 2014, pp. 2171–2177, doi: 10.1109/fuzz-ieee.2014.6891690.

[28] Wang, Z.—Yu, N.—Wang, J.: User Attributes Clustering-Based Collaborative Filtering Recommendation Algorithm and Its Parallelization on Spark. In: Zhang, L., Song, X., Wu, Y. (Eds.): Theory, Methodology, Tools and Applications for Modeling and Simulation of Complex Systems (AsiaSim 2016, SCS AutumnSim 2016). Springer,

Singapore, Communications in Computer and Information Science, Vol. 643, 2016, doi: 10.1007/978-981-10-2663-8_46.

[29] Kużelewska, U.—Guziejko, E.: A Recommender System Based on Content Clustering Used to Propose Forum Articles. In: Zamojski, W., Mazurkiewicz, J., Sugier, J., Walkowiak, T., Kacprzyk, J. (Eds.): Proceedings of the Ninth International Conference on Dependability and Complex Systems DepCoS-RELCOMEX. Springer, Cham, Advances in Intelligent Systems and Computing, Vol. 286, 2014, pp. 285–292, doi: 10.1007/978-3-319-07013-1_27.

[30] Boratto, L.—Carta, S.: Using Collaborative Filtering to Overcome the Curse of Dimensionality When Clustering Users in a Group Recommender System. Proceedings of the 16th International Conference on Enterprise Information Systems (ICEIS 2014), Vol. 2, 2014, pp. 564–572, doi: 10.5220/0004865005640572.

[31] Koohi, H.—Kiani, K.: A New Method to Find Neighbor Users That Improves the Performance of Collaborative Filtering. Expert Systems with Applications, Vol. 83, 2017, pp. 30–39, doi: 10.1016/j.eswa.2017.04.027.

[32] Najafabadi, M. K.—Mahrin, M. N.—Chuprat, S.—Sarkan, H. M.: Improving the Accuracy of Collaborative Filtering Recommendations Using Clustering and Association Rules Mining on Implicit Data. Computers in Human Behavior, Vol. 67, 2017, pp. 113–128, doi: 10.1016/j.chb.2016.11.010.

[33] Resnick, P.—Iacovou, N.—Suchak, M.—Bergstrom, P.—Riedl, J.: GroupLens: An Open Architecture for Collaborative Filtering of Netnews. Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work, ACM, 1994, pp. 175–186, doi: 10.1145/192844.192905.

[34] Wasid, M.—Ali, R.: An Improved Recommender System Based on Multi-Criteria Clustering Approach. Procedia Computer Science, Vol. 131, 2018, pp. 93–101, doi: 10.1016/j.procs.2018.04.190.

[35] Mahalanobis, P. C.: On the Generalized Distance in Statistics. Proceedings of the National Institute of Sciences of India, Vol. 2, 1936, No. 1, pp. 49–55.

Mohammed Wasid received his B.Tech. from Rajasthan Technical University and M.Tech. from The LNM Institute of Information Technology, Jaipur in 2012 and 2015, respectively. He is working toward the Ph.D. degree from Aligarh Muslim University, Aligarh. Currently he serves as Assistant Professor at the Department of Computer Science and Engineering, Government Engineering College, Bharatpur. His main research interests include recommender systems, evolutionary algorithms, and machine learning.

Rashid Ali received his B.Tech. and M.Tech. degrees from the AMU Aligarh, India in 1999 and 2001, respectively. He received his Ph.D. in computer engineering in 2010 from the AMU Aligarh. He has authored more than 100 papers in various international journals and international conference proceedings. He has presented papers in many international conferences and has also chaired sessions in few international conferences. He has reviewed articles for some of the reputed international journals and international conference proceedings. His research interests include web-searching, web-mining, recommender systems, online social network analysis, and soft computing techniques.

Computing and Informatics, Vol. 38, 2019, 621–646, doi: 10.31577/cai_2019_3_621

DISCOVERING FOREIGN KEYS ON WEB TABLES WITH THE CROWD

Xiaoyu Wu, Ning Wang*, Huaxi Liu

School of Computer and Information Technology, Beijing Jiaotong University
Beijing 100044, China
e-mail: {16112087, nwang, 13120407}@bjtu.edu.cn

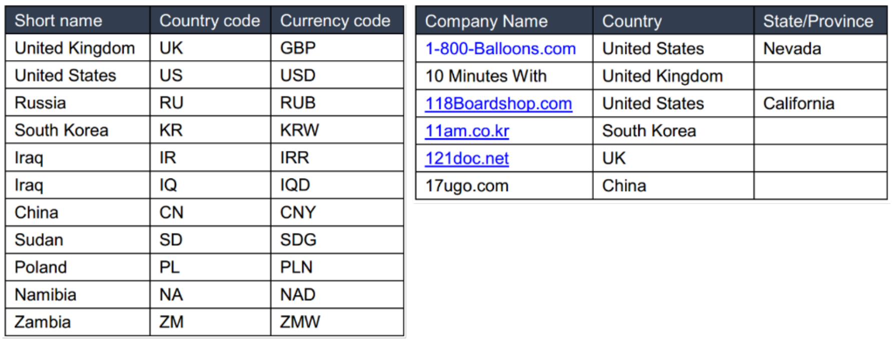
Abstract. Foreign-key relationship is one of the most important constraints between two tables. Previous works focused on detecting inclusion dependencies (INDs) or foreign keys in relational database. To discover foreign-key relationship is obviously helpful for analyzing and integrating data in web tables. However, because of poor quality of web tables, it is difficult to discover foreign keys by existing techniques based on checking basic integrity constraints. In this paper, we propose a hybrid human-machine framework to detect foreign keys on web tables. After discovering candidates and evaluating their confidence of being true foreign keys by machine algorithm, we verify those candidates leveraging the power of the crowd. To reduce the monetary cost, a dynamical task selection technique based on conflict detection and inclusion dependency is proposed, which could eliminate redundant tasks and assign the most valuable tasks to workers. Additionally, to make workers complete tasks more effectively and efficiently, sampling strategy is applied to minimize the number of tuples posed to the crowd. We conducted extensive experiments on real-world datasets and results show that our framework can obviously improve foreign key detection accuracy on web tables with lower monetary cost and time cost.

Keywords: Foreign key, web tables, crowdsourcing, task selection, task reduction, semantic recovery

* Corresponding author

1 INTRODUCTION

Foreign-key relationship is one of the most important constraints between two tables. Previous works focus on detecting inclusion dependencies (abbr. INDs) or foreign keys in relational database [1, 2, 3, 4], in which table name, uniqueness of key and strict inclusion dependency between foreign key (the dependent attribute) and primary key (the referenced attribute) are important factors for detecting foreign keys. The worldwide web contains a vast amount of tables on varieties of topics [5], and to discover foreign-key relationship is obviously helpful for analyzing and integrating data in web tables. Unfortunately, all the previous works could not be used directly on web tables which often lose table names and sometimes have noisy data. In fact, web tables may not satisfy the entity integrity constraint and referential integrity constraint. Figure 1 shows fragments of typical web tables from Google Table [6] which miss table names and have duplicated tuples. There is a foreign key relationship between tables in Figure 1, where country in Figure 1 b) is a foreign key referencing short name in Figure 1 a). However, foreign key detection method in relational database could not be used in such tables which lose some schema information and also do not satisfy basic integrity constraints. Furthermore, even if a web table has a table name, it often does not include meaningful information which could describe the semantics of this table exactly. Because of poor quality of web tables, it is difficult to discover foreign keys effectively only by machine algorithm.



Short name	Country code	Currency code
United Kingdom	UK	GBP
United States	US	USD
Russia	RU	RUB
South Korea	KR	KRW
Iraq	IR	IRR
Iraq	IQ	IQD
China	CN	CNY
Sudan	SD	SDG
Poland	PL	PLN
Namibia	NA	NAD
Zambia	ZM	ZMW

a)

Company Name	Country	State/Province
1-800-Balloons.com	United States	Nevada
10 Minutes With	United Kingdom	
118Boardshop.com	United States	California
11am.co.kr	South Korea	
121doc.net	UK	
17ugo.com	China	

b)

Figure 1. An example of web tables

Recent researches have shown that crowdsourcing could be used effectively to solve problems that are difficult for computers, such as entity resolution [7], sentiment analysis [8], and image recognition [9]. We propose a hybrid human-machine framework that leverages human intelligence to discover foreign keys on web tables effectively. Our framework implements foreign key detection in two phases, which are finding candidates by machine algorithm and validating candidates by the crowd.

The first phase is for candidate generation. Because web tables may not satisfy the entity integrity constraint and referential integrity constraint, we define uniqueness degree to measure the proportion of unique values in a column and coverage rate to measure the proportion of dependent attribute values contained in the referenced attribute. Then we use four features to evaluate the possibility of a candidate being a true foreign key which are the unique degree of the referenced attribute, the coverage rate of the referenced attribute to the dependent attribute, the column names' similarity and whether the dependent attribute is a key. Short of semantics, it is so difficult for computer to understand relationships between two tables that will result in some false positive candidates. Fortunately, with the intelligence of humans, those false positive candidates can be easily distinguished.

After the first phase, candidates are generated inevitably with some false positives, so crowdsourcing is used for distinguishing true foreign keys from all candidates. As the crowd is not free, cost control is one of the biggest challenges in data management with the crowd [10]. To reduce the monetary cost, number of tasks should be reduced. Considering some conflicts in candidates and inclusion dependency between dependent attributes, we propose dynamical task selection methods based on conflict detection and inclusion dependency. The experimental results show our method can effectively reduce the number of tasks.

Besides the monetary cost, time cost is also to be considered for crowdsourcing tasks. For foreign key validation, workers have to check content between two tables. Facing tables with too many tuples, workers will be impatient with taking long time to browse the whole table and make decision. So, to reduce the latency of tasks, we propose a task reduction method based on sampling strategy, which could reduce the volume of web tables under the condition that the original relationship between tables could be held.

The main contributions of this paper are:

- To our best knowledge, we are the first to propose a hybrid human-machine framework for discovering foreign keys on web tables which may not satisfy the entity integrity constraint and referential integrity constraint.
- To reduce monetary cost for crowdsourcing tasks, we propose a dynamical task selection technique based on conflict detection and inclusion dependency, which could eliminate redundant tasks and assign the most valuable tasks to workers.
- To avoid latency of tasks, we propose a task reduction method based on sampling strategy to minimize the number of tuples posed to the crowd.
- Based on real-world datasets, we evaluated the performance of human-machine hybrid approach and effectiveness of our dynamical task selection method and task reduction method.

The remainder of the paper is organized as follows. We present solution overview in Section 2. Section 3 gives the machine algorithm for generating and scoring foreign key candidates on web tables. Our dynamical task selection method and task reduction method are discussed in Section 4 and Section 5, respectively. Then we report results of experiment in Section 6, discuss related works in Section 7, and conclude the paper in Section 8.

2 SOLUTION OVERVIEW

To discover foreign keys on web tables, we propose a hybrid human-machine framework. Our framework takes as input a set of web tables and generates foreign key candidates by machine algorithm. Then the false positive candidates are verified by the crowd and true foreign keys are output. Figure 2 shows the framework.

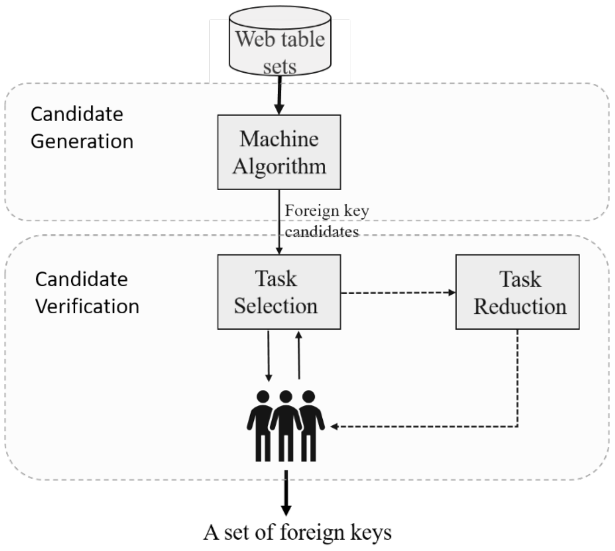


Figure 2. A hybrid human-machine framework for discovering foreign keys on web tables

There are two phases in our framework. In the first phase, machine algorithm is used to find candidates on input tables and calculate their confidences to be true foreign keys (the details will be described in Section 3). Then candidates with high confidence will be verified by the crowd in the second phase. As the crowd is not free, we take measures to reduce monetary cost by reducing the number of tasks. Because there are some tasks that could be deduced by other tasks, we dynamically select tasks based on conflict detection (Section 4.1) and inclusion dependency (Section 4.2). The task selection method based on conflict detection reduces tasks in conflict with those verified as true, while the task selection method based on inclusion dependency reduces tasks which can be deduced by those verified with the method. Browsing a whole table with a large volume will surely make workers impatient and lead a high latency. So, we try to reduce tables' volume leveraging a combinational sampling strategy (Section 5). After steps above, only most valuable tasks are posted to the crowd through the crowdsourcing platform, and the final verified results are returned.

3 FOREIGN KEY CANDIDATE GENERATION

In our framework for discovering foreign keys on web tables, generating candidates is the first step.

Definition 1 (Uniqueness Degree)**.** Given table S and its attribute $S.b$, the uniqueness degree of $S.b$, denoted as $UNI(S.b)$, is the ratio of $S.b$'s cardinality to S's cardinality.

$$UNI(S.b) = \frac{|S.b|}{|S|}. \tag{1}$$

Definition 2 (Coverage Rate)**.** Given a pair of tables R, S and attributes $R.a$, $S.b$ in R and S, respectively, the coverage rate of $S.b$ on $R.a$, denoted as $COV(S.b, R.a)$, could be calculated using following formula:

$$COV\,(S.b, R.a) = \frac{|R.a \cap S.b|}{|R.a|}. \tag{2}$$

We start the detection from measuring the confidence of attribute pairs to be true foreign keys. As web tables may lose or duplicate some cells or tuples, we relax checking the uniqueness of key and containment relationship between key and foreign key which are necessary conditions for foreign key detection in relational database. Let δ be the threshold of primary keys' uniqueness degree, λ be the threshold of primary keys' coverage rate to the foreign key, and $p = (R.a, S.b)$ denote a pair of attributes where R and S are corresponding tables. For attribute pair $(R.a, S.b)$ with $UNI(S.b) \geq \delta$ and $COV(S.b, R.a) \geq \lambda$, we use a scoring function $CTF(R.a, S.b)$ to measure the attribute pair's confidence to be a true foreign key. The scoring function is a weighted sum of 4 scores corresponding to 4 features as follows:

S.b's unique degree: $Score_1 = UNI(S.b)$.

S.b's coverage to R.a: $Score_2 = COV(S.b, R.a)$.

The similarity between attribute name of R.a and S.b: $Score_3 = Sim(R.a, S.b)$.

Whether R.a is a key: $Score_4 = 1$ if R.a is a key $(UNI(R.a) \geq \delta)$ otherwise $Score_4 = 0$.

$$CTF\,(R.a, S.b) = \sum_{i=1}^{4} \omega_i Score_i. \tag{3}$$

In Equation (3), $0 < \omega_1, \omega_2, \omega_3 < 1, \omega_4 < 0$. If $CTF(R.a, S.b)$ is higher than the threshold of confidence, $(R.a, S.b)$ is recognized as a foreign key candidate and denoted as $R.a \xrightarrow{\delta,\lambda} S.b$.

An important problem in this step is how to evaluate the string similarity. Generally, the similarity matching algorithm could be accurate matching and fuzzy matching. Accurate matching is usually used in traditional inclusion dependencies discovery in relational database [1, 2, 4]. For web tables often with noisy data, we use edit distance to evaluate the similarity between attributes' values and Jaro Winkler Distance [11] to measure similarity between attributes' names.

Though we try our best to improve accuracy of the candidate generation method, there are still many false positive candidates in the result. Therefore, we decide to utilize human intelligence to find true foreign keys from candidates.

4 DYNAMICAL TASK SELECTION

For discovering foreign keys on web tables, we adopt a human-machine hybrid approach which first uses machine algorithm to generate a foreign key candidate set, and then ask humans to verify candidates in the set as either foreign-key or non-foreign-key. As the crowd is not free, cost control is one of the most important problems in crowdsourced data management, and appropriate task selection will surely make the crowd work more efficiently. For reducing monetary cost, we dynamically detect redundant tasks and assign the most valuable tasks to workers. In this section, we propose the dynamical task selection method based on conflict detection and inclusion dependency between dependent attributes.

Since, in our setting, some candidates will be verified by crowd, and others will be deduced by the task selection method. We call the former as *crowdsourced*(*labeled*) *candidates*, and the latter as *deduced*(*labeled*)*candidates*.

4.1 Task Selection Based on Conflict Detection

In a list of foreign key candidates, there often exist some conflicts. If a candidate is verified to be true, its conflicts must be false. We could utilize conflict relationship between candidates to reduce number of crowdsourcing tasks.

Definition 3 (Foreign Key Candidates Reference Graph)**.** Given a set of foreign key candidates FC, a foreign key candidates reference graph is a weighted directed graph $FKRG = \langle \zeta, \varphi, \omega \rangle$, where:

- ζ is a set of attributes occurred in FC.
- φ is a set of foreign key candidates, and $\langle R.a, S.b \rangle \in \varphi$ iff $R.a \xrightarrow{\delta,\lambda} S.b \in FC$.
- ω is a set of weights, each of which corresponds to confidence on a foreign key candidate in φ.

Figure 3 is an example of foreign key candidates reference graph. Given two attributes $A.a$ and $B.b$, if $A.a$ is a candidate foreign key referencing $B.b$, there will be a directed edge from $A.a$ to $B.b$, this edge's weight (i.e. 0.53) represents the confidence of the foreign key candidate $R.a \xrightarrow{\delta,\lambda} S.b$.

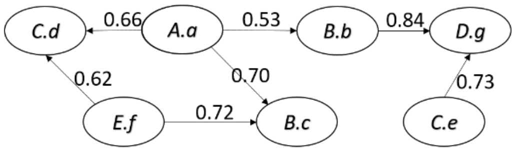


Figure 3. Foreign key candidates reference graph

A foreign key candidates reference graph is made up of vertexes and edges. While edges have weights with them reflecting the confidence of candidates, vertexes should be given weights reflecting average confidence of candidates related. The weight of a vertex is defined in Definition 4 as its influence.

Definition 4 (Influence of Attributes)**.** Given a foreign key candidates reference graph $FKRG = \langle \zeta, \varphi, \omega \rangle$, $T.m \in \zeta$, $RS(T.m) = \left\{ R.a \xrightarrow{\delta,\lambda} S.b | R.a = T.m \text{ or } S.b = T.m \right\}$, the influence of $T.m$, denoted as *Influence*$(T.m)$, could be calculated by the following formula:

$$\mathit{Influence}\,(T.m) = \frac{\sum_{i=1}^{|RS(T.m)|} CTF\,(R.a, S.b)}{|RS\,(T.m)|} \tag{4}$$

where $R.a \xrightarrow{\delta,\lambda} S.b \in RS(T.m)$.

For example, in Figure 3, $\mathit{Influence}\,(A.a) = \frac{CTF(A.a,B.b)+CTF(A.a,B.c)+CTF(A.a,C.d)}{3} = \frac{0.66+0.53+0.70}{3} = 0.63$. Intuitively, if a vertex has high influence, candidates related to this vertex may have high confidence of being true foreign keys and are more likely to be verified as a true foreign key.

In a true foreign key relationship, the dependent attribute couldn't be contained in the set of values of many other attributes, while the referenced attribute couldn't be referenced by multiple attributes from one table. Combining with web tables' characteristics, we get conflict detection rules below.

Conflict Detection Rules:

- A foreign key can only reference one primary key in the same referenced table.
- A primary key can only be referenced by one foreign key in the same dependent table.

Figure 4 gives examples about conflict rules. In Figure 4 a), there are foreign key candidates $R.a \xrightarrow{\delta,\lambda} S.b$ and $R.a \xrightarrow{\delta,\lambda} S.c$. In case any candidate is verified to be a true foreign key, another will be ruled out. This case indicates the similarity of b and c is very high, and there exists data redundancy in S. In Figure 4 b), there are foreign key candidates $R.a \xrightarrow{\delta,\lambda} S.b$ and $R.c \xrightarrow{\delta,\lambda} S.b$. In the same way, when any candidate is verified to be a true foreign key, another will be removed. This case indicates the similarity of attribute $R.a$ and $R.c$ is very high, and there exist data redundancy in R.

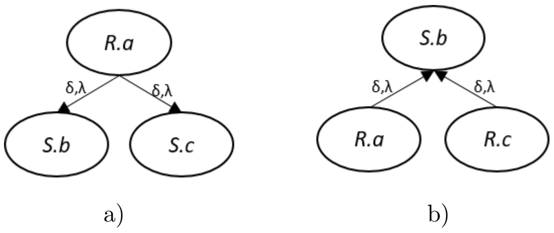


Figure 4. Conflicts of foreign key candidates

Based on the conflict detection rule, we propose to deduce unlabeled candidates with the labeled ones. Details of this candidate deduction algorithm are shown in Algorithm 1.

Algorithm 1 Candidate Deduction Based on Conflict Detection

```
Input: $LC$: a set of candidates that have been labeled as true foreign keys;
       $uc$: an unlabeled candidate;
Output: $rc$: the deduced result of $uc$;
 1: begin
 2: $rc \leftarrow null$;
 3: for $\forall c \in LC$ do
 4:    $Conf \leftarrow false$;
 5:    $F \leftarrow AttrSame(c.ref, uc.ref)$;
 6:    if $F$ then
 7:       $Conf \leftarrow TableSame(c.dep.t, uc.dep.t)$;
 8:    else
 9:       $F \leftarrow AttrSame(c.dep, uc.dep)$;
10:       if $F$ then
11:          $Conf \leftarrow TableSame(c.ref.t, uc.ref.t)$;
12:       end if
13:    end if
14:    if $Conf$ then
15:       $rc \leftarrow lable(uc, false)$;
16:    end if
17: end for
18: return $rc$;
19: end
```

Given a set of candidates LC that have been labeled as true foreign keys and an unlabeled candidate uc, for each labeled candidate in LC, this algorithm check whether its referenced attribute is the same with the referenced attribute of unlabeled candidate (lines 3–5). If it is the same, check whether their dependent attributes are from the same table, and label the unlabeled candidate as a conflict if the checking result is true (lines 6–7). Otherwise, check whether its dependent attribute is the same with the dependent attribute of unlabeled candidate or not, if it is the same, check whether their referenced attributes are from the same table, and label the unlabeled candidate as a conflict if the checking result is true (lines 8–13). The unlabeled candidate will be deduced as a non-foreign-key if it is labeled as conflict candidate (lines 14–16). Finally, the deduced result will be returned (lines 18–19).

Task selection aims at reducing crowdsourcing cost by reducing the number of tasks (i.e. crowdsourced candidates). Using the candidate deduction algorithm based on conflict detection, we dynamically reduce tasks (i.e. candidates) which can be deduced and select the most valuable task to be crowdsourced. When publishing tasks to crowdsourcing platform, we should give priority to the candidate which is most likely to be a true foreign key. Once a true foreign key is confirmed, other candidates conflicting with it will be removed from the task list.

The dynamical task selection method based on conflict selects tasks from two perspectives. From the global perspective, it chooses the high-influence vertex first. Then it chooses the foreign key candidates with high confidence first from the local perspective. Details of this method are shown in Algorithm 2.

Algorithm 2 Dynamical Task Selection Method Based on Conflict

```
Input: $G$: a foreign key candidates reference graph;
Output: $R$: the labeled result of candidates in G;
1: begin
2: $flag \leftarrow hasEdge(G)$;
3: while $flag$ do
4:     $v_h \leftarrow getTopV(G)$;
5:     $CS \leftarrow asDepAttr(v_h)$;
6:     $cfk_h \leftarrow getTopE(CS)$;
7:     $r \leftarrow crowd(cfk_h), R \leftarrow crowd(cfk_h)$;
8:     $elimEdge(cfk_h, G)$;
9:     if $r$ then
10:        $R \leftarrow deduceCan(r, CS), cc \leftarrow confCan(r, CS)$;
11:        $elimEdge(cc, G)$;
12:    end if
13: end while
14: return $R$;
15: end
```

Given a foreign key candidates reference graph G, we first calculate each vertex's weight and select the one (denoted as v_h) with the highest weight (lines 2–4). Then from the foreign key candidate set where v_h is a dependent attribute, we choose the candidate cfk_h with the highest confidence to be verified by the crowd (lines 5–7). After getting the verification result, we eliminate the corresponding edge of cfk_h from G (line 8). If cfk_h is verified to be a true foreign key, unlabeled candidates will be deduced using Algorithm 1, and corresponding edges will be removed (lines 9–11). Above steps are repeated until there is no edge in G, and then the labeled result of candidates in G will be returned (lines 13–15).

4.2 Task Selection Based on Inclusion Dependency

In addition to conflicts, there may be an inclusion dependency between dependent attributes which refer to the same referenced attribute. In this section, we introduce another task selection method based on inclusion dependency.

Suppose there are three foreign key candidates $A.a \xrightarrow{\delta,\lambda} B.b$, $C.c \xrightarrow{\delta,\lambda} B.b$ and $A.a \xrightarrow{\delta,\lambda} D.d$ that have been verified to be true foreign keys, and $COV(B.b, A.a) = COV(B.b, C.c) = COV(D.d, A.a) = 1$, i.e. $A.a \subseteq B.b$, $C.c \subseteq B.b$, $A.a \subseteq D.d$. If $C.c \subseteq A.a$, it is easy to get $C.c \subseteq D.d$. Because foreign key is a semantic relationship between attributes [1], we can get the conclusion that

1. the semantics of $B.b$ is similar to the semantics of $A.a$ and $C.c$, and
2. the semantics of $A.a$ is similar to the semantics of $D.d$.

Thus, we infer that

1. $A.a$ and $C.c$ are semantically related, and
2. $C.c$ and $D.d$ are semantically related.

Based on the above conditions, the candidate $C.c \xrightarrow{\delta,\lambda} D.d$ could be deduced as a true foreign key. This discovery gives us an inspiration of deducing candidates based on inclusion dependency between dependent attributes which is the core of this task selection method (based on inclusion dependency).

Next, we will describe the candidate deduction method based on inclusion dependency. Algorithm 3 gives the details of the method.

Given a set of candidates LC that have been labeled as true foreign keys and an unlabeled candidate uc, denote the referenced attribute and dependent attribute of uc as $P.k$ and $F.k'$, respectively (line 2), the algorithm first check whether there is any labeled candidate (denoted as c) of which the dependent attribute is the same with $F.k$ and coverage equals to 1 (lines 3–6). If any, it will try to discover another two labeled candidates (denoted as c' and c'') which have the same dependent attribute and satisfy

1. referenced attribute of c' is the same with referenced attribute of c and referenced attribute of c'' is the same with $P.k$,
2. the coverage of c' and c'' are equal to 1,
3. all values in $F.k'$ are a subset of dependent attribute of c'.

Algorithm 3 Candidate Deduction Based on Inclusion Dependency

```
Input: $LC$: a set of candidates that have been labeled as true foreign keys;
       $uc$: an unlabeled candidate;
Output: $rc$: the deduced result of $uc$;
1: begin
2: $rc \leftarrow null$, $P.k \leftarrow uc.refAttrrc$, $F.k' \leftarrow uc.depAttr$;
3: for $\forall c \in LC$ do
4:   if $c.depAttr == F.k$ && $c.coverage == 1$ then
5:     $C.r \leftarrow c.refAttr$, $LC' \leftarrow LC - \{c\}$;
6:   end if
7:   $S \leftarrow Pair2Can(LC')$;
8:   while $S$ do
9:     $s \leftarrow getEle(S)$, $c' \leftarrow s.c_1$, $c' \leftarrow s.c_2$, $remove(s)$;
10:    if $c'.depAttr == c''.depAttr$ && $c'.refAttr == C.r$ && $c'.refAttr ==$
       $P.k$ && $c'.coverage == c''.coverage == 1$ && $Fk' \subseteq c'.depAttr$ then
11:      $rc \leftarrow label(uc, true)$;
12:      break;
13:    end if
14:  end while
15:  if $rc$ then
16:    break;
17:  end if
18: end for
19: return $rc$;
20: end
```

If these candidates exist, then uc will be deduced as a true foreign key (lines 7–14). Finally, the deduced result will be returned (lines 15–19).

Example 1. For an unlabeled candidate $C.c \xrightarrow{\delta,\lambda} D.d$ with $COV(D.d, C.c) = 1$ (i.e. $C.c \subseteq D.d$), we check whether there is any candidate $C.c \xrightarrow{\delta,\lambda} B.b$ with $COV(B.b, C.c) = 1$ has been labeled as true. If any, we try to detect the labeled candidates which have the same dependent attribute ($T.m$) and satisfy the following conditions:

1. $T.m$ reference to $B.b$ and $D.d$ at the same time,
2. $COV(B.b, T.m) = COV(D.d, T.m) = 1$,
3. $C.c \subseteq D.d$.

If these candidates exist, then the candidate $C.c \xrightarrow{\delta,\lambda} D.d$ can be deduced as a true foreign key.

Generally, candidates deduced with high confidence are more credible. So, we are inclined to let candidates with low confidence be crowdsourced and candidates with high confidence to be deduced. Given a set of candidates to be verified, the task selection method based on inclusion dependency sorts them by their confidence in increasing order and gives priority to candidates with higher confidence first. Each time, candidate with the lowest confidence will be checked whether it can be deduced based on inclusion dependency. If not, its corresponding task will be posted to the crowdsourcing platform, otherwise, it will be deduced as a true foreign key.

4.3 A Combined Task Selection Method

To reduce the number of tasks as much as possible, we combine the task selection method based on conflict detection and inclusion dependency in practice. The above task selection methods (in Sections 4.1 and 4.2) publish the most valuable candidate to the crowdsourcing platform. Hence, long latency would be caused when only one candidate is posted to crowd at a time. To overcome this drawback, the combined task selection method places multiple candidates into a single task.

The core of the combined method is detecting redundant candidates in the task. There are two kinds of redundant candidates, certainly redundant candidates which could be deduced by labeled candidates and probably redundant candidates which are probably deduced by other unlabeled candidates in the same task.

Definition 5 (Certainly redundant candidate)**.** Given a set of labeled candidates C and an unlabeled candidate uc which will be crowdsourced, if uc could be deduced by candidates in C with any candidate deduction method (i.e. candidate deduction based on conflict detection or inclusion dependency), we say uc is a certainly redundant candidate.

Definition 6 (Probably redundant candidates)**.** Let UC be a set of unlabeled candidates which will be crowdsourced together in a single task, and uc is one of them, suppose all candidates except uc will be labeled as true foreign keys by crowd. Let C' be the labeled result set. If uc could be deduced by any candidates in C' with any candidate reduction method, we say uc is a probably redundant candidate in UC.

Next, we will introduce how to check and deal with two kinds of redundant candidates in a single task. Suppose k unlabeled candidates will be placed into a single task each time. For each unlabeled candidate, we need to check whether it could be deduced by labeled candidates first. If any, it will be recognized as a certainly redundant candidate and be deduced. A certainly redundant candidate will be removed and never be crowdsourced. Then the unlabeled candidates left should be checked whether they are probably redundant candidates in the task. For each candidate left, we suppose all the others will be labeled as true foreign keys and check whether it could be deduced with any candidate deduction method. If not, it will be recognized as a valuable candidate, otherwise, it will be recognized as a probably redundant candidate. Only valuable candidates will be placed into the crowdsourcing task, and probably redundant candidates will be hold on. After crowdsourcing results coming up, some probably redundant candidates might be deduced, and others not be deduced will be added into the unlabeled candidate set again. Algorithm 4 shows the details of the redundant candidate detection method.

Algorithm 4 Redundant Candidates Detection

```
Input: $LC$: a set of candidates which have been labeled as true foreign keys;
    $ULC$: a set of candidates that need to be crowdsourced;
Output: $RLC$: a candidate set without redundant candidate
 1: begin
 2: if $\exists c \in LC$ then
 3:     $ULC \leftarrow DeduceConflict(LC, ULC)$;
 4:     $ULC \leftarrow DeduceIND(LC, ULC)$;
 5: end if
 6: $Temp \leftarrow Label(ULC, true)$;
 7: $ULC \leftarrow DeduceConflict(Temp, ULC)$;
 8: $RLC \leftarrow DeduceIND(Temp, ULC)$;
 9: return $RLC$;
10: end
```

Given a set of candidates that need to be crowdsourced, we first deduce certainly redundant candidates with the candidate deduction methods (lines 2–5). Then we suppose all candidates left will be labeled as true foreign keys (line 6) and detect probably redundant candidates, i.e. detect whether they could be deduced by other candidates when they are labeled as true foreign keys (lines 7–8). Here, the procedure DeduceConflict is used to deduce candidates based on conflict detection, and the procedure DeduceIND is used to deduce candidates based on inclusion dependencies. Finally, the candidate set without redundant candidate will be returned (line 9).

Algorithm 5 gives the details of the combined task selection method.

Given a set of unlabeled foreign key candidates, we first sort them by their confidence in increasing order and select the top-k candidates to make up a task (lines 2–5). Then we remove the redundant candidates in the task and check the number of tasks (lines 8–9). If the number is less than k, we update the tasks (i.e. add new unlabeled tasks into the task) and repeat steps above until the number of candidates in the task is not less than k (lines 10–17).

Example 2. Consider the foreign key candidates in Figure 3. Suppose 5 candidates are contained in a task, $COV(C.d, A.a) = COV(C.d, E.f) = COV(B.c, A.a) = 1$ and $E.f \subseteq A.a$. Firstly, we sort these candidates by their confidence in increasing order and get the top-5 candidates, i.e., $A.a \xrightarrow{\delta,\lambda} B.b$, $E.f \xrightarrow{\delta,\lambda} C.d$, $A.a \xrightarrow{\delta,\lambda} C.d$, $A.a \xrightarrow{\delta,\lambda} B.c$ and $E.f \xrightarrow{\delta,\lambda} B.c$. For there is no labeled candidate, we just detect probably redundant candidate. For each candidate, suppose all the others will be labeled as true foreign keys. $A.a \xrightarrow{\delta,\lambda} B.c$ and $A.a \xrightarrow{\delta,\lambda} B.b$ are conflict candidates. In this case, we should crowdsource candidates with high confidence first. So, $A.a \xrightarrow{\delta,\lambda}$

Algorithm 5 The Combined Task Selection Method

```
Input: $LC$: a set of candidates that have been labeled as true foreign keys;
       $ULC$: an unlabeled candidate set;
       $k$: the number of candidates that need to be placed into a task;
Output: $T$: the task that needs to be crowdsourced;
1: begin
2: if $\exists c \in ULC$ then
3:    $S \leftarrow SortInc(ULC)$;
4:    $T \leftarrow getTopK(S)$;
5:    $RD \leftarrow true$;
6:    while $RD$ do
7:       $T \leftarrow RedundancyDetect(T, LC)$;
8:       if $Num(T) < k$ then
9:          $RD \leftarrow true$;
10:         if $RD$ then
11:            $T \leftarrow Update(T)$;
12:         end if
13:      else
14:         $RD \leftarrow false$;
15:      end if
16:   end while
17: end if
18: return $T$;
19: end
```

$B.c$ will be recognized as a probably redundant candidate. In addition, $E.f \xrightarrow{\delta,\lambda} B.c$ might be deduced by $A.a \xrightarrow{\delta,\lambda} B.c$, $E.f \xrightarrow{\delta,\lambda} C.d$ and $A.a \xrightarrow{\delta,\lambda} C.d$, therefore, it will be recognized as a probably redundant candidate and be temporarily removed. After eliminating redundant candidates, there are only 3 valuable candidates left. So, we need add another two tasks (i.e. $C.e \xrightarrow{\delta,\lambda} D.g$ and $B.b \xrightarrow{\delta,\lambda} D.g$) into the task and check the redundant tasks again. If $A.a \xrightarrow{\delta,\lambda} B.c$ is verified to be a true foreign key by crowd, $A.a \xrightarrow{\delta,\lambda} B.b$ will be labeled as false directly, otherwise, it will be added into subsequent task to be verified by crowd. Likewise, if $A.a \xrightarrow{\delta,\lambda} B.c$, $E.f \xrightarrow{\delta,\lambda} C.d$ and $A.a \xrightarrow{\delta,\lambda} C.d$ are verified to be true foreign keys, $E.f \xrightarrow{\delta,\lambda} B.c$ will be deduced as a true foreign key, otherwise, it will be added into subsequent task group.

5 TASK REDUCTION WITH SAMPLING

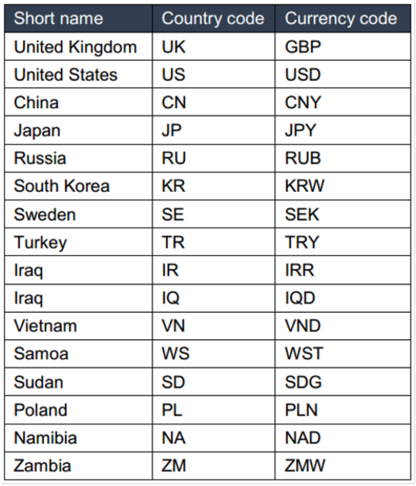
Browsing a large volume table will surely make workers impatient and lead a high cost. To control the tasks' latency, we propose a sampling strategy to sample some representative tuples to prompt to workers.

Generally, the sampling method can be divided into a uniform sampling and a biased sampling. A foreign key relationship involves two tables. Although records in single table are independent, there exists dependency among the attribute pair in a foreign key relationship. Hence, uniform sampling could not be used simply in this situation. We propose a combinational sampling method which can keep the original relationship between tables while reducing the tables' volume.

Our task reduction method based on sampling strategy consists of two phases: dependent table reduction and referenced table reduction. Suppose there is a candidate foreign key relationship $R.a \xrightarrow{\delta,\lambda} S.b$ between table R and S. We sample the dependent table R with the uniform sampling and get the reduced table R'. Based on R', we sample the referenced table S with a biased sampling method. We describe the details as follows.

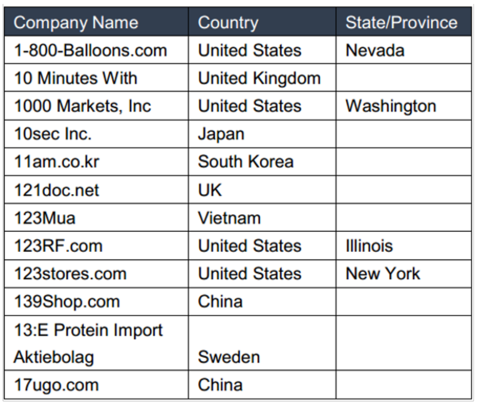
1. Dependent table reduction: Suppose there are m tuples in the dependent table R, the sample rate is ϕ. Considering that each tuple in the table has the same probability to be sampled though some of them may have the same value on dependent attribute, we randomly sample $\lfloor m \times \phi \rfloor$ tuples from R to make up the reduced dependent table R'.
2. Referenced table reduction: Suppose there are n tuples in the referenced table S, and the uniqueness degree of the referenced attribute $S.b$ is δ', only $\lfloor n \times \phi \rfloor$ rows are allowed to be sampled. If we randomly sample the referenced table, the original relationship between two tables (the dependent table and the referenced table) could not be hold. We partition S into covered part and uncovered part. The covered part consists of tuples in which the referenced attribute is referenced by the values sampled in the dependent attribute. The uncovered part consists of tuples left when removing the covered part from S. From the covered part, we extract all the tuples in which values of the referenced attribute is referenced by values of the dependent attribute in R' (reduced dependent table). Suppose the number of tuples sampled from the covered part is k, we then randomly select $\lfloor m \times \phi \rfloor - k$ tuples from the uncovered part, and combine all the tuples we sampled into the reduced referenced table S'.

Example 3. Consider two tables in Figure 5, in which Figure 5 a) is the referenced table S with 16 tuples, and Figure 5 b) is the dependent table R with 12 tuples. There exists a foreign key relationship $R.a \xrightarrow{\delta,\lambda} S.b$ on which the uniqueness degree on $S.b$ is 0.94, and $S.b$'s coverage to $R.a$ is 0.92. If the sampling rate is 0.5, R should be reduced to 6 tuples while S should be reduced to 8 tuples. According to our sampling method, we randomly sample 6 tuples from R and make up the reduced dependent table R' (see Figure 5 d)). After getting R', we should make corresponding reduction to the referenced table. We first partition S into covered part and uncovered part, then extract 4 tuples in which values of the referenced attribute short name is referenced by values of the dependent attribute country in R' (the value of short name is United States, United Kingdom, Japan and Vietnam) from the covered



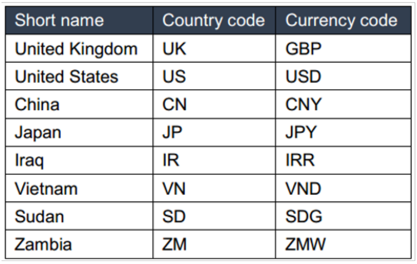
Short name	Country code	Currency code
United Kingdom	UK	GBP
United States	US	USD
China	CN	CNY
Japan	JP	JPY
Russia	RU	RUB
South Korea	KR	KRW
Sweden	SE	SEK
Turkey	TR	TRY
Iraq	IR	IRR
Iraq	IQ	IQD
Vietnam	VN	VND
Samoa	WS	WST
Sudan	SD	SDG
Poland	PL	PLN
Namibia	NA	NAD
Zambia	ZM	ZMW

a)



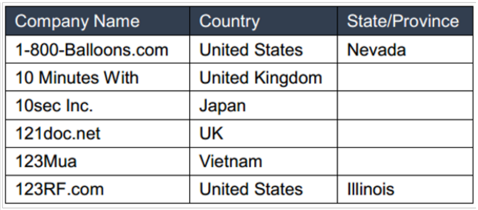
Company Name	Country	State/Province
1-800-Balloons.com	United States	Nevada
10 Minutes With	United Kingdom	
1000 Markets, Inc	United States	Washington
10sec Inc.	Japan	
11am.co.kr	South Korea	
121doc.net	UK	
123Mua	Vietnam	
123RF.com	United States	Illinois
123stores.com	United States	New York
139Shop.com	China	
13:E Protein Import Aktiebolag	Sweden	
17ugo.com	China	

b)



Short name	Country code	Currency code
United Kingdom	UK	GBP
United States	US	USD
China	CN	CNY
Japan	JP	JPY
Iraq	IR	IRR
Vietnam	VN	VND
Sudan	SD	SDG
Zambia	ZM	ZMW

c)



Company Name	Country	State/Province
1-800-Balloons.com	United States	Nevada
10 Minutes With	United Kingdom	
10sec Inc.	Japan	
121doc.net	UK	
123Mua	Vietnam	
123RF.com	United States	Illinois

d)

Figure 5. An example of task reduction by sampling

part, and randomly select 4 tuples from the uncovered part. The reduced referenced table S' is shown in Figure 5 c).

From this example, we can see that only small fraction of tuples in web tables can ensure the quality of crowdsourcing task. In Section 6, we will make suggestion for optimal sampling rate by comparing the performance of different sampled tasks.

6 EXPERIMENT

We evaluate our method using a number of real word web tables. The goals of our experiment are:

1. compare the performance of our candidate foreign key generation algorithm with the fast foreign key detection method proposed in [2],

2. evaluate the power of the crowd,
3. evaluate the effectiveness of the task selection method,
4. evaluate the effectiveness of the task reduction method with sampling.

Dataset: We crawled more than 1 000 web tables from Google tables [6] and selected 118 tables which have semantic relationships with each other to conduct our experiment. The content of these tables refers to sports, economy, technology, movies and so on. Tuples in these tables add up to 12 717, and the total columns are 699. For there is no declared semantic relationship between these web tables, we manually labeled 550 foreign key constraints as the "ground truth" with the help of machine algorithm.

6.1 Performance Comparison of Machine Algorithms

Before crowdsourcing, we preprocess the web tables using machine algorithms to find foreign key candidates with high probability. In this section, we compare the precision, recall and F-Measure of our candidate foreign key detection method denoted as WFD with the fast foreign key detection method denoted as FFD [2].

We run two algorithms to process the web tables, respectively and compare their performance. Under different thresholds of candidate confidence (varying from 0.5 to 1), we compare the precision, recall and the F-measure of FFD and WFD. Precision and recall are calculated by Equations (5) and (6), respectively.

$$Precision = \frac{|TAF|}{|AF|}, \tag{5}$$

$$Recall = \frac{|TAF|}{|WF|}. \tag{6}$$

Where TAF is the set of true foreign keys the machine-based algorithm discovered, AF denotes the foreign key candidates the machine-based algorithm detected, and WF is the true foreign keys in the dataset. *F-measure* is defined as the harmonic mean of precision and recall in following formula:

$$F = \frac{(1+\alpha) \times Precision \times Recall}{\alpha \times Precision + Recall} \tag{7}$$

where α is set to 1 in the experiment.

Table 1 shows the number of candidate foreign keys (FFD disc, WFD disc), true foreign keys they discovered (FFD true, WFD true). WFD finds more true foreign keys than FFD. The higher the threshold is, the less candidates they will find.

Figures 6, 7, 8 describe the precision, recall, and F-measures of the two algorithms, respectively.

When confidence threshold varies from 0.5 to 1, all precision values of WFD are higher than that of FFD. When the threshold is set to 0.9, the precision of WFD

θ	FFD disc	FFD true	WFD disc	WFD true
0.50	772	229	819	448
0.55	598	217	756	437
0.60	487	212	703	428
0.65	435	201	678	420
0.70	320	165	593	409
0.75	253	154	406	310
0.80	205	128	263	207
0.85	133	93	198	156
0.90	94	67	169	134
0.95	45	33	102	79
1.00	17	12	22	17

Table 1. Experiment result of FFD and WFD

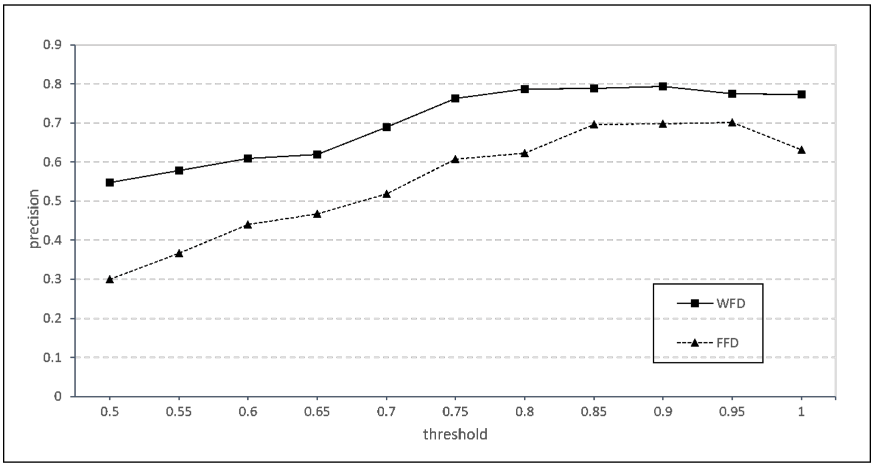


Figure 6. Precision of FFD and WFD

reaches to 79.29 %. The highest precision of FFD is 70.21 % when the threshold is 0.95. For a foreign key candidate $R.a \xrightarrow{\delta,\lambda} S.b$, FFD measures its confidence by four factors including similarity between table name of $R.a$ and $S.b$. However, most of the web tables' names could not describe the semantics of tables exactly. Eliminating the influence of web table's name makes WFD performs better than FFD in precision. The higher the confidence is, the higher the quality of foreign key candidates. Therefore, the precision of WFD and FFD trends to increase generally as confidence threshold increases.

With the increase of confidence threshold, the recall of the two methods decreases. When the threshold is less than 0.7, WFD's recall is much higher than

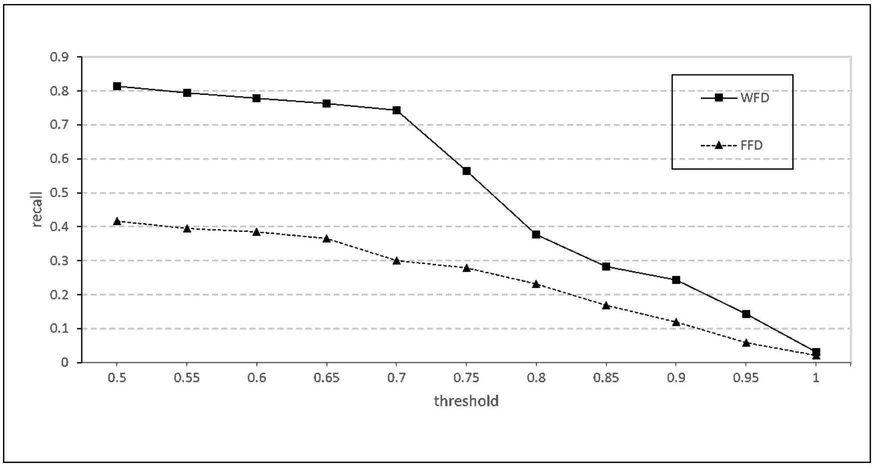


Figure 7. Recalls of FFD and WFD

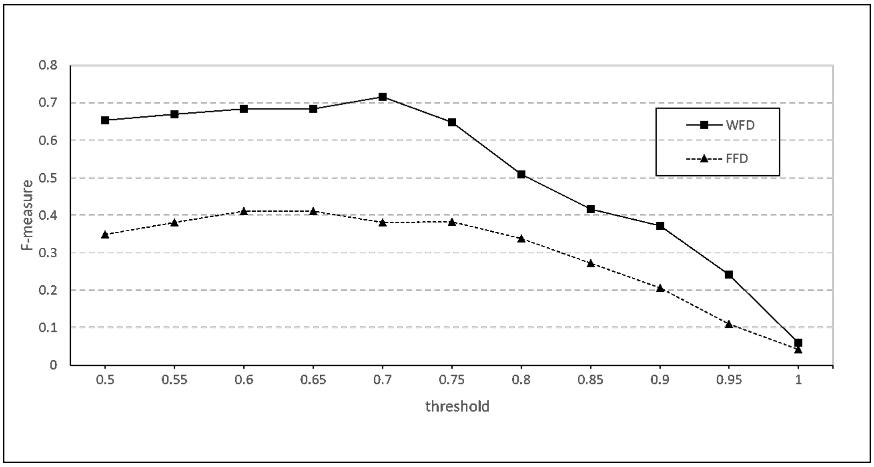


Figure 8. F-measures of FFD and WFD

that of FFD. Relatively, the FFD's recall decreases slowly with the increase of the threshold, while the WFD's recall has a sharp decline with threshold = 0.75. WFD relaxes requirements for uniqueness of key and strict inclusion dependency, therefore its recall is higher than FFD.

Generally, the precision and recall are mutually restricted, and F-measure is a combination of these two indicators which can inflect the overall performance of the method. From Figure 8, F-measure of WFD is higher than that of FFD under various threshold values. WFD and FFD with lower confidence thresholds (less than 0.7 and 0.65, respectively) could not achieve full effectiveness in detecting foreign key candidates. Obviously, it is more possible to contain false positives in candidates with low thresholds. As the threshold grows, the F-measure of WFD and FFD reaches highest (0.72 for WFD and 0.41 for FFD). After that, the F-measure value decreases because recall decreases along with high confidence threshold.

In summary, our machine algorithm performs better than the state-of-the-art algorithm for discovering foreign key candidates on web tables.

6.2 Evaluation of Power of the Crowd

Analyzing the result made by WFD, we find that the false positive candidates are generated mainly because of lack of tables' semantics. With the help of the crowd, those false positives could be easily distinguished. The crowdsourcing experiment is implemented on CrowdSR [12], which is a crowdsourcing platform for semantic recovering of web tables.

From Figure 8, we can see that the F-measure of WFD reaches to the maximum value when the confidence threshold is set to 0.7. So, we set the threshold to 0.7 in the following experiment which means that 593 foreign key candidates including 409 true foreign keys and 184 false positive candidates need to be verified by the crowd. We create corresponding microtasks and post them to the crowd. These tasks are organized to 36 groups, each of which includes the true foreign keys and false positive candidates. Each task is finished by at least three workers with professional knowledge, and the majority voting are used to aggregate answers. Before doing tasks, workers should pass a qualification test which consists of three simple foreign keys verification tasks.

As a result, 376 candidates are verified as true foreign keys, among which 371 true foreign keys are correctly verified, and 5 false positive candidates are verified as true foreign keys by mistake. The performance comparison of machine algorithm and hybrid method is shown in Figure 9.

Obviously, the precision and F-measure of the human-machine hybrid method is better than the machine algorithm. The precision of the hybrid method is improved to $371/376 = 98.67\,\%$ from $68.97\,\%$ with the help of the crowd, since most candidates verified as true foreign keys by crowd are proved to be true. The recall of the human-machine hybrid approach is lower ($67.45\,\%$ vs. $76.36\,\%$) than that of the machine algorithm, because all foreign keys discovered by the human-machine hybrid method are contained in the output of machine algorithm.

6.3 Evaluation of Effectiveness of Dynamical Task Selection Method

This experiment is conducted on the same candidate set mentioned in Section 6.2, which contains 593 foreign key candidates with 409 true foreign keys and 184 false positive candidates. We use the combined task selection method to select the most valuable candidates to be verified and compare candidate numbers and precision with the naive task assignment method (i.e. assign all candidates to workers). Table 2 shows the result.

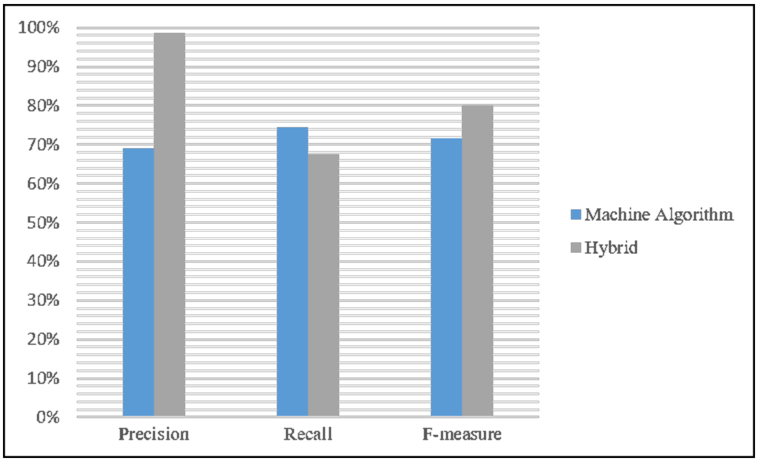


Figure 9. Performance comparison

Method	Crowdsourced Candidate Numbers	Foreign Keys Output	True Foreign Keys	Precision
Task Selection	322	384	368	95.83 %
Naive	593	376	371	98.67 %

Table 2. Naive task assignment method vs. task selection method

By the task selection method, total number of crowdsourced candidates is reduced to 322, about 54.30 % of the number of naive method. The task selection method outputs more foreign keys (384 vs. 376) because some candidates that are mislabeled as false positives in the naive method are deduced as true. There is no significant difference in the number of true foreign keys output by two methods (368 vs. 371). Conflict detection could help to remove candidates conflicted with the true foreign key, and inclusion dependency detection could help to deduce foreign key relationship. However, if candidates are mislabeled as true by the crowd, those candidates that are in conflict with them may be wrongly recognized as false positives. So, the precision of the task selection method is slightly lower (95.83 % vs. 98.67 %) than that of the naive method

To summarize these experimental results, our task selection method can effectively reduce the number of tasks thus reducing the monetary cost. Although some human errors may be amplified, foreign key detection precision under the task selection still reaches 95.83 %.

6.4 Evaluation of Effectiveness of the Task Reduction Method

Finally, we evaluate the task reduction method based on sampling strategy. Our main goal is to compare the time cost of the tasks reduced by our method with the naive tasks without sampling. In order to evaluate the effectiveness of the task reduction method, we vary the sampling rate from 0.1 to 0.5, and compare the finish time and precision.

We sample the dependent table randomly, then partition the referenced table into two parts, the covered part and the uncovered part. From the covered part, we extract all tuples related to the sampled dependent table. Figure 10 shows the average time cost for each task with different sampling rate. Obviously, the average latency is substantially proportional to the sample rate. The time cost of our task reduction method under sampling rate from 0.1 to 0.5 is always lower than the average time cost 54 s of naive tasks.

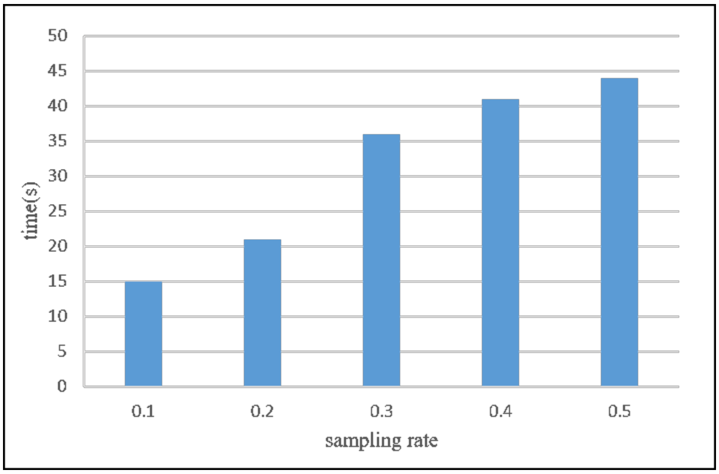


Figure 10. The average time cost under different sampling rate

Though our task reduction method keeps as much original table features as possible by using combinational sampling strategy, it is more easily for the crowd to make erroneous judgment when the table has been badly compressed with low sampling rate. Figure 11 shows the precision of the crowd verification result in different sampling rate. The precision increases with the increase of sampling rate. When the rate is set to 0.5, 364 candidates are verified as foreign keys by the crowd among which 353 are true foreign keys and the precision is close to the precision of the naive method. All the precisions of labeling results under sampling rate from 0.1 to 0.5 are more than 92 %.

In summary, our task reduction method with sampling strategy has a better performance than the naive method. Setting the sampling rate to 0.4, the average time cost is reduced to 41 s, while the precision of the verification result is very close to the precision of the naive method.

Quality vs. Cost. As both the precision and time cost are positively related to the sampling rate, there is a tradeoff between quality and cost. It is very im-

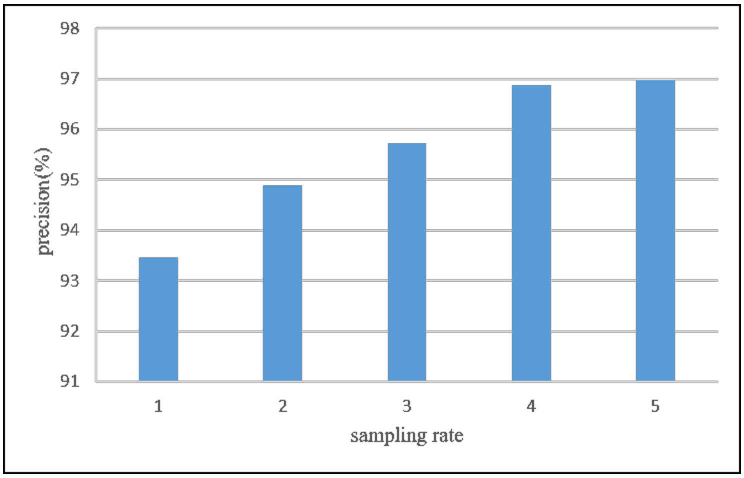


Figure 11. Precision comparison under different sampling rate

portant to select an appropriate sampling rate when using this task reduction method.

7 RELATED WORK

Recently, structured data from the web, such as the web tables, has been identified to have a high value in research. Thus, many researchers tend to recover [13, 14, 15] or integrate [16, 17] data in web tables.

Foreign key detection is an important work for analyzing and integrating data in web tables. Previous researches mainly focus on identifying inclusion dependencies. Bauckmann et al. proposed an algorithm named SPIDER for detecting unary inclusion dependencies [4], while some works [1, 2, 3] present a global way for foreign key detection in relational database. Rostin et al. detect putative foreign keys with a learning based method and propose some meaningful features for classifying inclusion dependencies [1], including DistinctDependentValues, Coverage, ColumeName, and DependentAndReferenced. Chen et al. propose a fast foreign-key detection method in PowerPivot [2] to perform this detection interactively and with high precision even when data sets scale to hundreds of millions of rows and the schema contains tens of tables and hundreds of columns. Randomness is proposed and an algorithm is developed to discover single-column and multi-column foreign keys in [3]. However, all the previous methods are not effective for discovering foreign keys on web tables of poor quality, which could not satisfy the entity integrity constraint and referential integrity constraint.

Fortunately, these problems could be solved easily with human's intelligence. Crowdsourcing is a good way to solve problems that are difficult for computers, and it is widely used in academia such as entity resolution [7], sentiment analysis [8], and image recognition [9]. In this paper, we propose a hybrid human-machine framework to detect foreign keys on web tables. After discovering candidates and evaluating their confidence of being true foreign keys by machine algorithm, we verify those false positives leveraging the power of crowd.

8 CONCLUSIONS

Previous researches on foreign key detection mainly focus on finding the foreign key relationships between the relational tables in database. We are the first to propose a hybrid human-machine framework for discovering foreign keys on web tables which may not satisfy the entity integrity constraint and referential integrity constraint. To reduce the monetary cost, we proposed a dynamical task selection method based on conflict detection and inclusion dependency. Besides, to make workers complete tasks more effectively, sampling strategy is applied to reduce the task volume. The experimental results show our hybrid human-machine approach could indeed achieve a much higher detection precision with lower monetary cost and time cost.

Acknowledgment

This work is supported by the National Key R&D Program of China (2018YFC0809-800), and the National Natural Science Foundation of China (61370060, 61673048).

REFERENCES

[1] Rostin, A.—Albrecht, O.—Bauckmann, J.—Naumann, F.—Leser, U.: A Machine Learning Approach to Foreign Key Discovery. 12th International Workshop on the Web and Databases (WebDB 2009), Providence, Rhode Island, USA, June 2009.

[2] Chen, Z.—Narasayya, V.—Chaudhuri, S.: Fast Foreign-Key Detection in Microsoft SQL Server PowerPivot for Excel. Proceedings of the VLDB Endowment, Vol. 7, 2014, No. 13, pp. 1417–1428, doi: 10.14778/2733004.2733014.

[3] Zhang, M.—Hadjieleftheriou, M.—Ooi, B. C.—Procopiuc, C. M.—Srivastava, D.: On Multi-Column Foreign Key Discovery. Proceedings of the VLDB Endowment, Vol. 3, 2010, No. 1-2, pp. 805–814, doi: 10.14778/1920841.1920944.

[4] Bauckmann, J.—Leser, U.—Naumann, F.—Tietz, V.: Efficiently Detecting Inclusion Dependencies. 2007 IEEE 23rd International Conference on Data Engineering, 2007, pp. 1448–1450, doi: 10.1109/icde.2007.369032.

[5] Cafarella, M. J.—Halevy, A.—Wang, D. Z.—Wu, E.—Zhang, Y.: WebTables: Exploring the Power of Tables on the Web. Proceedings of the VLDB Endowment, Vol. 1, 2008, No. 1, pp. 538–549, doi: 10.14778/1453856.1453916.

[6] Google Fusion Table. Availaible at: `https://research.google.com/tables`.

[7] Wang, J.—Kraska, T.—Franklin, M. J.—Feng, J.: CrowdER: Crowdsourcing Entity Resolution. Proceedings of the VLDB Endowment, Vol. 5, 2012, No. 11, pp. 1483–1494, doi: 10.14778/2350229.2350263.

[8] Zheng, Y.—Wang, J.—Li, G.—Cheng, R.—Feng, J.: QASCA: A Quality-Aware Task Assignment System for Crowdsourcing Applications. Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15), 2015, pp. 1031–1046, doi: 10.1145/2723372.2749430.

[9] Welinder, P.—Perona, P.: Online Crowdsourcing: Rating Annotators and Obtaining Cost-Effective Labels. 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition – Workshops (ACVHL), 2010, pp. 25–32, doi: 10.1109/cvprw.2010.5543189.

[10] Li, G.—Wang, J.—Zheng, Y.—Franklin, M. J.: Crowdsourced Data Management: A Survey. IEEE Transactions on Knowledge and Data Engineering, Vol. 28, 2016, No. 9, pp. 2296–2319, doi: 10.1109/TKDE.2016.2535242.

[11] Winkler, W. E.: String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. Proceedings of the Section on Survey Research Methods, 1990, pp. 354-359.

[12] Liu, H.—Wang, N.—Ren, X.: CrowdSR: A Crowd Enabled System for Semantic Recovering of Web Tables. In: Dong, X., Yu, X., Li, J., Sun, Y. (Eds.): Web-Age Information Management (WAIM 2015). Springer, Cham, Lecture Notes in Computer Science, Vol. 9098, 2015, pp. 581–583, doi: 10.1007/978-3-319-21042-1_67.

[13] Venetis, P.—Halevy, A.—Madhavan, J.—Paşca, M.—Shen, W.—Wu, F.—Miao, G.—Wu, C.: Recovering Semantics of Tables on the Web. Proceedings of the VLDB Endowment, Vol. 4, 2011, No. 9, pp. 528–538, doi: 10.14778/2002938.2002939.

[14] Wang, J.—Wang, H.—Wang, Z.—Zhu, K. Q.: Understanding Tables on the Web. In: Atzeni, P., Cheung, D., Ram, S. (Eds.): Conceptual Modeling (ER 2012). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 7532, 2012, pp. 141–155, doi: 10.1007/978-3-642-34002-4_11.

[15] Deng, D.—Jiang, Y.—Li, G.—Li, J.—Yu, C.: Scalable Column Concept Determination for Web Tables Using Large Knowledge Bases. Proceedings of the VLDB Endowment, Vol. 6, 2013, No. 13, pp. 1606–1617, doi: 10.14778/2536258.2536271.

[16] Yoshida, M.—Torisawa, K.—Tsujii, J.: A Method to Integrate Tables of the World Wide Web. Proceedings of the International Workshop on Web Document Analysis (WDA 2001), 2001, pp. 31–34.

[17] Gonzalez, H.—Halevy, A. Y.—Jensen, C. S.—Langen, A.—Madhavan, J.—Shapley, R.—Shen, W.—Goldberg-Kidon, J.: Google Fusion Tables: Web-Centered Data Management and Collaboration. Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data (SIGMOD '10), 2010, pp. 1061–1066, doi: 10.1145/1807167.1807286.

Xiaoyu Wu is a Ph.D. candidate in computer science in Beijing Jiaotong University. Her main research areas include web data integration, big data management and data mining.

Ning Wang received her Ph.D. degree in computer science in 1998 from Southeast University in Nanjing, China. She is currently serving as Professor in School of Computer and Information Technology, Beijing Jiaotong University, China. Her research interests include web data integration, big data management, data quality and crowdsourcing.

Huaxi Liu received his Master degree in computer science from Beijing Jiaotong University in 2016 and currently works in Huawei Technologies Co., Ltd. His research interests include web data integration, data mining and crowdsourcing.

Computing and Informatics, Vol. 38, 2019, 647–678, doi: 10.31577/cai_2019_3_647

IMAGE ENCRYPTION ALGORITHM WITH PLAINTEXT RELATED CHAINING

Ľuboš Ovseník, Ján Turán, Tomáš Huszaník, Jakub Oravec

Department of Electronics and Multimedia Communications
Technical University of Košice
Němcovej 32, 040 01 Košice, Slovakia
e-mail: {lubos.ovsenik, jan.turan, tomas.huszanik, jakub.oravec}@tuke.sk

Ondrej Kováč

Department of Technologies in Electronics
Technical University of Košice
Park Komenského 2, 040 01 Košice, Slovakia
e-mail: ondrej.kovac@tuke.sk

Milan Oravec

Department of Safety and Quality of Production
Technical University of Košice
Letná 9, 040 01 Košice, Slovakia
e-mail: milan.oravec@tuke.sk

Abstract. This paper describes a plaintext related image encryption algorithm that utilizes the Mojette transform for computation of bins that are subsequently combined with pixels of the processed image. While the bins are computed solely from pixel intensities of a plain image and also the combination depends only on intensities of plain image pixels, the parameters of bins are rearranged according to used key. This design results in a great sensitivity of the proposed image encryption algorithm to both plain images and keys, which is verified by a set of experiments. The paper also tests the resistance of the proposal against statistical and differential attacks by means of commonly used measures as correlation coefficients, entropy, NPCR and UACI. Furthermore, the paper analyses computation speed reached by the proposed solution. Computed values of all parameters are discussed and then compared with results obtained by some recent plaintext related image encryption algorithms.

Keywords: Image encryption, logistic map, Mojette transform, pixel intensity chaining, plaintext related operation

Mathematics Subject Classification 2010: 94A60, 68U10

1 INTRODUCTION

Security of data transmitted over public networks is an important task these days. One of the possible solutions for establishing security of data is encryption. Conventional encryption algorithms such as Advanced Encryption Standard (AES) were usually designed in a way that is suitable for encryption of rather small blocks of hexadecimal or binary characters [1, 2]. However, this approach is not suitable for all data types. Digital images known for their redundancy (vast amount of pixels) and high correlation between adjacent pixels can be considered as an example.

Encryption of image data by conventional encryption algorithms can lead to various undesirable results. For example, a basic mode of operation for AES, called Electronic CodeBook (ECB), only replaces blocks of plain image data by calculated blocks of encrypted data. In the case that some plain image blocks are the same, all of them are substituted with identical blocks of encrypted data. This situation is shown in Figure 1 where AES in ECB mode was used for encryption of the image with resolution of 256×256 pixels and color depth of 8 bits per pixel. Used encryption key was `0× C4 EB 50 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5`. This key was acquired from the first 128 bits of binary representation of decimal part of π.

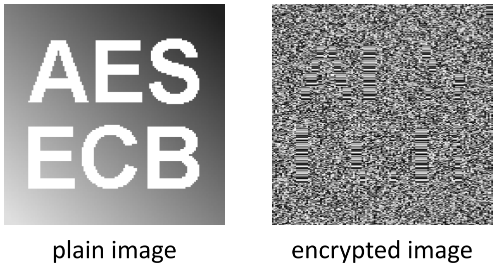


Figure 1. Contours in encrypted image after encryption by ECB mode of AES

As seen in Figure 1, the smooth image areas located in the characters of the plain image, where the intensities of adjacent pixels do not change, produce several identical blocks in the encrypted image. On the other hand, the background of image, which contains a grayscale gradient filling, was substituted with many different blocks. The contrast between these blocks can lead to identification of contours of the characters from the plain image, and that is undesirable.

The example illustrated in Figure 1 can be viewed as an exaggeration as the used plain image is an uncompressed bitmap image. However, also images processed by lossy compression algorithms can contain some identical blocks of pixels. Therefore this problem can arise also in the encrypted versions of such images.

Furthermore, the encryption of images by conventional encryption algorithms can lead to bad performance in terms of computational speed. In the case that used platform does not enable hardware acceleration of encryption [3], the dedicated image encryption algorithms are usually faster [4]. This is due to their better optimization for the task of image encryption. Considering the mentioned properties of images and conventional encryption algorithms, images as a data type require specific and more efficient approach in order to achieve an acceptable performance.

The first dedicated image encryption algorithms were designed in late 1990s. Probably one of the most popular early image encryption algorithms is the Fridrich's algorithm published in 1998 [5], which used chaotic Bakers' map for achieving the desired performance. Application of a chaotic map inspired other researchers [6, 7, 8, 9] and nowadays the majority of image encryption algorithms is based on a suitable chaotic system. Fridrich's paper was also important because it described a useful architecture of image encryption algorithms, consisting of confusion and diffusion stage. This architecture corresponds with ideas of Shannon [10], and with some modifications it is still used nowadays.

First attempts to break chaotic image encryption algorithms can be traced back to early 2000s. However, majority of these papers tried to break only one specific image encryption algorithm [11, 12]. In 2010, Solak et al. proposed a chosen ciphertext attack based on relations between image pixels during decryption [13]. Solak's attack is not only able to break Fridrich's encryption algorithm, but it can be also used for some other designs with the similar architecture [13]. The performance of Solak's attack was later discussed by Xie et al. in [14], where some minor improvements were suggested.

Presentation of the Solak's attack caused changes in the Fridrich's architecture. Most authors tried to establish relations between the steps of image encryption algorithm and used plain image (image before encryption). Therefore these approaches are described as plaintext related image encryption algorithms.

Despite chaotic image encryption algorithms still evolve, the existing algorithms have already found some applications. They usually require sensitive data to remain in the form of an image: image steganography [15, 16, 17], transfer and storage of biometric features [18] or medical images [19].

Image encryption algorithms can be also used for verification of data integrity of images, because any image processing technique applied on an encrypted image leads to the incorrect decryption. If users wish to enhance or modify some image, it needs to be processed before the encryption. Then the decryption of the encrypted image leads to the processed image.

The rest of the paper is organized as follows: Section 2 describes works of other authors in the field of plaintext related chaotic image encryption. Section 3 explains techniques used in our proposal and describes the steps of encryption and decryption algorithms. Section 4 presents obtained experimental results and compares them with the results achieved by some other similar algorithms. Section 5 discusses the properties of the proposed solution and concludes the paper.

2 RELATED WORK

Since the introduction of Solak's attack [13] in 2010, several solutions for suppressing or even eliminating a possibility of a successful attack have been provided. Fu et al. proposed their solution in 2012 [20] where parameter of Chebyshev's map is modified according to intensity of the previously encrypted pixel. However, these parameter modifications can lead to fixed points, where behavior of chaotic maps is constant and therefore unsuitable. Fu et al. used a mechanism that prevents occurrence of fixed points, however this step decreases the overall computational speed.

A proposal by Kanso et al. from 2012 [21] changed the amount of iterations of Arnold's cat map by intensities of currently processed image pixel. As the pixel intensities are divided by 10 and then rounded to the closest smaller integer number, multiple intensities lead to identical amount of iterations. Also, each additional map iteration causes the longer computational time, so timing attacks [22] are possible.

A paper by Fu et al. [23] from 2013 used cyclic shifts of bits from pixel intensities controlled by intensities of previous image pixels. As there are only 8 possible ways of cyclic shift of 8 bits, this solution can lead to equivalent shifts for multiple intensities (there are 256 possible pixel intensities for color depth of 8 bits per pixel).

In 2014, Zhang proposed an encryption algorithm [24] based on a different architecture, where one iteration of diffusion is followed by plaintext related confusion and then second iteration of diffusion. However, this approach can be prone to chosen plaintext attacks [25] in cases when one iteration of diffusion produces a similar results for two images, and the plaintext related confusion – a rearrangement of image pixels only shuffles these slightly different image pixels. Similar problems can happen also with Zhang's later design from 2015 [26].

Murillo-Escobar et al. proposed an image encryption algorithm in 2015 [27] that utilized a sum of plain image pixel intensities for calculation of an initial condition and parameter of logistic map. This solution has two drawbacks: the same sum can be computed from various images and it is not possible to calculate sum used during encryption from the encrypted image. For enabling successful decryption, Murillo-Escobar et al. suggested that this value can be hidden as intensity of one of encrypted image pixels. This approach was broken by Fan et al. in 2018 [28] by usage of chosen and known plaintext attacks.

Three similar plaintext related image encryption algorithms were designed by Chai et al. in 2017 [29], by Wang et al. [30] and Li et al. [31] in 2018. These algorithms use hash functions for calculation of the digests of plain images. The digests are then used for modifying initial conditions or parameters of chaotic maps. Hash functions should provide significantly different digests for similar plain images, however, their usage usually negatively affects the computational speed of the whole algorithm. This drawback is visible mainly in the case of [30], which uses two hash functions.

We already published one paper describing a plaintext related image encryption algorithm [32]. However, as the relation was established individually between each image pixel and each key element by means of Arnold's cat map, it was necessary to provide a key with the length of used plain image. This problem was solved by a key extension algorithm. As the decryption algorithm requires the last elements of the extended key at start of the decryption, the keys for encryption and decryption are different. The fact that approach [32] is asymmetric can be viewed as a drawback.

3 PROPOSED SOLUTION

The algorithm proposed in this paper can be used for encryption of images with arbitrary resolution higher than 16×16 pixels and color depths of 8 or 24 bits per pixel. The restriction placed on image resolution is caused by usage of multiple different image rows for processing of each row of image pixels in the proposed algorithm. The plaintext related operation utilizing the Mojette transform (MoT) requires a matrix of 12×12 pixel intensities. Together with other 2 rows of pixel intensities used for plaintext unrelated operations, one row that chooses the computed Mojette bins and the actually processed row of pixel intensities, the amount of required rows reaches 16. The amount of necessary columns of image pixels, which is 12 was then enlarged to 16 in order to produce so-called square resolution (16×16 pixels). The mentioned color depths are common for grayscale and true color images.

The proposed image encryption algorithm uses key with length of 128 bits, stored in a hexadecimal notation. As it will be shown in Section 4.1, this key length can be considered as sufficient by means of a brute-force attack. Architecture of the proposed algorithm is inspired by Fridrich's approach, however several stages are added as it is shown in Figure 2.

Steps of the proposed image encryption algorithm were carefully chosen for obtaining acceptable performance by means of commonly used measures (presented in Section 4) and also reasonably fast speed of encryption or decryption (investigated in Section 4.4). The following paragraphs describe individual stages of the proposed image encryption algorithm.

Combinations with generated pseudo-random sequences (PRSs) are used for two purposes. The first combination used during encryption helps to achieve better results for plain images with simple scenes, such as "black" images where intensities of all pixels are zero. The second purpose is suppression of the possibility of a suc-

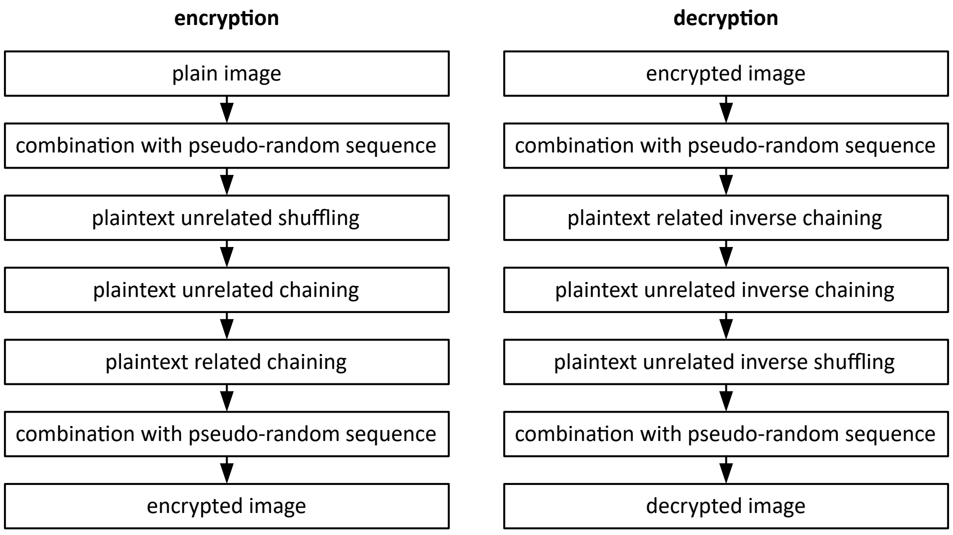

encryption
plain image
combination with pseudo-random sequence
plaintext unrelated shuffling
plaintext unrelated chaining
plaintext related chaining
combination with pseudo-random sequence
encrypted image
decryption
encrypted image
combination with pseudo-random sequence
plaintext related inverse chaining
plaintext unrelated inverse chaining
plaintext unrelated inverse shuffling
combination with pseudo-random sequence
decrypted image

Figure 2. An architecture of the proposed image encryption algorithm

cessful attack. If the plain image is combined with two PRSs before and after the encryption, an attacker would need to guess at least one of the used PRSs before attacking the encryption algorithm. All PRSs used in our encryption algorithm are generated by logistic map (LM) and then processed and quantized.

Confusion stage (shuffling of image pixels) consists of two steps. The first step rearranges image pixels in individual rows of image, while the second step shuffles pixels in individual columns of the image. The rearrangement of image pixels helps to suppress the relations between adjacent image pixels, such as their correlation.

Diffusion stage exploits the fact that the intensities of image pixels were already combined with one PRS. Therefore the diffusion can be achieved simply by performing chaining of pixel intensities. The chaining is done in two steps, the first one is not related to plain image and it is used only for establishing relations between intensities of all image pixels. Second step of the chaining is plaintext related operation based on MoT.

The main contribution of the proposed solution is a description of a novel approach of plaintext related image encryption based on MoT. MoT was chosen because it operates directly with matrices of pixel intensities and therefore it enables relatively simple implementation in the second step of the pixel intensity chaining. Also, MoT has a relatively large amount of parameters even for matrices with small sizes. Finally, as these parameters can be rearranged, the usage of MoT brings a desired nonlinear operation to the proposed image encryption algorithm.

The novel application of relatively simple MoT should lead to a faster performance than that obtained by image encryption algorithms based on more complicated techniques. As it is shown in Section 4.5, the proposed image encryption algorithm is one of the fastest algorithms when compared to similar plaintext related image encryption algorithms. Section 4.5 also discusses properties of the proposed algorithm in contrast with features of other published approaches.

MoT computes bins by using various projections of plain image blocks. Each bin is calculated as a sum of exactly three pixel intensities chosen from various color planes of image (if it has multiple color planes). As image pixels are already rearranged at the point of MoT based chaining, the bin computations use pixel intensities from various locations of the plain image.

Proposed approach utilizes encryption keys also for shuffling parameters of computed Mojette bins. Therefore, the calculated bins depend on both pixel intensities and the key. The rearrangement of Mojette parameters also causes a nonlinearity in calculations, where slight changes in parameters result in big differences in computed bin values. Calculated bins are then combined with rows of the image according to rows that are not yet encrypted, therefore this operation is plaintext related.

3.1 Preliminaries

3.1.1 Logistic Map

Logistic map (LM) is an example of one-dimensional chaotic map controlled by one parameter. LM was popularized mainly by work of May [33]. The equation for calculation of successive iterates generated by LM can be expressed as Equation (1):

$$x_{n+1} = r \cdot x_n(1 - x_n) \tag{1}$$

where $x_{n+1} \in (0, 1)$ is value of a successive iterate, $r \in (0, 4)$ is a parameter of the map and $x_n \in (0, 1)$ is value of a current iterate. Calculations of the first iterate x_1 utilize value x_0 known as an initial condition or initial value.

Chaotic behavior of LM is presented on its bifurcation diagram shown in Figure 3. While the behavior of the map is predictable after the first bifurcation that occurs at $r \sim 3$, after several more bifurcations the predictability gradually decreases. The point where $r \sim 3.56995$ is known as "an onset of chaos" [34] and as a lower bound of parameter r that causes a suitable chaotic behavior of the LM.

LM often uses so-called transient period for providing more complex chaotic behavior of generated sequences. The iterates generated during transient period are used only for modification of initial condition x_0. Usual lengths of transient period are powers of 10, e.g. 1 000 iterates.

3.1.2 Mojette Transform

The Mojette transform (MoT) described in 1995 by Guédon et al. [35] is a discrete two-dimensional transform that sums matrix elements over projection lines [36, 37]. These sums are called bins and projection lines are given by three parameters – b which selects summed elements and p and q which determine discrete projection

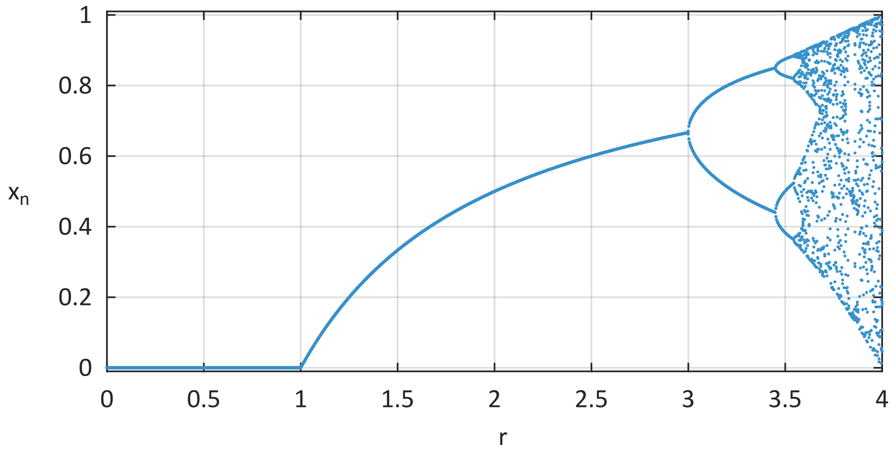


Figure 3. A bifurcation diagram of the logistic map

angle. Mathematically, calculation of Mojette bins for an image is given as Equation (2):

$$bin(b,p,q) = \sum_{k=1}^{w} \sum_{l=1}^{h} Im(l,k) \cdot \delta(b + kq - pl) \tag{2}$$

where $k = 1, 2, \ldots, w$ is column index of image Im, w is width of the image, $l = 1, 2, \ldots, h$ is its row index, h is height of the image and $\delta(x)$ is a Kronecker delta function, $\delta(x) = 1$ if $x = 0$, $\delta(x) = 0$ otherwise.

The calculation of Mojette bins for a simple image and a set of two projections is illustrated in Figure 4. Please note that additions use modulo 256 operation in order to preserve the interval of inputs (given by 8 bits – a set $\{0, 1, \ldots, 255\}$) [38].

MoT has various applications including image coding [39, 40]. In our previous work, we found out that MoT can be used for establishing relations between image pixel intensities [41]. This feature is also the goal of diffusion stages of the image encryption algorithms. Furthermore, MoT has a property of redundancy, which causes multiple usages of pixel intensities during bin computations. While this can be viewed as a drawback from the point of computational speed, we utilize it to suppress possibility of differential attacks as each different pixel intensity would affect several bins.

3.2 Encryption Algorithm

The encryption algorithm takes a plain image P and a key K with length of 16 hexadecimal characters as inputs. Resolution of P needs to be at least 16×16 pixels for enabling computations of sufficient number of Mojette bins. The only output is encrypted image E.

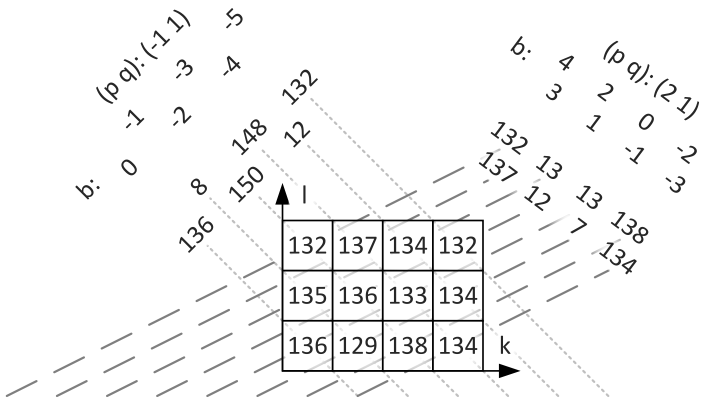


Figure 4. An example of calculated Mojette bins

3.2.1 Generation and Processing of the Pseudo-Random Sequences

Inputs: plain image P, key K.

Output: six processed and quantized PRSs seq'_{1m}, seq'_2 to seq'_5 and seq'_{6m}.

Step 1: Height h, width w and number of color planes num_{cp} of plain image P are determined. These values are used for computation of extended width $w_{ext} = w \cdot num_{cp}$ and total number of pixels $num_{px} = w \cdot h \cdot num_{cp}$.

Step 2: Key K is divided to four parts: K_1 uses first four bytes of K, K_2 utilizes bytes 5 to 8, K_3 is made of bytes 9 to 12 and K_4 uses the last four bytes of K. All four key parts K_1 to K_4 are then converted from hexadecimal to decimal notation.

Step 3: Four key parts K_1 to K_4 are utilized for calculation of four LM (1) parameters r_1 to r_4:

$$r_i = 3.9999 + 25 \cdot 10^{-6} \cdot (i - 1 + 2^{-32} \cdot K_i) \tag{3}$$

where $i = 1, 2, 3, 4$ is an index of parameter and key part, coefficient of $(i-1) \cdot 25 \cdot 10^{-6}$ ensures that each r_i stays in a different interval and constant of 2^{-32} fixes the interval of all possible K_i to $[0, 1)$.

Step 4: Six PRSs seq_1 to seq_6 are generated by six LMs (1). The initial condition x_0 is equal to 0.5 in all cases, values of parameter r are changed during the transient period according to Table 1. The changing of parameter helps to establish relations between all generated sequences and all key parts. Lengths of generated sequences are included in the bottom row of Table 1.

Step 5: The PRSs seq_1 to seq_6 are quantized by applying Equation (4). Quantized sequences are denoted as seq'_1 to seq'_6. Maximal possible value of element after

	Sequence	seq_1	seq_2	seq_3	seq_4	seq_5	seq_6
r used	iterates 1 to 250	r_1	r_2	r_3	r_4	r_2	r_3
during	iterates 251 to 500	r_2	r_3	r_4	r_1	r_1	r_2
transient	iterates 501 to 750	r_3	r_4	r_1	r_2	r_4	r_1
period	iterates 751 to 1 000	r_4	r_1	r_2	r_3	r_3	r_4
r used for following iterates		r_1	r_2	r_3	r_4	r_2	r_3
length of sequence [iterates]		num_{px}	w_{ext}	h	16	16	num_{px}

Table 1. Parameters of logistic maps and lengths of generated sequences

quantization is individual for each sequence and it is determined by Table 2.

$$seq_i' = \left\lfloor (max_i + 1) \cdot \left(10^5 \cdot seq_i \pmod 1\right)\right\rfloor \tag{4}$$

where $i = 1, 2, \ldots, 6$ is an index of sequence and max_i is the maximal possible quantized value for each sequence.

Sequence	seq_1	seq_2	seq_3	seq_4	seq_5	seq_6
max_i	255	$h - 1$	$w_{ext} - 1$	15	15	255

Table 2. Maximal possible values of the sequence elements after quantization

The multiplication of LM iterates by a constant of 10^5 and the modulo operation help to provide more uniform distribution of values, as it is demonstrated on the example in Figure 5. This example used two sequences, both with length of 10^6 iterates, initial condition $x_0 = 0.5$ and parameter $r = 4 - 10^{-15}$. While the sequence denoted as "before processing" was simply generated by LM (1), the second sequence was processed by the multiplication and the modulo operation.

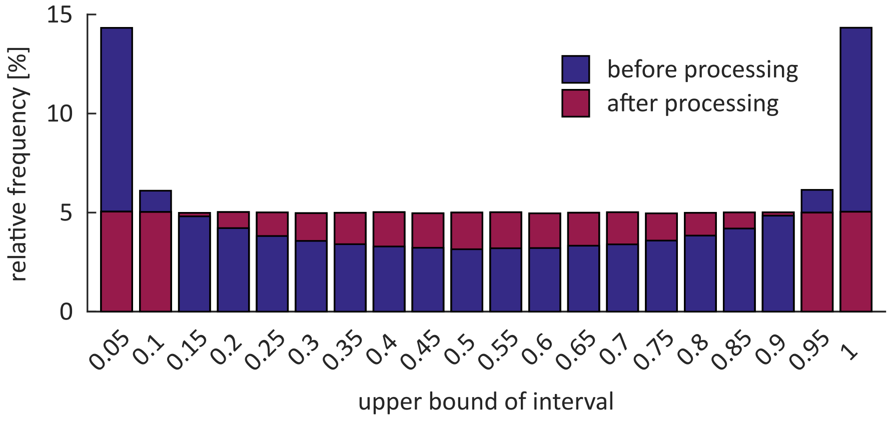


Figure 5. Effect of multiplication and modulo operation on distribution of iterate values

Step 6: Sequences seq'_1 and seq'_6 are rearranged to matrices seq'_{1m} and seq'_{6m} with h rows and w_{ext} columns. The rearrangement uses rows first scanning pattern.

3.2.2 Combination with First Pseudo-Random Sequence

Inputs: plain image P, number of color planes num_{cp}, width w, pseudo-random sequence seq'_{1m}.

Output: extended image P_{ext}.

Step 1: Plain image P is rearranged to extended image P_{ext}. Grayscale images ($num_{cp} = 1$) are simply copied to P_{ext}, while true color images ($num_{cp} = 3$) are decomposed to individual color planes. Columns of red color plane are then copied into columns of P_{ext} with indexes $1 + 3 \cdot (i - 1)$, where $i = 1, 2, \dots, w$. Columns of green and blue color planes are copied into columns of P_{ext} with indexes $2 + 3 \cdot (i - 1)$, and $3 \cdot i$ where $i = 1, 2, \dots, w$, respectively.

Step 2: Sequence seq'_{1m} is combined with extended image P_{ext} by means of bitwise eXclusive OR (XOR) using Equation (5). This step is necessary for images with simple scenes (images where most pixel intensities are similar).

$$P_{ext} = P_{ext} \oplus seq'_{1m} \tag{5}$$

where $\oplus$ is an operator of bitwise XOR.

3.2.3 Confusion Stage

Inputs: extended image P_{ext}, extended width w_{ext}, height h, pseudo-random sequences seq'_2 and seq'_3.

Output: extended image P_{ext} with rearranged image pixels.

Step 1: Pixel intensities in columns of extended image P_{ext} are shuffled by a cyclic shift to the bottom side of P_{ext}. Size of shift is individual for each column of P_{ext} and it is determined by corresponding element of sequence seq'_2 (Equation (6)):

$$P_{ext}(l, k) = P_{ext}\left(1 + (l - 1 + seq'_2(k) \pmod{h}), k\right) \tag{6}$$

where $l = 1, 2, \dots, h$ is row index and $k = 1, 2, \dots, w_{ext}$ is column index.

Step 2: Pixel intensities in rows of extended image P_{ext} are shuffled by a cyclic shift to the right side of P_{ext}. Size of shift is individual for each row of P_{ext} and it is determined by corresponding element of sequence seq'_3 (Equation (7)):

$$P_{ext}(l, k) = P_{ext}\left(l, 1 + (k - 1 + seq'_3(l) \pmod{w_{ext}})\right). \tag{7}$$

3.2.4 Diffusion Stage – Plaintext Unrelated Chaining

Inputs: extended image P_{ext}, height h, extended width w_{ext}.

Output: extended image P_{ext} with chained intensities of image pixels.

Step 1: Rows of extended image P_{ext} are scanned from the top to the bottom. Pixel intensities in individual rows of P_{ext} are chained with pixel intensities in neighboring rows. The chaining is done by element-wise modulo 256 addition of intensities from the previous scanned row and then bitwise XOR with intensities from the next scanned row, as shown in Figure 6. The top row uses the bottom one as the previous scanned row and the bottom row uses the top row as the next scanned row.

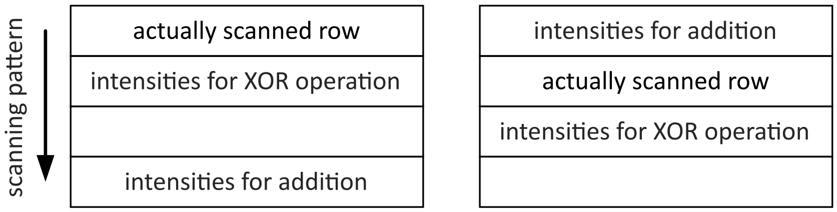


Figure 6. Demonstration of chaining for the first two rows

Step 2: Columns of extended image P_{ext} are scanned from the leftmost one to the rightmost one. Pixel intensities in individual columns of P_{ext} are chained with pixel intensities in neighboring columns. The chaining is done by element-wise modulo 256 addition of intensities from previous scanned column and then bitwise XOR with intensities from next scanned column. The leftmost column uses the rightmost one as the previous scanned column and the rightmost one uses the leftmost one as the next scanned column.

Step 3: Rows of extended image P_{ext} are scanned from the bottom to the top. Pixel intensities in individual rows of P_{ext} are chained by bitwise XOR with intensities from the next scanned row and then element-wise modulo 256 addition of intensities from the previous scanned row. The bottom row uses the top row as the next scanned row and the top row uses the bottom one as the previous scanned row.

Step 4: Columns of extended image P_{ext} are scanned from the rightmost one to the leftmost one. Pixel intensities in individual columns of P_{ext} are chained by bitwise XOR with intensities from next scanned column and element-wise modulo 256 addition of intensities from previous scanned column. The rightmost one uses the leftmost one as next scanned column and the leftmost column uses the rightmost one as previous scanned column.

Four different scans are used for establishing relations between all pixel intensities. An example of this property is shown in Figure 7 where the fill of matrix element means difference between two images. The two operations with neighboring rows or columns are utilized for creating relations during both encryption and decryption.

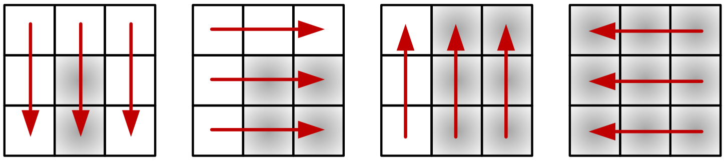


Figure 7. Scanning pattern used during plaintext unrelated chaining

3.2.5 Diffusion Stage – Plaintext Related Chaining

Inputs: extended image P_{ext}, extended width w_{ext}, height h, pseudo-random sequences seq'_4 and seq'_5.

Output: extended image P_{ext} with intensities of image pixels chained according to its pixel intensities.

Step 1: A set of parameters of Mojette bins represented by matrix mot_{par} is rearranged similarly as the extended image P_{ext} during confusion stage. The matrix mot_{par} has 16 rows and 16 columns. Its elements with linear indexes (rows first scanning pattern) are shown in Table 3. Firstly, a cyclic shift shuffles the parameters in columns of mot_{par} according to sequence seq'_4. Then the parameters in rows of mot_{par} are rearranged depending on sequence seq'_5.

Indexes	$(p\ q)$	Set of b
1 to 20	$(-5\ 1)$	$\{-56, -55, -51, -50, -46, -45, -41, -40, \dots, -16, -15, -11, -10\}$
21 to 36	$(-5\ 2)$	$\{-57, -55, -52, -50, -47, -45, -42, -40, \dots, -27, -25, -22, -20\}$
37 to 48	$(-5\ 3)$	$\{-58, -55, -53, -50, -48, -45, -43, -40, -38, -35, -33, -30\}$
49 to 56	$(-5\ 4)$	$\{-59, -55, -54, -50, -49, -45, -44, -40\}$
57 to 96	$(-4\ 1)$	$\{-47, -46 - 45, -44, -43, -42, -41, -40, -39, -38, \dots, -10, -9, -8\}$
97 to 120	$(-4\ 3)$	$\{-53, -50, -49, -47, -46, -45, -44, \dots, -32, -31, -30, -28, -27, -24\}$
121 to 128	$(-4\ 5)$	$\{-59, -55, -54, -50, -49, -45, -44, -40\}$
129 to 168	$(4\ 1)$	$\{-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, \dots, 34, 35, 36\}$
169 to 192	$(4\ 3)$	$\{-9, -6, -5, -3, -2, -1, 0, 1, 2, 3, 4, \dots, 12, 13, 14, 16, 17, 20\}$
193 to 200	$(4\ 5)$	$\{-15, -11, -10, -6, -5, -1, 0, 4\}$
201 to 220	$(5\ 1)$	$\{-1, 0, 4, 5, 9, 10, 14, 15, 19, 20, 24, 25, \dots, 39, 40, 44, 45\}$
221 to 236	$(5\ 2)$	$\{-2, 0, 3, 5, 8, 10, 13, 15, \dots, 28, 30, 33, 35\}$
237 to 248	$(5\ 3)$	$\{-3, 0, 2, 5, 7, 10, 12, 15, 17, 20, 22, 25\}$
249 to 256	$(5\ 4)$	$\{-4, 0, 1, 5, 6, 10, 11, 15\}$

Table 3. A set of parameters utilized for computing Mojette bins

Rearrangement of parameters according to sequences generated by a key establishes the relations between values of computed bins and the key. As the parameters are shuffled, even small change in their indexes should result in a significant difference of computed bin values – this step causes nonlinearity.

The set of parameters was chosen according to several rules – the greatest common denominator of p and q is equal to one and q is always positive. These two conditions were formulated by Guédon et al. [38] and ensure uniqueness of projection angles. Furthermore, as our approach uses MoT for specific purposes, we introduced other conditions:

- $|p| > 3$ ensures that the summed intensities belong to different pixels,
- $|p| \pmod 3 \neq 0$ provides pixel intensities from various color planes,
- values of b were chosen in a way that bins are always made as sums of exactly three intensities.

Step 2: Rows of extended image P_{ext} are scanned from top to bottom. Intensities of an actually scanned row are stored in a vector row_{act}. Intensities of a row that is two rows under the row_{act} are stored in a vector row_{modif}. Then, the following 144 intensities (columns first scanning pattern) are copied to vector vec_{int}. These operations are depicted in Figure 8. If row indexes for row_{modif} and vec_{int} are higher than height h of P_{ext}, the algorithm uses rows from the top of P_{ext}.

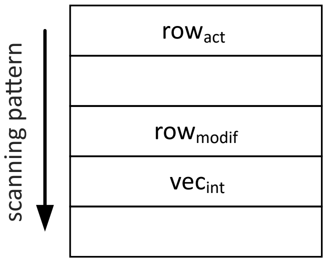


Figure 8. Operations with rows of extended image

Please note that pixel intensities rows of P_{ext} were already shuffled during confusion stage. Therefore the 144 intensities in vec_{int} were chosen from the whole image. Also, the requirement of 144 pixel intensities causes a restriction of minimal image resolution – the size of 16×16 pixels was chosen as the closest square resolution to 16×12 pixels.

The sixteen rows are necessary as one is actually scanned row (row_{act}), another is used for modifications (row_{modif}), two rows are utilized during plaintext unrelated chaining and the remaining 12 rows produce a block of 12×12 pixels required for computation of Mojette bins.

Step 3: Vector vec_{int} is rearranged to matrix mat_{int} with 12×12 elements (rows first scanning pattern). This matrix is used for computation of 256 bins by MoT with parameters from rearranged matrix mot_{par}. Bin values are stored in a vector vec_{bins}. The redundancy property of MoT causes that 256 bins are calculated from various triples of 144 pixel intensities (they are used multiple times).

Step 4: Intensities of row_{act} are element-wise combined with bins vec_{bins} by means of bitwise XOR (Equation (8)). A bin value for each intensity from row_{act} is chosen according to corresponding intensity from row_{modif}. The resulting vector of intensities is stored in the matching row of extended image P_{ext}.

$$P_{ext}(1{:}h, k) = P_{ext}(1{:}h, k) \oplus vec_{bins}\left(row_{modif}(k)\right) \tag{8}$$

where $1{:}h$ denotes sequence $1, 2, \ldots, h$, $k = 1, 2, \ldots, w_{ext}$ is column index and $\oplus$ is an operator of bitwise XOR.

3.2.6 Combination with Second Pseudo-Random Sequence

Inputs: extended image P_{ext}, number of color planes num_{cp}, width w, pseudo-random sequence seq'_{6m}.

Output: encrypted image E.

Step 1: Sequence seq'_{6m} is combined with extended image P_{ext} by means of bitwise XOR using Equation (9). This step suppresses possibility of attacks on the plaintext related diffusion stage.

$$P_{ext} = P_{ext} \oplus seq'_{6m}. \tag{9}$$

Step 2: Extended image P_{ext} is rearranged to encrypted image E. Grayscale images ($num_{cp} = 1$) are simply copied to E, while true color images ($num_{cp} = 3$) are combined from their individual color planes. Columns of red color plane of E are achieved from columns of P_{ext} with indexes $1 + 3 \cdot (i - 1)$, where $i = 1, 2, \ldots, w$. Columns of green and blue color planes of E are obtained from columns of P_{ext} with indexes $2 + 3 \cdot (i - 1)$, and $3 \cdot i$ where $i = 1, 2, \ldots, w$, respectively.

3.3 Decryption Algorithm

Steps of decryption algorithm are analogous to encryption. The opposite order of operations can be seen in Figure 2. One exception of the opposite order is decomposition to two-dimensional extended image E_{ext} and rearrangement to a matrix with one or three color planes. Generation and processing of the PRSs is the same as during encryption. Combinations with PRSs are swapped, the first one is done with sequence seq'_{6m}, while the second one uses seq'_{1m}. Plaintext unrelated inverse chaining utilizes subtraction instead of addition, the usage of all bitwise XOR operations stays the same. Also, the order of subtraction and bitwise XOR operations is reversed. The shifts of image pixels during inverse confusion utilize negative values of elements from sequences seq'_3 and seq'_2.

4 EXPERIMENTAL RESULTS

All experiments described in this section were performed on a PC running MATLAB R2015a on Windows 10 OS, with a 2.5 GHz Intel Core i7-6500U Skylake CPU and 12 GB of RAM. A set of experimental plain images is shown in Figure 9. Their parameters are mentioned in Table 4. Please note that images *black* and *blackG* are magnified in all following figures, as their low resolution enables detection of possible patterns by a naked eye.

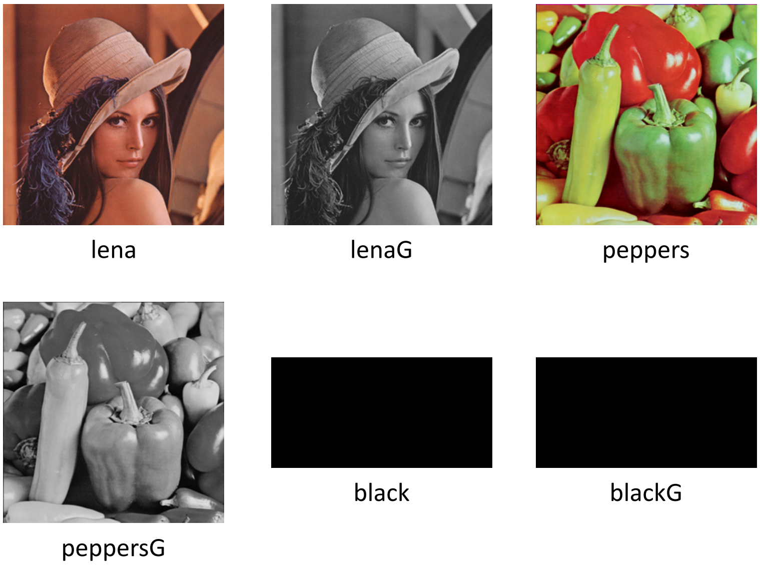


Figure 9. A set of experimental plain images

Image	*lena*	*lenaG*	*peppers*	*peppersG*	*black*	*blackG*
height [px]	512	512	512	512	16	16
width [px]	512	512	512	512	32	32
color depth [bits/px]	24	8	24	8	24	8

Table 4. Parameters of used plain images

A set of three experimental keys in hexadecimal notation is displayed in Table 5. Value of K_1 was obtained from the first 128 bits of binary representation of decimal part of π. The difference between K_1 and K_2 is highlighted by italics.

Key	Value
K_1	0× C4 EB 50 BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5
K_2	0× C4 EB *51* BC 0E C5 EB 50 BC 0E C5 EB 50 BC 0E C5
K_3	0× 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Table 5. A set of experimental keys

Encrypted versions of some plain images from Figure 9 are shown in Figure 10. All encryptions used key K_1.

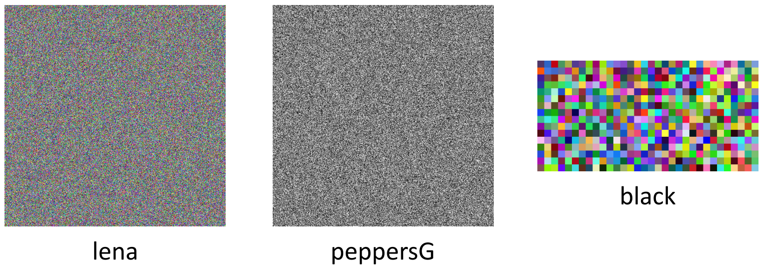


Figure 10. Some examples of encrypted images

4.1 Size of Key Space

The proposed algorithm uses key with length of 16 bytes. Therefore the size of key space can be expressed as $256^{16} = 2^{8 \cdot 16} = 2^{128}$. Considering that one decryption of color image with resolution of 512×512 pixels takes approx. 420 ms (refer to Section 4.4 for details), the brute-force attack would require approx. 2.7192×10^{32} years to complete. Therefore we consider this type of attack as infeasible.

4.2 Key and Plaintext Sensitivity Analysis

4.2.1 Key Sensitivity

Key sensitivity of the proposed algorithm is illustrated in Figure 11. The top row of images was created by encryption of plain image *lena* by three various keys. Left image in the bottom row shows differences between images encrypted by two different keys. The other two images in the bottom row illustrate decryption by the correct key (middle image) and by incorrect key (right image).

4.2.2 Plaintext Sensitivity

Sensitivity of encryption algorithm to slight changes of plaintext (in the form of plain image before encryption or encrypted image before decryption) can be demonstrated by two simple experiments.

The first experiment increases intensity of the last scanned pixel (in the bottom right corner of blue color plane) of image *black*. The pixel intensity is increased by one level from 0 to 1. The left and middle images in Figure 12 illustrate effect of one different pixel intensity before encryption with the same key.

The second experiment starts with encryption of image *black* by key K_1. The result is shown in left image in Figure 12. Then, the encrypted image is modified by increasing the intensity of last scanned image pixel (in the top left corner of red color plane) of encrypted image. The modified image is finally decrypted by key K_1. The decryption without modification should lead to the original plain image *black*, as

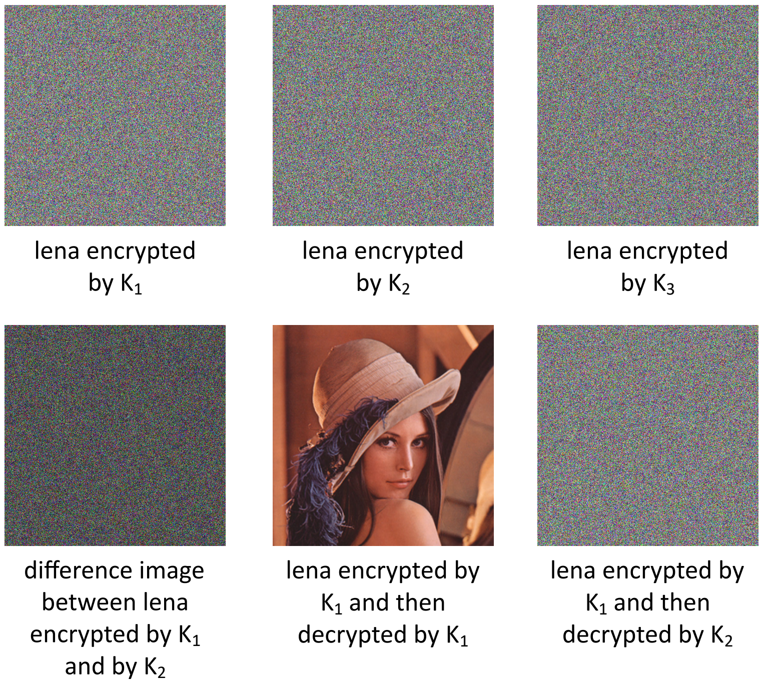

lena encrypted by K_1
lena encrypted by K_2
lena encrypted by K_3
difference image between lena encrypted by K_1 and by K_2
lena encrypted by K_1 and then decrypted by K_1
lena encrypted by K_1 and then decrypted by K_2

Figure 11. Demonstration of key sensitivity

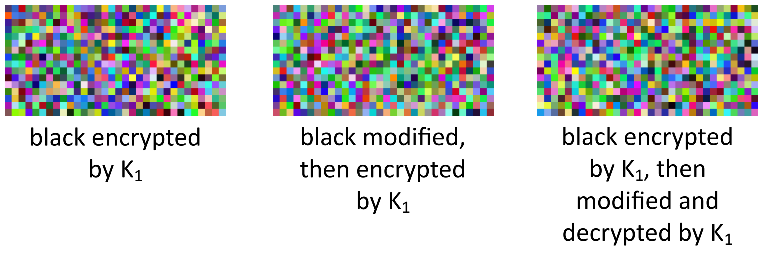

black encrypted by K_1
black modified, then encrypted by K_1
black encrypted by K_1, then modified and decrypted by K_1

Figure 12. Effects of modifications of image before and after encryption

shown in Figure 9. However, as seen in the right image in Figure 12, the decryption of modified image produced a totally different image.

4.3 Robustness Against Certain Types of Attacks

This section investigates the robustness of the proposed image encryption algorithm against commonly used attacks in a field of image encryption. The ways how measures used for assessment of the robustness are computed can be interpreted as examples of certain types of well known attacks.

In the case that the proposed image encryption algorithm improves values of these measures, it can be stated that the algorithm resists not only the mentioned basic attacks, but also some of the more complicated attacks based on these general attacks. A detailed analysis of robustness against all published attacks would be extensive, so this paper investigates only robustness against certain general attacks.

4.3.1 Statistical Attacks

Robustness of image encryption algorithms against statistical attacks can be examined by several measures. First of all, usage of encryption should reduce significant peaks present in the histogram of a plain image. This situation is illustrated in Figure 13 by histograms of plain image *lenaG* and its version encrypted by key K_1.

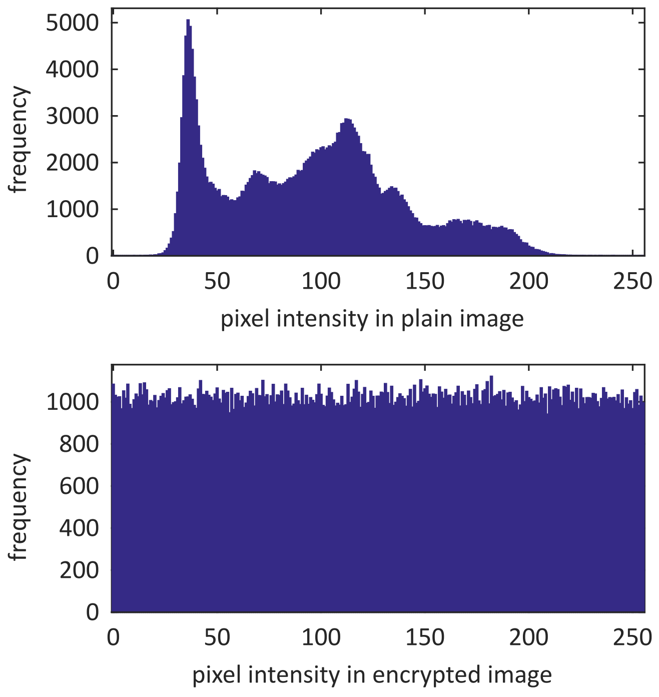


Figure 13. Comparison of image histograms

Secondly, the encryption algorithms should suppress the correlation of adjacent image pixels. This feature can be shown by correlation diagrams, which use intensities of two pixels as coordinates of one plotted point. If pixel intensities are highly correlated, the diagram should have points near line $y = x$. Encryption should result in nearly uniform distribution of points in these diagrams. An example of correlation diagrams for plain image *lenaG* and its version encrypted by key K_1 is shown in Figure 14. Both diagrams contain 1 000 points, which were plotted according to intensities of 1 000 randomly chosen pairs of horizontally adjacent image pixels.

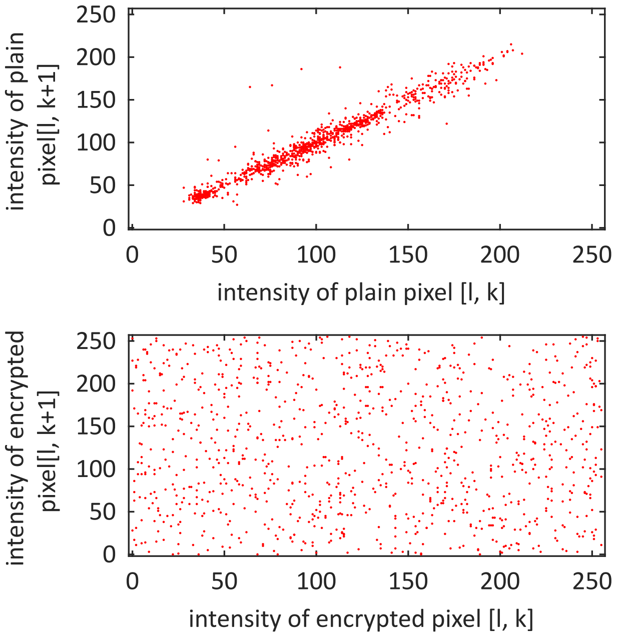


Figure 14. An example of correlation diagrams

Various image encryption algorithms can be compared by numerical parameters. These measures include correlation coefficients ρ, which are calculated separately for horizontally (ρ_h), vertically (ρ_v) and diagonally (ρ_d) adjacent pairs of image pixels and entropy H. Both correlation coefficients and entropy are computed individually for each color plane of the analyzed image.

Correlation coefficients ρ can be calculated by Equation (10):

$$\rho = \frac{\sum_{pp=1}^{num_{pp}} \left(vec_1(pp) - \overline{vec_1}\right) \cdot \left(vec_2(pp) - \overline{vec_2}\right)}{\sqrt{\sum_{pp=1}^{num_{pp}} \left(vec_1(pp) - \overline{vec_1}\right)^2 \cdot \sum_{pp=1}^{num_{pp}} \left(vec_2(pp) - \overline{vec_2}\right)^2}} \; [\text{-}] \tag{10}$$

where $pp = 1, 2, \dots, num_{pp}$ is an index of pixel pair, num_{pp} is total amount of pixel pairs, vectors vec_1 and vec_2 contain intensities of the first and the second pixels from pixel pairs, respectively, and $\overline{vec}$ denotes arithmetic mean of vector vec.

Entropy H is computed by applying Equation (11):

$$H = -\sum_{in=0}^{2^L-1} p(in) \cdot \log_2\left(p(in)\right) \; [\text{bits/px}] \tag{11}$$

where L is color depth of color plane, in denotes intensity of image pixel and $p(in)$ stands for probability of occurrence of pixel with intensity in. The theoretical upper bound of entropy H is given by color depth of the color planes.

Computed values of correlation coefficients ρ and entropy H are presented in columns 4 to 7 of Table 6. Symbol "–" denotes either the only color plane for grayscale images (in the second column) or usage of plain images (in the third column).

4.3.2 Differential Attacks

Differential attacks are used for revealing properties of image encryption algorithms by comparing two encrypted images E_1 and E_2. These two images were created by encryption of plain images P_1 and P_2, where P_2 is slightly modified version of P_1. The modification should be minimal, i. e. intensity of one image pixel from one color plane is changed by one level.

Robustness of image encryption algorithms against differential attacks is tested by two measures: Number of Pixel Change Ratio (NPCR) and Unified Average Changing Intensity (UACI) [42]. While the first one only counts the amount of different pixels, the second one also takes into account the size of differences.

NPCR for images E_1 and E_2 is computed as Equation (12):

$$NPCR = \frac{100}{h \cdot w} \sum_{l=1}^{h} \sum_{k=1}^{w} Diff_{mat}(l,k) \ [\%] \tag{12}$$

where h is height of images E_1 and E_2, w is their width, l and k are line and column indexes and $Diff_{mat}$ is a difference matrix, $Diff_{mat}(l,k) = 1$ if $E_1(l,k) \neq E_2(l,k)$, $Diff_{mat}(l,k) = 0$ otherwise.

UACI for the same pair of images is calculated as Equation (13):

$$UACI = \frac{100}{h \cdot w} \sum_{l=1}^{h} \sum_{k=1}^{w} \frac{|E_1(l,k) - E_2(l,k)|}{2^L - 1} \ [\%] \tag{13}$$

where L is color depth of color plane.

Calculated values of NPCR and UACI are included in columns 8 and 9 of Table 6. Each value is an arithmetic mean of 100 repeated measurements with randomly chosen modified pixel intensity in the plain image. Symbol "–" denotes either the only color plane for grayscale images (in second column) or usage of plain images (in third column). NPCR and UACI of plain images could not be computed as the modification of one plain image pixel intensity is not spread to other intensities (images are not encrypted).

The paper by Wu et al., which analyzed NPCR and UACI [42] also mentions so-called expected values of these parameters for certain resolutions of encrypted images. If the computed values of NPCR and UACI are greater than the expected values, it can be concluded that encryption algorithm successfully suppressed the similarity of plain images P_1 and P_2. Wu et al. also defined significance levels α, which can be used for predicting amount of successful differential attacks on a pair of encrypted images.

The computed expected values of NPCR and UACI for a color plane with resolution of 512×512 pixels are 99.6094 % for NPCR and 33.4635 % for UACI [42].

4.3.3 Discussion

Image and Color Plane		Key	ρ_h [–]	ρ_v [–]	ρ_d [–]	H [bits/px]	NPCR [%]	UACI [%]
lena	R	–	0.9749	0.9782	0.9593	7.5889	not computed	
	G		0.9632	0.9729	0.9497	7.106		
	B		0.9376	0.9515	0.9212	6.8147		
	R	K_1	0.0004	0.0013	0.0005	7.9993	99.6101	33.4738
	G		−0.001	−0.0021	−0.0037	7.9994	99.6109	33.4742
	B		−0.0032	−0.002	0.0017	7.9992	99.6103	33.4746
	R	K_2	0.0033	0.0026	−0.001	7.9994	99.6105	33.4742
	G		0.0006	0.0015	−0.0016	7.9993	99.6113	33.4749
	B		0.0016	0.0013	0.003	7.9993	99.6101	33.4743
	R	K_3	0.0028	−0.0012	0.0002	7.9993	99.6101	33.4736
	G		−0.0007	0.002	0.0002	7.9993	99.6109	33.4731
	B		−0.0042	−0.0017	0.0005	7.9993	99.6098	33.4744
lenaG	–	–	0.9679	0.9761	0.955	7.2344	not computed	
		K_1	0.0008	0.0005	−0.0004	7.9992	99.61	33.4714
		K_2	0.0015	0.001	−0.0012	7.9993	99.6099	33.4725
		K_3	0.0051	−0.0004	−0.0001	7.9993	99.6105	33.4704
peppers	R	–	0.9635	0.9663	0.9564	7.3388	not computed	
	G		0.9811	0.9818	0.9687	7.4963		
	B		0.9665	0.9664	0.9475	7.0583		
	R	K_1	0.0007	−0.005	0.0009	7.9994	99.6112	33.4719
	G		−0.0009	0.0006	−0.0018	7.9993	99.6106	33.4727
	B		0.0013	−0.0011	0.0026	7.9993	99.611	33.4712
	R	K_2	−0.0007	0.0004	−0.0026	7.9994	99.6114	33.4707
	G		−0.0028	−0.0015	0.0031	7.9993	99.6103	33.4718
	B		−0.0005	−0.0019	0.0011	7.9993	99.6108	33.4723
	R	K_3	0.0034	−0.0005	0.0006	7.9992	99.61	33.471
	G		−0.0025	−0.0014	−0.0007	7.9994	99.6102	33.4708
	B		−0.0035	0.0003	0.0006	7.9992	99.6109	33.4714
peppersG	–	–	0.9768	0.9792	0.9639	7.5944	not computed	
		K_1	0.0051	0.0007	−0.0001	7.9991	99.6106	33.4721
		K_2	0.0019	−0.0011	−0.0008	7.9991	99.6121	33.4729
		K_3	−0.0002	−0.0006	−0.0016	7.9993	99.6117	33.4735

Table 6. Achieved numerical parameters

Results presented in Table 6 show that encryption by the proposed algorithm helps to decrease values of correlation coefficients ρ. Also, the values of ρ are quite similar for different keys. The biggest difference of ρ caused by usage of other key is present for image *peppersG* where key K_1 produces $\rho_h = 0.0051$, while other keys produce values of 0.0019 and -0.0002, respectively. However, the high value of ρ_h is balanced by values of ρ_v and ρ_d, which are better for K_1. Values of ρ for other two images (*black* and *blackG*) are affected by their low resolution and the simplicity of their scene (all image pixels have zero intensity). Encryption of image *blackG* by key K_1 resulted in $\rho_h = -0.0191$, $\rho_v = -0.0616$ and $\rho_d = -0.0181$.

The encryption also increases entropy H of the images. Obtained values are close to theoretical bound of 8 bits per pixel and they are more uniform among various color planes. Usage of different keys does not significantly affect the values of H. Image *blackG* encrypted by key K_1 produced $H = 7.5929$ bits per pixel.

Values of NPCR and UACI are similar for various combinations of color planes and keys applied on individual images. Images *black* and *blackG* have greater variance of these values as their low resolution makes even slight differences in amount of changes (NPCR) or their intensity (UACI) more noticeable. Examples of NPCR and UACI for image *blackG* encrypted by key K_1 are 99.6719 % and 33.4304 %, respectively.

All arithmetic means of NPCR and UACI for images *lena*, *lenaG*, *peppers* and *peppersG* are higher than the expected values. However, some of the 100 measurements have lower values than the expected value, which causes limited confidence about possibility of a successful differential attack. Figure 15 illustrates 100 repeated measurements of NPCR for image *lenaG* encrypted by key K_1. The displayed significance level $\alpha = 0.001$ results in a confidence level of 99.9 % (NPCR > 99.5717 %). Therefore 1 out of 1 000 predictions of a possible differential attack can be wrong [42].

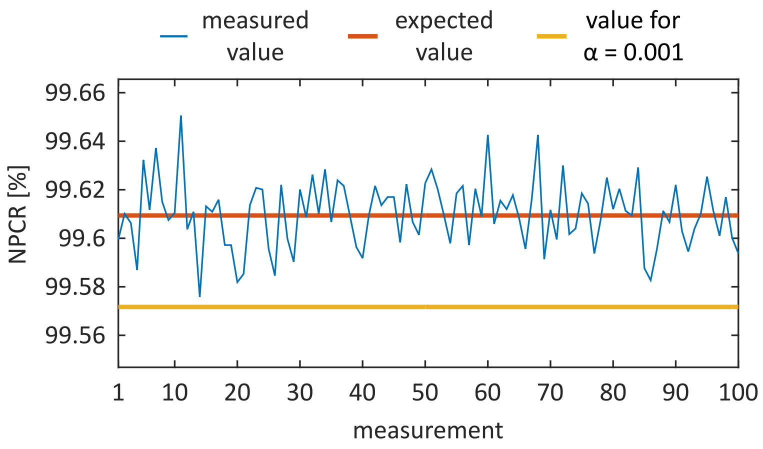


Figure 15. The illustration of 100 measured NPCR values

The properties of the proposed image encryption algorithm regarding commonly used attacks in the field of image encryption can be evaluated from several aspects. Firstly, the key space of the proposed algorithm is big enough for preventing brute-force attacks. Secondly, the design of the proposed algorithm ensures certain level of sensitivity to both used keys and plain or encrypted images. Also, the architecture of the proposed algorithm contains a plaintext related stage, which eliminates application of Solak's attack [13]. Finally, the robustness against statistical and differential attacks can be evaluated by values of correlation coefficients ρ, entropy H, NPCR and UACI.

Therefore it can be concluded that the proposed image encryption algorithm is robust against all currently known attacks in the field of image encryption.

4.4 Analysis of Computational Speed

Encryption and decryption times are other important properties of the image encryption algorithms. The speed of performed operations can be also evaluated via encryption speed v_{enc} or decryption speed v_{dec}, which take into account the resolution and color depth of images (Equation (14)):

$$v_{oper} = \frac{h \cdot w \cdot d}{2^{20} \cdot t_{oper}} \text{ [MB/s]} \tag{14}$$

where h, w and d are height, width and color depth of image, 2^{20} is a constant representing amount of bytes in a megabyte and t_{oper} is time in seconds needed for an operation (either encryption or decryption).

Considering different specifications of various computers used for experiments, the speed of image encryption algorithms can be given by the number of processor cycles required for operations with one byte of data. These values are denoted as cyc_{enc} for encryption of one image byte and cyc_{dec} for decryption of one image byte. The number of cycles necessary for an operation (either encryption or decryption) with one byte of data cyc_{oper} can be computed by Equation (15):

$$cyc_{oper} = \frac{f_{cpu}}{v_{oper}} \text{ [cycles/B]} \tag{15}$$

where f_{cpu} is processor clock frequency measured in Hz and v_{oper} is speed of the investigated operation given in B/s. Please note that the computed values of cycles required for operations with one byte expect that only one core of the processor is utilized (at 100 %) for the purposes of the image encryption algorithms.

Times, speeds and numbers of processor cycles for operations with images and keys from the experimental set are presented in Table 7. The times are arithmetic means of 100 repeated measurements. Speeds and numbers of the processor cycles were computed from these means.

Results shown in Table 7 lead to several conclusions. Firstly, the speed of operations does not depend on the key used. Secondly, both the encryption or

Image	Key	t_{enc} [ms]	t_{dec} [ms]	v_{enc} [MB/s]	v_{dec} [MB/s]	cyc_{enc} [cycles/B]	cyc_{dec} [cycles/B]
lena	K_1	436.2245	420.5649	1.7193	1.7833	1454.08	1401.9
	K_2	436.3845	420.3934	1.7187	1.784	1454.59	1401.35
	K_3	436.7181	421.081	1.7174	1.7811	1455.69	1403.63
lenaG	K_1	126.2499	122.7206	1.9802	2.0371	1262.5	1227.23
	K_2	126.9473	122.6332	1.9693	2.0386	1269.49	1226.33
	K_3	126.905	122.6084	1.97	2.039	1269.04	1226.09
peppers	K_1	436.3264	419.8888	1.7189	1.7862	1454.42	1399.62
	K_2	436.4397	421.8564	1.7185	1.7779	1454.76	1406.15
	K_3	435.8724	419.6389	1.7207	1.7873	1452.9	1398.76
peppersG	K_1	127.1529	122.8078	1.9661	2.0357	1271.55	1228.08
	K_2	126.8485	122.7695	1.9709	2.0363	1268.46	1227.72
	K_3	127.3672	122.8902	1.9628	2.0343	1273.69	1228.92
black	K_1	2.557	2.6487	0.5729	0.553	4363.76	4520.8
	K_2	2.5451	2.6251	0.5756	0.558	4343.29	4480.29
	K_3	2.535	2.6628	0.5779	0.5501	4326.01	4544.63
blackG	K_1	1.5223	1.5587	0.3207	0.3133	7795.45	7979.57
	K_2	1.5124	1.564	0.3229	0.3122	7742.34	8007.69
	K_3	1.4991	1.5803	0.3257	0.309	7675.78	8090.61

Table 7. Measured times, speeds and the numbers of the processor cycles

decryption speeds and the number of the processor cycles required for processing of one byte of data for images with the same resolution and color depth are very similar. Also, the speeds for images with a higher resolution (such as *lena* and *peppers*) are lower than the speeds for images with a lower resolution. This can be caused by generating of longer PRSs since LM (1) is a recursive function where each currently generated iterate (x_{n+1}) requires the previous iterate (x_n) for its computations. The lower speeds result in an increased number of the processor cycles required for processing of one byte of data.

The values for images *black* and *blackG* are distorted by rather low resolution of the images (32×16 pixels). For images with resolution of 512×512 pixels, the encryption speeds v_{enc} are approx. 1.7 MB/s for true color images and approx. 1.95 MB/s for grayscale images. The decryption speeds v_{dec} are slightly higher.

4.5 Comparison with Other Approaches

The comparison of numeric parameters obtained by image encryption algorithms is not an easy task, as researchers tend to use different plain images and various measures. This section summarizes results reported in papers [20, 21, 24, 26, 27, 29, 30, 31, 32], which all describe plaintext related image encryption algorithms. The comparison of parameters presented in Table 8 is divided into two parts – the first part includes values obtained by papers that used true color image *lena*, while the second part shows values from papers that utilized grayscale image *lenaG*.

Results for this paper were obtained by using key K_1. Some papers used plain images with different resolution: refs. [20, 30] used resolution of 256×256 pixels, while refs. [24, 26] used resolution of 357×317 pixels. Algorithm published in [27] was already broken in [28]. An approach presented in [32] is asymmetric (meaning that the encryption and decryption keys are different).

The numbers of the processor cycles required for encryption of one byte cyc_{enc} were calculated with respect to both resolutions of used images and specifications of the computers used. All compared approaches obtained the encryption times used for calculation of cyc_{enc} in various versions of the MATLAB computational environment. None of the compared algorithms mentioned the usage of multiple processor cores.

If some paper presented multiple values for one parameter, the best results were chosen for the comparison. The most outstanding value was highlighted by italics for each parameter.

Approach	ρ_h [–]	ρ_v [–]	ρ_d [–]	H [bits/px]	NPCR [%]	UACI [%]	cyc_{enc} [cycles/B]
red color plane of true color image *lena* (512×512 pixels)							
proposed	*0.0004*	*0.0013*	*0.0005*	7.9993	99.6101	33.4738	1454.08
ref. [21]	−0.0029	−0.015	0.0129	*7.9997*	99.62	*33.51*	∼2270
ref. [27]	0.0135	not reported		7.9974	*99.63*	33.31	*648.53*
grayscale image *lenaG* (512×512 pixels)							
proposed	*0.0008*	*0.0005*	*−0.0004*	7.9992	99.61	33.4714	1262.5
ref. [20]	0.0088	−0.0087	−0.006	7.9902	*99.62*	33.46	9063.44
ref. [24]	−0.0046	−0.0511	−0.0168	*7.9993*	99.6101	33.4679	8230.32
ref. [26]	−0.0084	0.0041	−0.0463	7.9984	99.6077	33.4441	3366.6
ref. [29]	0.0044	0.0151	0.0012	*7.9993*	*99.62*	33.45	15120.97
ref. [30]	−0.0037	−0.0029	0.0047	7.9975	99.5956	*33.5512*	43151.97
ref. [31]	0.0013	0.0008	0.0066	*7.9993*	99.6107	33.4436	5185.19
ref. [32]	0.0042	0.0022	−0.0045	7.9992	99.1802	33.3483	*795.17*

Table 8. Comparison of numerical parameters

As seen in Table 8, the proposed solution provides the best values of correlation coefficients ρ among presented values. The results for entropy H are similar, and in most cases close to the theoretical bound of 8 bits per pixel. The highest entropy value for red color plane of color image *lena* is achieved by approach [21], while highest entropy values for grayscale image *lenaG* are obtained by algorithms [24, 29, 31].

Majority of NPCR and UACI results are higher than the expected values. The best value of NPCR for color images was obtained by design [27], however this algorithm was already broken [28]. NPCR values for grayscale images are similar, except for proposal [32]. The best values were produced by approaches [20, 29]. The best UACI values were achieved by algorithm [21] for true color images and by approach [30] for grayscale images.

Reported numbers of the processor cycles required for encryption of one byte cyc_{enc} are greatly influenced by the architecture of the algorithms. This fact is most visible in the approaches [29, 30], which used hash functions and the algorithm [27], which inserted plaintext related parameter into the encrypted image. While hash functions significantly slow down the whole algorithms, the other solution needs a relatively small number of processor cycles for the encryption. However, as it was already pointed out, this solution was already broken [28]. Another approach with different properties was described in [32], where an asymmetric image encryption algorithm was proposed. Except these solutions, the design proposed in this paper is the fastest.

5 CONCLUSIONS

This paper described an image encryption algorithm, which employed the Mojette transform for plaintext related chaining of pixel intensities. The values of Mojette bins, which are going to be combined with image pixels are chosen by pixel intensities that are yet not encrypted, therefore this proposal is plaintext related. Moreover, the Mojette bins are computed also from pixel intensities of plain images.

Parameters of Mojette bins were chosen in a way that each bin is affected by pixel intensities from various color planes. Also, the pixel intensities that are summed by the Mojette transform are chosen from different columns of the processed images.

As the combinations of images and computed Mojette bins are done with whole rows of pixel intensities, the proposed solution is among the fastest plaintext related image encryption designs. Also, the experimental results show that the algorithm proposed in this paper reaches excellent values of correlation coefficients. The obtained values of entropy, NPCR and UACI are comparable with other proposals.

Probably the biggest drawback of the proposed algorithm is the fact that not all of 100 performed NPCR and UACI measurements are higher than the expected values. This disadvantage could be improved in the future.

Acknowledgment

The work was supported by the research grant APVV-17-0208.

REFERENCES

[1] Zhang, Y.-Q.—Wang, X.-Y.: A Symmetric Image Encryption Algorithm Based on Mixed Linear-Nonlinear Coupled Map Lattice. Information Sciences, Vol. 273, 2014, pp. 329–351, doi: 10.1016/j.ins.2014.02.156.

[2] Hua, Z.—Zhou, Y.: Image Encryption Using 2D Logistic-Adjusted-Sine Map. Information Sciences, Vol. 339, 2016, pp. 237–253, doi: 10.1016/j.ins.2016.01.017.

[3] Gueron, S.: Intel Advanced Encryption Standard (AES) New Instructions Set. Cited 2019-04-07. Available online at: `https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf`.

[4] Liu, L.—Miao, S.: A New Image Encryption Algorithm Based on Logistic Chaotic Map with Varying Parameter. SpringerPlus, Vol. 5, 2016, No. 1, pp. 289–300, doi: 10.1186/s40064-016-1959-1.

[5] Fridrich, J.: Symmetric Ciphers Based on Two-Dimensional Chaotic Maps. International Journal of Bifurcation and Chaos, Vol. 8, 1998, No. 6, pp. 1259–1284, doi: 10.1142/S021812749800098X.

[6] Chen, G.—Mao, Y.—Chui, C. K.: A Symmetric Image Encryption Scheme Based on 3D Chaotic Cat Maps. Chaos, Solitons and Fractals, Vol. 21, 2004, No. 3, pp. 749–761, doi: 10.1016/j.chaos.2003.12.022.

[7] Pareek, N. K.—Patidar, V.—Sud, K. K.: Image Encryption Using Chaotic Logistic Map. Image and Vision Computing, Vol. 24, 2006, No. 9, pp. 926–934, doi: 10.1016/j.imavis.2006.02.021.

[8] Wong, K.-W.—Kwok, B. S.-H.—Law, W.-S.: A Fast Image Encryption Scheme Based on Chaotic Standard Map. Physics Letters A, Vol. 372, 2008, No. 15, pp. 2645–2652, doi: 10.1016/j.physleta.2007.12.026.

[9] Ye, G.—Wong, K.-W.: An Efficient Chaotic Image Encryption Algorithm Based on a Generalized Arnold Map. Nonlinear Dynamics, Vol. 69, 2012, No. 4, pp. 2079–2087, doi: 10.1007/s11071-012-0409-z.

[10] Shannon, C. E.: Communication Theory of Secrecy Systems. The Bell System Technical Journal, Vol. 28, 1949, No. 4, pp. 656–715, doi: 10.1002/j.1538-7305.1949.tb00928.x.

[11] Li, S.—Zheng, X.: Cryptanalysis of a Chaotic Image Encryption Method. Proceedings of the 2002 IEEE International Symposium on Circuits and Systems (ISCAS 2002), Phoenix-Scottsdale, May 2002, pp. 708–711, doi: 10.1109/ISCAS.2002.1011451.

[12] Rhouma, R.—Belghith, S.: Cryptanalysis of a New Image Encryption Algorithm Based on Hyper-Chaos. Physics Letters A, Vol. 372, 2008, No. 38, pp. 5973–5978, doi: 10.1016/j.physleta.2008.07.057.

[13] Solak, E.—Çokal, C.—Yildiz, O. T.—Biyikoğlu, T.: Cryptanalysis of Fridrich's Chaotic Image Encryption. International Journal of Bifurcation and Chaos, Vol. 20, 2010, No. 5, pp. 1405–1413, doi: 10.1142/S0218127410026563.

[14] Xie, E. Y.—Li, C.—Yu, S.—Lü, J.: On the Cryptanalysis of Fridrich's Chaotic Image Encryption Scheme. Signal Processing, Vol. 132, 2017, pp. 150–154, doi: 10.1016/j.sigpro.2016.10.002.

[15] Broda, M.—Hajduk, V.—Levický, D.: Universal Statistical Steganalytic Method. Journal of Electrical Engineering, Vol. 68, 2017, No. 2, pp. 117–124, doi: 10.1515/jee-2017-0016.

[16] Oravec, J.—Turán, J.: Substitution Steganography with Security Improved by Chaotic Image Encryption. Proceedings of the 2017 IEEE 14th International Scientific

Conference on Informatics (Informatics 2017), Poprad, November 2017, pp. 284–288, doi: 10.1109/INFORMATICS.2017.8327261.

[17] Fang, D.—Sun, S.: A New Scheme for Image Steganography Based on Hyperchaotic Map and DNA Sequence. Journal of Information Hiding and Multimedia Signal Processing, Vol. 9, 2018, No. 2, pp. 392–399.

[18] Abundiz-Pérez, F.—Cruz-Hernández, C.—Murillo-Escobar, M. A.—López-Gutiérrez, R. M.—Arellano-Delgado, A.: A Fingerprint Image Encryption Scheme Based on Hyperchaotic Rössler Map. Mathematical Problems in Engineering, Vol. 2016, 2016, Art. No. 2670494, pp. 1–15, doi: 10.1155/2016/2670494.

[19] Chen, X.—Hu, C.-J.: Adaptive Medical Image Encryption Algorithm Based on Multiple Chaotic Mapping. Saudi Journal of Biological Sciences, Vol. 24, 2017, No. 8, pp. 1821–1827, doi: 10.1016/j.sjbs.2017.11.023.

[20] Fu, C.—Chen, J.-J.—Zou, H.—Meng, W.-H.—Zhan, Y.-F.—Yu, Y.-W.: A Chaos-Based Digital Image Encryption Scheme with an Improved Diffusion Strategy. Optics Express, Vol. 20, 2012, No. 3, pp. 2363–2378, doi: 10.1364/OE.20.002363.

[21] Kanso, A.—Ghebleh, M.: A Novel Image Encryption Algorithm Based on a 3D Chaotic Map. Communications in Nonlinear Science and Numerical Simulation, Vol. 17, 2012, No. 7, pp. 2943–2959, doi: 10.1016/j.cnsns.2011.11.030.

[22] Arroyo, D.—Rhouma, R.—Alvarez, G.—Li, S.—Fernandez, V.: On the Security of a New Image Encryption Scheme Based on Chaotic Map Lattices. Chaos: An Interdisciplinary Journal of Nonlinear Science, Vol. 18, 2008, No. 3, pp. 1–8, doi: 10.1063/1.2959102.

[23] Fu, C.—Hou, S.—Zhou, W.—Liu, W.-Q.—Wang, D.-L.: A Chaos-Based Image Encryption Scheme with a Plaintext Related Diffusion. Proceedings of 2013 9th International Conference on Information, Communications and Signal Processing (ICICS 2013), Tainan, December 2013, pp. 1–5, doi: 10.1109/ICICS.2013.6782914.

[24] Zhang, Y.: A Chaotic System Based Image Encryption Algorithm Using Plaintext-Related Confusion. TELKOMNIKA Indonesian Journal of Electrical Engineering and Computer Science, Vol. 12, 2014, No. 11, pp. 7952–7962.

[25] Fan, H.—Li, M.: Cryptanalysis and Improvement of Chaos-Based Image Encryption Scheme with Circular Inter-Intra-Pixels Bit-Level Permutation. Mathematical Problems in Engineering, Vol. 2017, 2017, Art. No. 8124912, 11 pp., doi: 10.1155/2017/8124912.

[26] Zhang, Y.: The Image Encryption Algorithm with Plaintext-Related Shuffling. IETE Technical Review, Vol. 33, 2016, No. 3, pp. 310–322, doi: 10.1080/02564602.2015.1087350.

[27] Murillo-Escobar, M. A.—Cruz-Hernández, C.—Abundiz-Pérez, F.—López-Gutiérrez, R. M.—Acosta Del Campo, O. R.: A RGB Image Encryption Algorithm Based on Total Plain Image Characteristics and Chaos. Signal Processing, Vol. 109, 2015, pp. 119–131, doi: 10.1016/j.sigpro.2014.10.033.

[28] Fan, H.—Li, M.—Liu, D.—An, K.: Cryptanalysis of a Plaintext-Related Chaotic RGB Image Encryption Scheme Using Total Plain Image Characteristics. Multimedia Tools and Applications, Vol. 77, 2018, No. 15, pp. 20103–20127, doi: 10.1007/s11042-017-5437-8.

[29] Chai, X.—Gan, Z.—Zhang, M.: A Fast Chaos-Based Image Encryption Scheme with a Novel Plain Image-Related Swapping Block Permutation and Block Diffusion. Multimedia Tools and Applications, Vol. 76, 2017, No. 14, pp. 15561–15585, doi: 10.1007/s11042-016-3858-4.

[30] Wang, X.—Zhu, X.—Wu, X.—Zhang, Y.: Image Encryption Algorithm Based on Multiple Mixed Hash Functions and Cyclic Shift. Optics and Lasers in Engineering, Vol. 107, 2018, pp. 370–379, doi: 10.1016/j.optlaseng.2017.06.015.

[31] Li, Z.—Peng, C.—Li, L.—Zhu, X.: A Novel Plaintext-Related Image Encryption Scheme Using Hyper-Chaotic System. Nonlinear Dynamics, Vol. 94, 2018, No. 2, pp. 1319–1333, doi: 10.1007/s11071-018-4426-4.

[32] Oravec, J.—Turán, J.—Ovseník, Ľ.—Ivaniga, T.—Solus, D.—Márton, M.: Asymmetric Image Encryption Approach with Plaintext-Related Diffusion. Radioengineering, Vol. 27, 2018, No. 1, pp. 281–288, doi: 10.13164/re.2018.0281.

[33] May, R. M.: Simple Mathematical Models with Very Complicated Dynamics. Nature, Vol. 261, 1976, No. 5560, pp. 459–467, doi: 10.1038/261459a0.

[34] Gleick, J.: Chaos. Vintage Books, London, UK, 1998.

[35] Guédon, J.-P.—Barba, D.—Burger, N.: Psychovisual Image Coding via an Exact Discrete Radon Transform. Proceedings of Visual Communications and Image Processing '95 (VCIP 95), Taipei, April 1995. Proceedings of the SPIE, Vol. 2501, 1995, pp. 562–572, doi: 10.1117/12.206765.

[36] Guédon, J.-P.: The Mojette Transform. Wiley, London, UK, 2009.

[37] Turán, J.—Szoboszlai, P.—Vásárhelyi, J.: Mojette Transform Software – Hardware Implementations and Its Applications. Infocommunications Journal (Híradástechnika), Vol. 3, 2011, No. 1, pp. 39–47.

[38] Guédon, J.-P.—Normand, N.: The Mojette Transform: The First Ten Years. In: Andres, E., Damiand, G., Lienhardt, P. (Eds.): Discrete Geometry for Computer Imagery (DGCI 2005). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 3429, 2005, pp. 79–91, doi: 10.1007/978-3-540-31965-8_8.

[39] Verbert, P.—Ricordel, V.—Guédon, J.-P.: Analysis of Mojette Transform Projections for an Efficient Coding. Proceedings of Workshop on Image Analysis for Multimedia Interactive Services 2004 (WIAMIS 2004), Lisboa, April 2004, pp. 1–4.

[40] Oravec, J.—Ovseník, Ľ.—Turán, J.: An Iterative Approach for Image Coding Using Mojette Transform. Proceedings of the 2016 26th International Conference Radioelektronika (Radioelektronika 2016), Košice, April 2016, pp. 1–4, doi: 10.1109/RADIOELEK.2016.7477351.

[41] Oravec, J.—Turán, J.—Ovseník, Ľ.: LSB Steganography with Usage of Mojette Transform for Secret Image Scrambling. Proceedings of 23rd International Conference on Systems, Signals and Image Processing (IWSSIP 2016), Bratislava, May 2016, pp. 1–4, doi: 10.1109/IWSSIP.2016.7502754.

[42] Wu, Y.—Noonan, J. P.—Agaian, S.: NPCR and UACI Randomness Tests for Image Encryption. Journal of Selected Areas in Telecommunications, Vol. 2, 2011, No. 4, pp. 31–38.

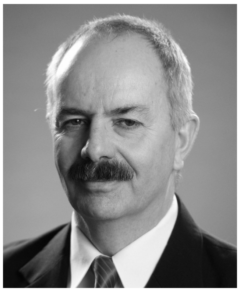
Ľuboš Ovseník received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 1990 and 2002. He works at the Technical University of Košice, currently as Associate Professor. His research interests are fiber optic communication systems and sensor networks.

Ján Turán received Ing. (M.Sc.) degree in physical engineering with honours from the Czech Technical University, Prague, Czech Republic, in 1974, and R.N.Dr. degree in experimental physics with honours from Charles University, Prague, Czech Republic, in 1980. He received a C.Sc. (Ph.D.) and Dr.Sc. (D.Sc.) degrees in radioelectronics from Technical University of Košice, Slovakia, in 1983, and 1992, respectively. Since March 1979, he has been at the Technical University of Košice as Full Professor for electronics and information technology. His research interests include digital signal processing and fiber optics, communication and sensing.

Tomáš Huszaník received his M.Sc. degree from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2017. Currently, he is a Ph.D. student at the same department. His research interests include all optical networks and degradation mechanisms in all optical WDM systems.

Jakub Oravec received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2015 and 2019. The topic of his dissertation was researching transform techniques in the new fields of image processing. His research interests included image encryption, steganography and digital image processing.

Ondrej Kováč received his M.Sc. and Ph.D. degrees from the Department of Electronics and Multimedia Communications, Technical University of Košice in 2011 and 2015. He works at the Technical University of Košice, currently as Assistant Professor. His Ph.D. thesis was focused on texture generation, 3D modeling and coding of human head.



Milan Oravec received Ing. (M.Sc.) and C.Sc. (Ph.D.) degrees from the Department of Safety and Quality of Production, Technical University of Košice in 1984 and 1992. He works at the Technical University of Košice, currently as Full Professor. His research interests include prevention of industrial accidents and safety assessments of critical infrastructure systems.

Computing and Informatics, Vol. 38, 2019, 679–700, doi: 10.31577/cai_2019_3_679

LEARNING SPARQL QUERIES FROM EXPECTED RESULTS

Jedrzej Potoniec

Faculty of Computing
Poznan University of Technology
ul. Piotrowo 3
60-965 Poznan, Poland
e-mail: `jpotoniec@cs.put.poznan.pl`

Abstract. We present LSQ, an algorithm learning SPARQL queries from a subset of expected results. The algorithm leverages grouping, aggregates and inline values of SPARQL 1.1 in order to move most of the complex computations to a SPARQL endpoint. It operates by building and testing hypotheses expressed as SPARQL queries and uses active learning to collect a small number of learning examples from the user. We provide an open-source implementation and an on-line interface to test the algorithm. In the experimental evaluation, we use real queries posed in the past to the official DBpedia SPARQL endpoint, and we show that the algorithm is able to learn them, 82 % of them in less than a minute and asking the user just once.

Keywords: SPARQL, RDF, active learning

1 INTRODUCTION

Querying information with a complex structure is an inherently hard task for a user. The user must spend a lot of time learning a vocabulary and relations used in the data. This will become more and more important, as we develop more complex artificial intelligence systems, using vast amount of information encoded in a complex representation formalism. In this paper, we aim to remedy this issue in the context of information represented as an Resource Description Framework (RDF) graph.

Consider the following use case scenario: a user has some informal criteria to select a subset of nodes of a graph. He/she knows some of the relevant nodes in the graph, he/she also knows some of the irrelevant nodes. Moreover, he/she can distinguish a relevant node from an irrelevant one for some price, e.g. by spending his/her time verifying if a node is a relevant one or not. He/she wants to obtain a formal query corresponding to the informal criteria to be able to query the graph.

To address this use case, we propose an algorithm for learning a SPARQL query corresponding to these criteria. The algorithm is designed in such a way that it moves most of the computational work to a SPARQL endpoint by posing quite complex queries to it. It is a reasonable decision: the RDF graph is stored there, so that is the best place to perform any optimization.

Our contribution is as follows: we present the first algorithm for constructing SPARQL queries from examples, which is *at the same time interactive and saves computational power of the client*, by moving most of the computations to the SPARQL server.

The rest of the paper is organized as follows: in Section 2 we present a short overview of the most important aspects of RDF and SPARQL. Section 3 discusses related research. The description of the algorithm, along with the required definitions, is presented in Section 4, and in Section 5 we introduce a web application implementing the presented algorithm. Section 6 presents an experimental evaluation of the algorithm in two setups:

1. using a set of real-world SPARQL queries as a gold standard;
2. using a benchmark for the task of binary classification in the structured machine learning.

We conclude in Section 7.

2 PRELIMINARIES

2.1 Resource Description Framework

Resource Description Framework (RDF) is a framework designed to represent information in the Web in a way accessible for machines [33]. The core concept of RDF is an *RDF triple*, which consists of a *subject*, a *predicate* and an *object*. A customary meaning assigned to a triple is such that the entity represented by the subject is in relation denoted by the predicate with the entity represented by the object.

A set of triples constitutes an *RDF graph*, where subjects and objects jointly form the set of *nodes* of the graph and each triple represents a directed edge from the subject of the triple to the object, labeled with the predicate.

An *RDF term* is either an IRI (Internationalized Resource Identifier), that serves as a global identifier for some entity in the universe of discourse; a blank node, that is a local identifier for some entity in the universe of discourse; a literal, that represents a concrete value such as a string of characters or a number. Usually, the non-unique name assumption is made, stating that a single entity may be referenced by multiple identifiers. The subject of a triple may be either an IRI or a blank node, the predicate must be an IRI and the object may be an arbitrary RDF term.

In formalizing RDF we follow [22] and we start by introducing three pairwise disjoint sets: the set of all IRIs $\mathbf{I}$, the set of all blank nodes $\mathbf{B}$ and the set of all literals $\mathbf{L}$. A subject of a triple is any element of the set $\mathbf{I} \cup \mathbf{B}$, the predicate of a triple is an element of the set $\mathbf{I}$, while object is any element of the set $\mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. An RDF graph G, being a set of triples, is thus a subset of a Cartesian product: $G \subseteq (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$.

Throughout this work, we use a subset of the Turtle syntax [7], where each triple is written in the order subject, predicate, object and ends with a dot. We represent IRIs using the prefix notation well-known from XML, i.e., `prefix:localName`, and list the common prefixes used throughout this work in Table 1. A literal is represented in quotation marks, with its datatype after `^^`. For example, the triple `dbr:Warsaw dbo:populationTotal "1740119"^^xsd:integer.` states that an entity denoted by the IRI `dbr:Warsaw` (in full: `http://dbpedia.org/resource/Warsaw`) has a total population of 1 740 119 citizens.

Prefix	Corresponding IRI
`dbr:`	`http://dbpedia.org/resource/`
`dbo:`	`http://dbpedia.org/ontology/`
`dbp:`	`http://dbpedia.org/property/`
`dct:`	`http://purl.org/dc/terms/`
`xsd:`	`http://www.w3.org/2001/XMLSchema#`

Table 1. The common IRI prefixes used in the paper and their corresponding IRIs

One of the most prominent applications of RDF is the Web of Data (also called Linked Data)[1], a large, distributed collection of RDF graphs concerning different topics from life sciences, to media, to governmental data. In the center of the Web of Data is DBpedia[2], the result of a complex knowledge extraction process from Wikipedia [1, 3, 20].

2.2 SPARQL Query Language

Every form of information representation requires a querying language and RDF is not an exception here. SPARQL Query Language is by far the most popular query language for RDF, built around graph pattern matching, yet having a lot in common with SQL known from relational databases [14]. Below, we summarize the most important aspects of SPARQL.

A SPARQL SELECT query is of form

```
SELECT head
WHERE { pattern }
[GROUP BY variables]
```

[1] `http://lod-cloud.net/`
[2] `http://dbpedia.org`

```
[HAVING criterion]
[LIMIT limit]
```

where the square brackets denote optional parts of the query. Every time the pattern is matched against the queried RDF graph, it yields a mapping from variables of the query to the RDF terms.

A *variable* in SPARQL is prefixed with a question mark, e.g. `?var`. A *blank node* `[]` denotes an anonymous, existentially quantified variable. A *triple pattern* is an RDF triple with an arbitrary number of components replaced by variables, e.g. `dbr:Warsaw dbo:populationTotal ?population.`. A *Basic Graph Pattern* (BGP) is a set of triple patterns that are matched jointly against the queried graph.

Denote by $\mathbf{V}$ the set of all variables. Let G be an RDF graph, P a BGP and denote by $vars(P)$ the set of all variables in P. We consider a mapping, i.e., partial function $\mu\colon \mathbf{V} \mapsto \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$. We say that μ is a *solution* to P if the domain of μ is the set $vars(P)$ and there exists a function $\sigma\colon \mathbf{B} \mapsto \mathbf{I} \cup \mathbf{B} \cup \mathbf{L}$ such that $\mu(\sigma(P)) \subseteq G$. We abuse the notation and by applying μ and σ to P we understand applying it element-wise to each part of each triple pattern in P.

A BGP may be optionally extended with a filter expression `FILTER(`*`condition`*`)`, to the effect that if μ(`condition`) evaluates to false, then μ is rejected. An alternative of two BGPs may be realized using the UNION keyword: `bgp1 UNION bgp2`, and we say that μ is a solution for it if it is a solution to either one of the BGPs. Finally, the clause `VALUES ?var` {*`allowed assignments`*} limits the set of allowed assignments for the variable `?var` in the solutions to the RDF terms listed in the curly braces.

The multi-set of solutions μ can be represented as a table, which can be then processed according to the standard SQL semantics of GROUP BY, HAVING and LIMIT. We thus abstain from formally defining their semantics and refer the interested reader to [14].

SPARQL is frequently employed with the SPARQL Protocol, which defines means to use HTTP (Hypertext Transfer Protocol) to query an RDF graph [32]. A server capable of answering SPARQL queries posed using SPARQL Protocol is called a *SPARQL endpoint* and is identified by its URL.

3 RELATED WORK

For the past few years, researchers proposed many approaches for querying an RDF graph alternative to writing a formal query by hand. Roughly, they can be divided into a few categories: faceted browsing, natural language interfaces, visual interfaces and recommendations. In faceted browsing a user is presented with an interface dynamically generated from an RDF graph to filter the graph according to his/her needs, e.g. see [23]. Also variants tailored to a specific types of data were proposed, e.g. [21] describes an interface for geospatial data. A recent system *Sparklis* combines faceted browsing with natural language generation to enable a non-expert user to query an RDF graph [11].

Natural language interfaces are concerned with answering a query specified in a natural language, e.g. by translating it to SPARQL and then posing it to an endpoint. These systems are frequently tailored to some specific task and/or a specific RDF graph. For example, *Xser* [34] is a system for answering factual questions and *CubeQA* [15] is designed to answer statistical queries requiring aggregate functions. A recent survey [16] gives a comprehensive overview of such systems.

The visual interfaces offer a possibility of constructing a query by visual means, e.g. by navigating over a displayed part of a graph [9] or by constructing a SPARQL query from building blocks [4].

A system using recommendations still requires from a user knowledge of SPARQL, but it helps to deal with an unknown vocabulary. [13] recommends predicate names based on an inferred type of a variable in a triple pattern. [6] describes a method for recommending query terms based on a graph summary, while [5] pushes it even further by performing the recommendations on-line, by posing appropriate queries to a SPARQL endpoint.

A similar idea to ours was already proposed in [19]. The main difference are the assumptions: [19] performs all the computation on the client-side, obtaining information about resources using an approach similar to Concise Bounded Description [27] and then computing their intersections. Our approach leverages new features of SPARQL 1.1, namely grouping, aggregates and providing inline values, and thus moves most of the learning complexity directly to a SPARQL endpoint.

Methods for learning queries from a set of examples were also discussed in other contexts. [17] uses a pattern mining approach to learn a set of SPARQL queries, which then are used as binary patterns for a normal classification algorithm. In [24] the authors propose a method for unsupervised mining of data mining features, which directly correspond to SPARQL queries. [18] and [10] are methods for learning Description Logics class expressions from a set of positive and negative examples. The expressions can then be used as queries to an ontological knowledge base to retrieve individuals fulfilling the class expression, e.g. using the *DL Query* tab of *Protégé* [12].

4 LEARNING SPARQL QUERIES

4.1 Basic Concepts

Throughout this section, we use the following example: the user wants to formulate a SPARQL query which allows him/her to query DBpedia for the capitals of states of the European Union. He/she knows some of them: Warsaw, Berlin, Zagreb, Nicosia and Vilnius. He/she can also recognize whether an arbitrary DBpedia IRI refers to one of the capitals by reading a Wikipedia article corresponding to the IRI. Of course, reading consumes his/her time, so he/she wants to limit the number of articles read. He/she also knows that Oslo, a capital of Norway, which is not a European Union member, should not be presented in the results.

Generally speaking, LSQ works by formulating a hypothesis and verifying it with the user.

Definition 1 (hypothesis)**.** A *hypothesis*, denoted $H(?iri)$, consists of SPARQL triple patterns and filters. Every triple pattern in the hypothesis has a fixed predicate (i.e., the predicate is an IRI), the subject and object may be an IRI, a literal or a variable. Every filter contains only an expression of form *variable* `>=` *fixed literal* or *variable* `<=` *fixed literal.* The hypothesis may contain an unlimited number of variables, but there is a single, distinguished variable $?iri$, which must be present in at least one triple pattern. We require the undirected graph corresponding to the basic graph pattern defined by a hypothesis to be a connected graph.

An *empty hypothesis* is a hypothesis which contains no triple patterns or filters. For example, the algorithm may generate the following hypothesis `H(?iri)`:

```
dbr:European_Union dbo:wikiPageWikiLink ?iri .
?iri dct:subject dbr:Category:Capitals_in_Europe .
dbr:Member_state_of_the_European_Union dbo:wikiPageWikiLink ?iri.
```

It may be that the user already has some SPARQL query and wants to extend it by providing examples. We allow the user to provide a BGP $U(?iri)$, such that it shares the distinguished variable $?iri$ with the hypothesis, but all the remaining variables are separate. Some of the variables from $U(?iri)$ may be also present in the head of the final query obtained from the algorithm. We observe that if the user does not know SPARQL or does not have any query to extend, it is sufficient to assume that $U(?iri) = \emptyset$.

Definition 2 (query corresponding to a hypothesis)**.** A *query corresponding to a hypothesis* $H(?iri)$ is a SPARQL SELECT query containing the variable $?iri$ in the head, optionally with other variables coming from $U(?iri)$. The WHERE clause contains the hypothesis $H(?iri)$ and the user-provided BGP $U(?iri)$.

To formulate a hypothesis, LSQ uses a set of positive examples `P` and a set of negative examples `N`. Both sets contain IRIs from the RDF graph. `P` is a subset of IRIs expected in the results of the query corresponding to the final hypothesis. `N` is a set of IRIs which are forbidden to appear in these results. They do not need to be fully specified before the algorithm is run, instead it is enough that the user provides a few IRIs in both sets, and then they are extended during the execution of the algorithm. By `n_pos` we denote for the number of IRIs in the set `P`. In our running example, `P` initially consists of `dbr:Warsaw`, `dbr:Berlin`, `dbr:Zagreb`, `dbr:Nicosia` and `dbr:Vilnius`, and thus `n_pos` = 5. `N` contains only `dbr:Oslo`.

The algorithm uses well-known information retrieval measures. Denote by TP the number of IRIs from the set `P` which were retrieved by the query corresponding to a hypothesis, and by FP the number of IRIs from the set `N` which were retrieved by the query corresponding to the hypothesis. Following [2], we define three measures.

Definition 3 (precision)**.** *Precision* is the fraction of the number of the positive IRIs retrieved by the query corresponding to the hypothesis over the number of both positive and negative IRIs retrieved by the query corresponding to the hypothesis

$$p = \frac{TP}{TP + FP}.$$

Definition 4 (recall)**.** *Recall* is the fraction of the number of the positive IRIs retrieved by the query corresponding to the hypothesis over the number of known positive IRIs `n_pos`

$$r = \frac{TP}{\texttt{n\_pos}}.$$

Definition 5 (F_1 measure)**.** F_1 *measure* is the harmonic mean of precision and recall

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}.$$

4.2 Overview of the Algorithm

The flowchart of LSQ is presented in Figure 1 and its pseudocode in Algorithm 1. In the beginning, the user is asked to provide a few positive and a few negative examples. The hypothesis considered by the algorithm is set to an empty hypothesis. The hypothesis is then refined, as described in Section 4.5. Then, the algorithm generates a few examples that follow the hypothesis and a few examples that contradict the hypothesis, and the user is asked to assign them to one of the two sets. If any of the examples following the hypothesis is assigned to the set `N`, it means that the hypothesis is invalid and the last refinement is retracted. If the hypothesis is good enough w.r.t. the known examples (c.f. Section 4.3), it is presented to the user along with the full result of posing its corresponding query to the SPARQL endpoint. The user then must either accept the hypothesis or add at least one new positive or negative example, e.g. by selecting an IRI from the result which should not be there, or by adding an IRI which is missing. If the hypothesis is not good enough or the user adds a new example, the algorithm goes back to generating a new refinement.

4.3 Measuring Quality of a Hypothesis

To compute with a SPARQL endpoint how good a hypothesis is, the query presented in Listing 1 is used. Such a query computes the measures described above. The variable `?s` (resp. `?t`) is mapped to all IRIs from the set `P` (resp. `N`) that follow the hypothesis, thus `?tp` corresponds to TP, `?fp` to FP and so on. If the value of F_1 measure is high enough, the hypothesis is *good enough* w.r.t. to the examples available to the algorithm, and it is presented to the user as the final answer.

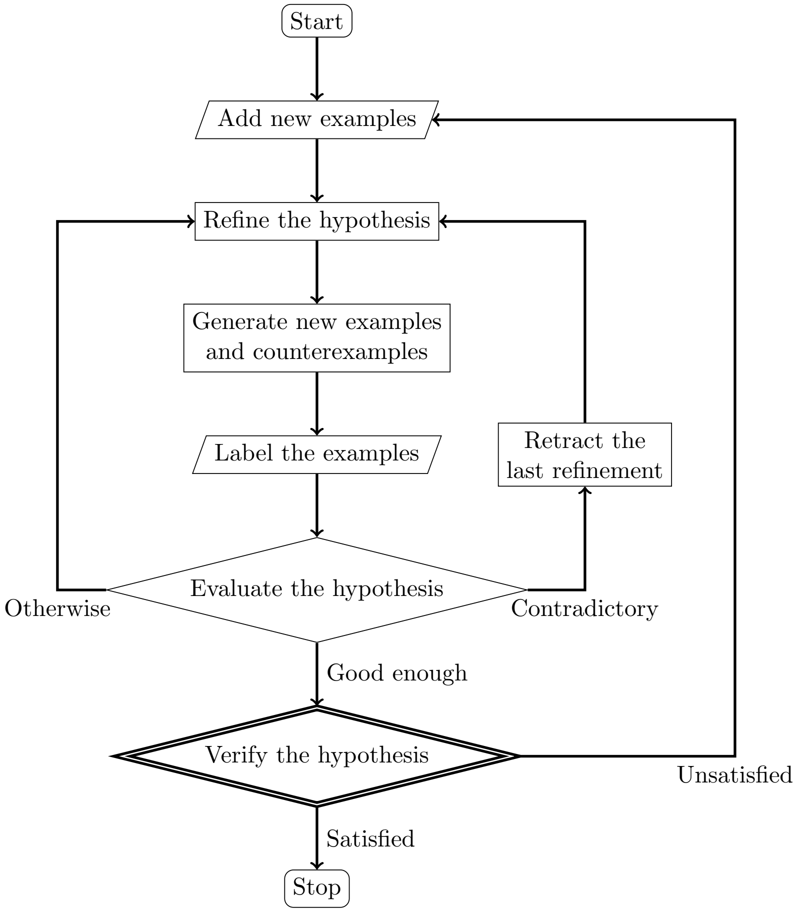


Figure 1. The flowchart of LSQ. A trapezoid denotes an input from the user, a rectangle denotes computations, a single-edged rhombus denotes a decision made by the algorithm and a double-edged rhombus denotes a decision made by the user.

4.4 Generating New Examples and Querying the User

If the hypothesis is not good enough, the algorithm should gather more evidence, to either confirm it or to reject it. To do so, the algorithm must generate a small set of examples that follow the hypothesis and another small set of examples that contradicts the hypothesis. To generate examples following the hypothesis, the query presented in Listing 2 is used. `H(?iri)` in the query ensures that an example follows the hypothesis, and `FILTER` ensures that it is a new example, i.e. one that is not present in either of the sets `P` and `N`. `LIMIT 3` is to ensure that we query the user only about a small number of new examples. Recall the example

```
$h \leftarrow$ empty hypothesis;
$fn \leftarrow 0$;
while $h$ is not good enough do
    if $fn > 0$ then h.pop_back();
    $ready\_to\_query \leftarrow false$;
    while not $ready\_to\_query$ do
        while $|h| > 0$ do
            $p, n \leftarrow$ generate new examples;
            if $|p| > 0$ and $|n| > 0$ then break;
            else h.pop_back();
        end
        $cand \leftarrow \emptyset$;
        foreach $var \in variables(h)$ do
            $cand \leftarrow cand \cup$ refinements for variable $var$ (cf. Section 4.5);
        end
        sort $cand$ according to the value of precision;
        foreach $c \in cand$ do
            h.push_back(c);
            $p, n \leftarrow$ generate new examples;
            if $h$ is good enough or $(|p| > 0$ and $|n| > 0)$ then
                $ready\_to\_query \leftarrow true$;
                break;
            end
            h.pop_back();
        end
    end
    ask the user to label examples in $p$ and $n$;
    $fn \leftarrow$ number of examples from $p$ that user labeled as negative;
end
```

Algorithm 1: The pseudo-code of the LSQ algorithm. Deciding whether the hypothesis is good enough is described in details in Section 4.3 and generating new examples in Section 4.4. Function pop_back removes the last element of the array, while push_back extends the array with its argument.

and let `H(?iri)` be `dbr:European_Union dbo:wikiPageWikiLink ?iri.` Possible new examples are `dbr:Above_mean_sea_level`, `dbr:Afroasiatic_languages`, `dbr:Andorra`, neither of them following the criteria we aim for since the hypothesis is too broad.

Generating negative examples is more difficult. Let `Hp(?iri)` be a hypothesis the hypothesis `H(?iri)` directly originated from, i.e. `Hp(?iri)` is the hypothesis `H(?iri)` without the last refinement. Consider the SPARQL query presented in Listing 3. The only common variable between `Hp(?iri)` and `H(?iri)` is `?iri`, the

```
SELECT (COUNT(DISTINCT ?s) as ?tp)
    (COUNT(DISTINCT ?t) AS ?fp)
    (?tp/(?tp+?fp) AS ?precision)
    (?tp/n_pos AS ?recall)
    (2/((1/?precision)+(1/?recall)) AS ?f1)
WHERE {{
        U(?s)
        H(?s)
        VALUES ?s { P }
    } UNION {
        U(?t)
        H(?t)
        VALUES ?t { T }
    }}
```

Listing 1. A query used to compute how good the hypothesis `H(?s)`

```
SELECT DISTINCT ?iri
WHERE {
    U(?iri)
    H(?iri)
    FILTER(?iri NOT IN (P N))
} LIMIT 3
```

Listing 2. A query used to generate new examples following the hypothesis `H(?iri)`

rest is uniquely renamed. The renaming is such that no variable except `?iri` is shared with the user-provided BGP `U(?iri)`. This query provides a set of examples that follow the hypothesis except for the last refinement. `FILTER` and `LIMIT` are used for the same purpose as before. The process of generating new examples follows the idea of active learning, where a learning algorithm selects unknown examples to

```
SELECT DISTINCT ?iri
WHERE {
    U(?iri) {
        { Hp(?iri) }
        MINUS
        { H(?iri) }
    } FILTER(?iri NOT IN (P N))
} LIMIT 3
```

Listing 3. A query used to generate new negative examples following the hypothesis `H(?iri)`, where `Hp(?iri)` is the previous hypothesis

```
SELECT ?p ?o (COUNT(DISTINCT ?s) AS ?tp)
    (COUNT(DISTINCT ?t) AS ?fp)
    (?tp/(?tp+?fp) AS ?precision)
    (?tp/n_pos AS ?recall)
    (2/((1/?precision)+(1/?recall)) AS ?f1)
WHERE {{
        U(?s)
        H(?s)
        ?s ?p ?o .
        VALUES ?s { P }
    } UNION {
        U(?t)
        H(?t)
        ?t ?p ?o .
        VALUES ?t { N }
    }}
GROUP BY ?p ?o
HAVING (?recall>=.99)
```

Listing 4. A query used to compute the set of possible refinements of the hypothesis `H(?iri)`, that have a fixed predicate and object

maximize benefit from getting the correct labels for them, i.e., to remove as much uncertainty as possible [8].

After the examples are generated, they are presented to the user. He/she must then assign each of them either to the set `P` or `N`. After the assignment is done, the algorithm continues, as described in Section 4.2.

4.5 Hypothesis Refinement

If the hypothesis is not good enough, the algorithm must refine it. First, the algorithm checks whether it is possible to generate new examples using the current hypothesis. If it is not, the most recent refinement of the hypothesis is retracted and the condition is checked again. The process continues until it becomes possible to generate new examples. In the worst case, it means emptying the hypothesis.

To refine the hypothesis, the algorithm must generate a set of possible refinements and select the best of them. First, consider a refinement consisting of a single triple pattern with a fixed predicate and object, and a subject being a variable already present in the hypothesis. For example, such a refinement could be `?iri dct:subject dbr:Category:Capitals_in_Europe`. To generate a set of such refinements the query presented in Listing 4 is used. Observe that the results are grouped by the pair `?p ?o`, so effectively what this query does is to compute the measures for a lot of hypotheses at once, each consisting of the original hypothesis H and a refinement with the subject being a variable already present in the hypothesis (`?s` and `?t`) and fixed values for the predicate and the object. We require for the recall to reach at least 0.99, to ensure that all known positive examples are covered by a new hypothesis.

Recall the example, and let `H(?iri)` be

```
dbr:European_Union dbo:wikiPageWikiLink ?iri.
```

Consider the refinement

```
?iri dct:subject dbr:Category:Capitals_in_Europe.
```

Its recall is $r = 1$, as it matches all five elements of the set `P`, and its precision is $p = 1$ as it does not match the negative example, i.e., Oslo. On the other hand, the refinement `?iri dbo:utcOffset "+2"` will not be considered, as its recall is 0.8 (`dbr:Berlin` does not occur in such a triple).

To compute refinements with a fixed subject, but a variable object, the algorithm uses the same query, but replaces `?s ?p ?o` (resp. `?t ?p ?o`) with `?o ?p ?s` (resp. `?o ?p ?t`). Finally, to compute refinements with a fixed predicate, a new variable as the object (resp. subject) and an existing variable as the subject (resp. object), the query with `?o` replaced with a blank node `[]` in the triple patterns and without `?o` in the head and in the GROUP BY clause is used.

A more elaborate query, presented in Listing 5, is required to provide refinements with FILTER. The query operates in two steps. First, the subquery extracts all the pairs consisting of a predicate `?p` and a literal `?l` for the set `P`. Then, for every pair `?p ?l` it computes the measures for a hypothesis consisting of the original hypothesis `H(?s)`, a triple pattern `?s ?p ?xl` (`?xl` is a new variable), and a filter comparing the new variable `?xl` to a literal `?l` from the subquery. Again, we require the recall to be at least 0.99 to ensure appropriate coverage of positive examples. A sample refinement obtained this way is

```
?iri dbo:populationTotal ?var.
FILTER(?var >= "205934"^^ xsd:nonNegativeInteger).
```

When the set of the possible refinements is collected from the endpoint, the algorithm must choose the right one. The set of refinements is sorted according to the descending values of the F_1 measure and (in case of ties on F_1) the precision. For every refinement, the algorithm checks if the current hypothesis with the refinement added is good enough. If it is, the hypothesis is displayed to the user, as described in Section 4.2. Otherwise, the algorithm tries to generate new positive and negative examples (c.f. Section 4.4). If it is not possible, the refinement is retracted from the hypothesis and the refinement next in order is checked. If the examples were generated, the algorithm proceeds as described earlier. If the algorithm fails to find any suitable refinement, it terminates with a failure.

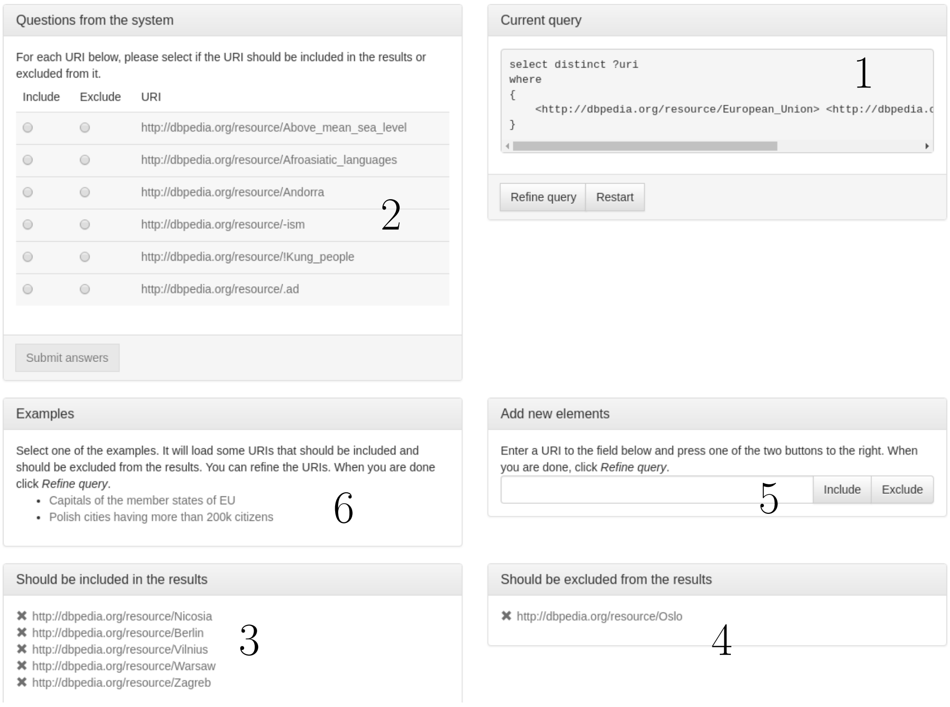
```
SELECT ?p ?l (COUNT(DISTINCT ?s) AS ?tp)
    (COUNT(DISTINCT ?t) as ?fp)
    (?tp/(?tp+?fp) AS ?precision)
    (?tp/n_pos AS ?recall)
    (2/((1/?precision)+(1/?recall)) AS ?f1)
WHERE {{
        U(?s)
        H(?s)
        ?s ?p ?xl.
        FILTER(isLiteral(?xl))
        VALUES ?s { P }
    } UNION {
        U(?t)
        H(?t)
        ?t ?p ?xl.
        FILTER(isLiteral(?xl))
        VALUES ?t { N }
    } FILTER(?xl <= ?l) {
        SELECT DISTINCT ?p ?l
        WHERE {
            U(?s)
            H(?s)
            ?s ?p ?l.
            FILTER(isLiteral(?l))
            VALUES ?s { P }
        }}}
GROUP BY ?p ?l
HAVING (?recall >= .99)
```

Listing 5. A query used to compute the set of possible refinements of the hypothesis `H(?iri)`, that use a FILTER clause

5 IMPLEMENTATION

To make it easier to reuse the algorithm, we provide a *Python* implementation of the algorithm available at: `https://semantic.cs.put.poznan.pl/ltq/src.tgz`. We also developed an on-line interface to the implementation which we coupled with *Blazegraph 2.1.1*[3] loaded with *DBpedia 2015-04* and made publicly accessible at `https://semantic.cs.put.poznan.pl/ltq/`. In the implementation, we assume a hypothesis to be good enough if the F_1 measure is at least 0.99. Screenshots of the interface are presented in Figures 2 and 3 and the detailed description is presented in [25].

[3] `https://www.blazegraph.com/`


Questions from the system

For each URI below, please select if the URI should be included in the results or excluded from it.

Include	Exclude	URI
○	○	http://dbpedia.org/resource/Above_mean_sea_level
○	○	http://dbpedia.org/resource/Afroasiatic_languages
○	○	http://dbpedia.org/resource/Andorra
○	○	http://dbpedia.org/resource/-ism
○	○	http://dbpedia.org/resource/!Kung_people
○	○	http://dbpedia.org/resource/.ad

Submit answers

Current query

```
select distinct ?uri
where
{
    <http://dbpedia.org/resource/European_Union> <http://dbpedia.c
}
```

Refine query Restart

Examples

Select one of the examples. It will load some URIs that should be included and should be excluded from the results. You can refine the URIs. When you are done click *Refine query*.

- Capitals of the member states of EU
- Polish cities having more than 200k citizens

Add new elements

Enter a URI to the field below and press one of the two buttons to the right. When you are done, click *Refine query*.

Include Exclude

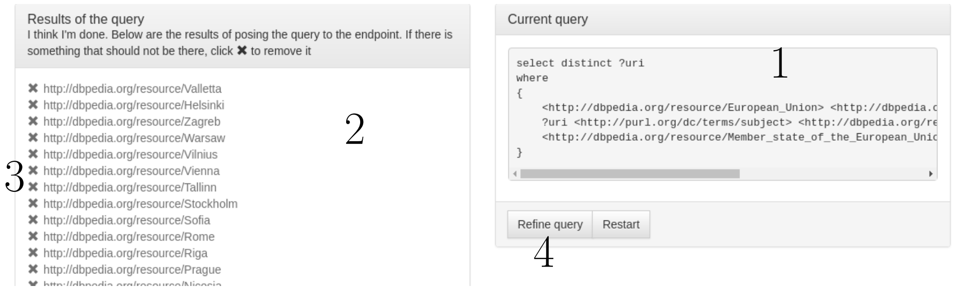
Should be included in the results

✖ http://dbpedia.org/resource/Nicosia
✖ http://dbpedia.org/resource/Berlin
✖ http://dbpedia.org/resource/Vilnius
✖ http://dbpedia.org/resource/Warsaw
✖ http://dbpedia.org/resource/Zagreb

Should be excluded from the results

✖ http://dbpedia.org/resource/Oslo

Figure 2. A screenshot of the on-line interface to the implementation of the algorithm. It presents 1) the query corresponding to a current hypothesis, 2) a set of new examples the user is supposed to assign to one of the two sets and already known 3) positive, and 4) negative examples. It is also possible to 5) add a new example by entering its IRI or 6) load one of the demo scenarios.


Results of the query
I think I'm done. Below are the results of posing the query to the endpoint. If there is something that should not be there, click ✖ to remove it

✖ http://dbpedia.org/resource/Valletta
✖ http://dbpedia.org/resource/Helsinki
✖ http://dbpedia.org/resource/Zagreb
✖ http://dbpedia.org/resource/Warsaw
✖ http://dbpedia.org/resource/Vilnius
✖ http://dbpedia.org/resource/Vienna
✖ http://dbpedia.org/resource/Tallinn
✖ http://dbpedia.org/resource/Stockholm
✖ http://dbpedia.org/resource/Sofia
✖ http://dbpedia.org/resource/Rome
✖ http://dbpedia.org/resource/Riga
✖ http://dbpedia.org/resource/Prague
✖ http://dbpedia.org/resource/Nicosia

Current query

```
select distinct ?uri
where
{
    <http://dbpedia.org/resource/European_Union> <http://dbpedia.c
    ?uri <http://purl.org/dc/terms/subject> <http://dbpedia.org/re
    <http://dbpedia.org/resource/Member_state_of_the_European_Unic
}
```

Refine query Restart

Figure 3. After a good enough hypothesis is reached, the interface displays 1) the query corresponding to the hypothesis and 2) results of posing it to the SPARQL endpoint. The user can then use 3) the X buttons to remove some of the unwanted results and further 4) refine the query.

6 EXPERIMENTAL EVALUATION

6.1 Using Query Logs to Simulate the Users

To validate the algorithm, we posed the following research question: is the algorithm able to cover the requirements of the real users of DBpedia? To answer the question, we collected a set of SPARQL queries from the logs of the official DBpedia SPARQL endpoint and used the queries as a gold standard. The queries and the raw results of the experiment are available with the source code. Below, we describe the details of the experiment.

6.1.1 Setup

The *Linked SPARQL Queries* dataset contains queries obtained from query logs of SPARQL endpoints for various popular Web of Data datasets [26]. Among others, it contains queries from the official DBpedia SPARQL endpoint, for the period from 30. 04. 2010 to 20. 07. 2010. From this set, we selected queries that are SELECT queries and contain only a single variable in the head or use star in the head, but contain only a single variable. This way we obtained 415 342 queries (415 145 character-wise distinct queries). We then randomly selected 50 queries, which did not fail when posed to a DBpedia SPARQL endpoint and resulted in at least 20 different IRIs. The endpoint run on *Blazegraph 2.1.1* and contained DBpedia 2015-04. For the selected 50 queries, the smallest number of different IRIs retrieved was 20, and the largest 2 060 507.

We used each of the selected queries to simulate a user. Each query was posed to the SPARQL endpoint and its results (any duplicates removed) were used as a gold standard, i.e. the set of all the IRIs the simulated user is interested in. Out of the gold standard, we randomly selected 5 IRIs, which were given to the algorithm as the initial positive examples. To provide the negative examples we randomly selected the following six IRIs: `dbr:Bydgoszcz`, `dbr:Murmur_%28record_label%29`, `dbr:Julius_Caesar`, `dbo:abstract`, `dbp:after`, `dbo:Agent`. If any of these negative examples was in the gold standard, it was removed from the set of negative examples. We used an empty user-provided BGP.

We then run the algorithm until it converged to a hypothesis that resulted in exactly the same set of IRIs as the gold standard, or for 10 good enough hypotheses generated by the algorithm, whichever came first. If a good enough hypothesis was not perfect, we randomly added up to 5 new positive examples (i.e., new IRIs which were present in the gold standard, but were not present in the results of the hypothesis) and up to 5 new negative examples (i.e., new IRIs that were present in the results of the hypothesis, but were not present in the gold standard). We also counted the number of interactions of the simulated user with the algorithm and measured the wall time.

6.1.2 Results

For 41 of the selected queries the first good enough hypothesis presented to the simulated user was perfect, for 8 the second, and for 1 only the third hypothesis was perfect. The WHERE clause of the gold standard query corresponding to this last case was `?value dbp:subdivisionType dbr:List_of_counties_in_Montana`. The first hypothesis was too broad, consisting of a single triple pattern

```
?iri dbp:areaCode "406"^^xsd:integer.
```

Apparently, this is a correct telephone area code for the state of Montana [31], but so it is for Gümüşhane Province in Turkey [30]. The second try was, on the other hand, too narrow:

```
?iri dbp:subdivisionType
    dbr:Political_divisions_of_the_United_States .
?iri dbp:subdivisionType
    dbr:List_of_counties_in_Montana .
```

It omitted four resources from the gold standard: `dbr:Browning,_Montana`, `dbr:Missoula,_Montana`, `dbr:Great_Falls,_Montana`, `dbr:Helena,_Montana`. After they were added as positive examples, the algorithm converged to the correct hypothesis.

For 38 of the queries, the simulated user was asked only once to label a set of six examples, in case of 4 queries the user was asked twice, for 3 queries thrice, for 2 queries four times, for another 2 queries five times and once six times. The user waited on average for 56 ± 22 seconds (median: 41 seconds) during the whole process for the algorithm to generate new examples or present a good enough hypothesis. In the case of queries that achieved the gold standard with the first good enough hypothesis, the average time was 50 ± 2 seconds. For 82 % of the queries the user obtained a perfect hypothesis in less than a minute and was asked to label at most 12 examples. That means the algorithm is really able to cover requirements of the users, it does not waste their time in waiting and we can answer positively to the research question.

6.2 Using the Algorithm to Solve Classification Problems

To prove that LSQ is not a DBpedia-specific algorithm, we used *SML-Bench*, a benchmarking framework for structured machine learning [29]. SML-Bench provides 9 datasets of varying complexity and size, and integrates a sizable number of learning systems. The authors of SML-Bench performed an extensive evaluation with 8 datasets and 15 configurations of the learning systems. We performed a similar experiment with LSQ and report the details below, comparing to the results reported by the authors of SML-Bench.

6.2.1 Setup

In order to perform the experiment, we had to prepare a wrapper adapting a learning problem to our interactive approach. SML-Bench provides the wrapper with a set of positive and negative learning examples, and an OWL file with the background knowledge. The wrapper then runs an instance of Blazegraph, loads the provided OWL file and initializes LSQ by randomly selecting 10 of the positive and 10 of the negative learning examples and providing them as the initial examples to LSQ. Next, LSQ is executed until it does not converge to a good enough hypothesis or until the prescribed amount of time is exceeded, whichever comes first. Every time LSQ generates a set of examples to be labeled by the user, the current hypothesis is stored and the examples are labeled as follows: if an example is present in the set of positive learning examples, it is labeled as positive, otherwise it is labeled as negative. If the maximal execution time is to be exceeded, all the stored hypotheses are reevaluated on the set of learning examples gathered so far by the algorithm and the one with the highest score is selected as the final result. We assume that the user-provided BGP is empty.

We used the SML-Bench framework to perform the experiment using the same parameters as Westphal et al. [29]: 8 datasets; 5 minutes for a single execution of the algorithm, including loading the data and the final reevaluating of the hypotheses (enforced by the framework); 10-fold cross-validation. To execute the experiment we used a workstation with *Intel Core i7-3770* CPU with 4 cores (8 threads) clocked at 3.40 GHz and equipped with 16 GB of RAM. It must be noted that this is a considerably weaker configuration than the one used by Westphal et al. and thus the limit of 5 minutes was, in fact, more strict.

6.2.2 Results

Following Westphal et al. we report the accuracy and the F_1 score in, respectively, Table 2 and Table 3. The measures were averaged over all the folds of the cross-validation. For comparison, we report the four best systems from the experiment by Westphal et al.: Aleph[4] and DLLearner [18] in three configurations. It must be noted that there was no clear winner in the experiment: different systems were the best on different learning problems.

In one case (the learning problem nctrer/1) LSQ achieved results far better than the remaining systems. In all the folds of the cross-validation LSQ generated exactly the same hypothesis consisting of a single triple pattern:

```
?s <http://dl-learner.org/ont/ActivityOutcome_NCTRER>
    "active"^^xsd:string .
```

Our suspicion is that the true label is encoded in the background knowledge, and thus the obtained result is not credible.

[4] `http://www.cs.ox.ac.uk/activities/machinelearning/Aleph/aleph.html`

For the remaining 8 datasets, we used Welch's unequal variances t-test [28] to compare the average accuracy values between the values achieved by LSQ and the highest average accuracy of the remaining systems. The null hypothesis was that both averages are equal. In two cases (pyrimidine/1 and carcinog./1) it was not possible to reject the null hypothesis (p-values, respectively, 0.234 and 0.263), in the five remaining cases the null hypothesis was rejected (p-value below 0.01).

Comparing LSQ with the best algorithm is unfavourable for LSQ, because its hypothesis space is limited to a single SPARQL BGP, and is considerably smaller those of Aleph, DLLearner (CELOE) and DLLearner (OCEL). We thus performed additional analysis comparing LSQ with DLLearner (ELTL). ELTL stands for the *EL Tree Learner* algorithm, that has the target language restricted to OWL EL, similar in its expressivity to the expressivity of SPARQL BGPs.

In the case of four learning problems (carcinog./1, hepatitis/1, lymphogr./1, mutag./42), it was not possible to reject the null hypothesis (p-values, respectively, 0.263, 1.0, 0.071, 0.753). Comparison for the learning problem pyrimidine/1 was not possible due to the missing results for ELTL, and for the learning problems mammogr./1 and prem.leag./1 we rejected the null hypotheses (p-values below 10^{-4}). In both cases we observe that the average accuracy of LSQ is higher than this of ELTL.

From the performed comparison we conclude that the performance of LSQ is at least as good as the performance of ELTL. Moreover, we observe that in some cases LSQ is on a par with the best of the remaining algorithms.

Learning Problem	LSQ	Aleph	DLLearner (CELOE)	DLLearner (OCEL)	DLLearner (ELTL)
carcinog./1	0.53 ± 0.05	0.48 ± 0.10	0.55 ± 0.02	no results	0.55 ± 0.02
hepatitis/1	0.43 ± 0.03	0.67 ± 0.05	0.47 ± 0.05	0.66 ± 0.14	0.41 ± 0.01
lymphogr./1	0.54 ± 0.03	0.83 ± 0.10	0.83 ± 0.11	0.73 ± 0.12	0.54 ± 0.03
mammogr./1	0.52 ± 0.03	0.65 ± 0.04	0.49 ± 0.02	0.82 ± 0.05	0.46 ± 0.01
mutag./42	0.31 ± 0.07	0.72 ± 0.25	0.94 ± 0.13	0.53 ± 0.29	0.30 ± 0.07
nctrer/1	1.00 ± 0.00	0.72 ± 0.14	0.59 ± 0.02	0.81 ± 0.09	0.58 ± 0.02
prem.leag./1	0.89 ± 0.09	0.95 ± 0.09	0.99 ± 0.04	0.85 ± 0.10	0.49 ± 0.02
pyrimidine/1	0.85 ± 0.20	0.95 ± 0.16	0.83 ± 0.17	0.85 ± 0.24	no results

Table 2. The average accuracy and its standard deviation over the 10-folds cross-validation for LSQ, and for the other systems, their values reported from [29]

7 CONCLUSIONS

We presented an algorithm for learning SPARQL queries using examples provided by the user. The algorithm uses active learning to minimize the required number of learning examples. It is also suitable for any RDF graph accessible through a SPARQL endpoint and does not require any preprocessing or initialization phase. By using aggregates, grouping and inline values from SPARQL 1.1, the algorithm is able to

Learning problem	Learning SPARQL Queries	Aleph	DLLearner (CELOE)	DLLearner (OCEL)	DLLearner (ELTL)
carcinog./1	0.69 ± 0.04	0.46 ± 0.12	0.71 ± 0.01	no results	0.71 ± 0.01
hepatitis/1	0.59 ± 0.01	0.38 ± 0.12	0.60 ± 0.02	0.64 ± 0.07	0.58 ± 0.01
lymphogr./1	0.70 ± 0.03	0.84 ± 0.09	0.87 ± 0.07	0.76 ± 0.10	0.70 ± 0.03
mammogr./1	0.64 ± 0.02	0.48 ± 0.08	0.64 ± 0.01	0.78 ± 0.08	0.63 ± 0.00
mutag./42	0.47 ± 0.08	0.43 ± 0.47	0.93 ± 0.14	0.29 ± 0.42	0.46 ± 0.08
nctrer/1	1.00 ± 0.00	0.71 ± 0.18	0.73 ± 0.02	0.85 ± 0.06	0.73 ± 0.02
prem.leag./1	0.86 ± 0.12	0.94 ± 0.11	0.99 ± 0.04	0.97 ± 0.06	0.66 ± 0.02
pyrimidine/1	0.81 ± 0.30	0.90 ± 0.32	0.84 ± 0.15	0.80 ± 0.13	no results

Table 3. The average F_1 score and its standard deviation over the 10-folds cross-validation for LSQ, and the other algorithms, their values reported from [29]

move most of the complex computations to the SPARQL endpoint. Contemporary RDF stores (e.g. *Blazegraph*) are very sophisticated and are able to deal with such queries without any problem. To prove that the algorithm works, we provide an online interface for testing, available at `https://semantic.cs.put.poznan.pl/ltq/`. To prove the usability of the algorithm, we performed two experiments: one using real queries from the *Linked SPARQL Queries* and DBpedia; the other using 8 datasets from the structured machine learning benchmark SML-Bench. In the first case, we showed that the algorithm is able to converge to the gold standard with only a minimal amount of interaction with the user. In the second case, the algorithms performance was similar to those algorithms with similarly restricted hypothesis space.

In the future, we would like to analyze whether using a different measure may provide even faster convergence to a correct hypothesis. We would also like to extend the algorithm with a possibility of obtaining some prior knowledge from the user.

Acknowledgements

Jedrzej Potoniec acknowledges the support of the Polish National Science Center, grant DEC-2013/11/N/ST6/03065 and the support of grant 09/91/DSMK/0659.

REFERENCES

[1] Auer, S.—Bizer, C.—Kobilarov, G.—Lehmann, J.—Cyganiak, R.—Ives, Z. G.: DBpedia: A Nucleus for a Web of Open Data. In: Aberer, K. et al. (Eds.): The Semantic Web (ISWC 2007, ASWC 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4825, 2007, pp. 722–735, doi: 10.1007/978-3-540-76298-0_52.

[2] Baeza-Yates, R.—Ribeiro-Neto, B.: Modern Information Retrieval. 2nd ed. Addison-Wesley Professional, 2011.

[3] Bizer, C.—Lehmann, J.—Kobilarov, G.—Auer, S.—Becker, C.—Cyganiak, R.—Hellmann, S.: DBpedia – A Crystallization Point for the Web of Data. Journal of Web Semantics, Vol. 7, 2009, No. 3, pp. 154–165, doi: 10.1016/j.websem.2009.07.002.

[4] Bottoni, P.—Ceriani, M.: SPARQL Playground: A Block Programming Tool to Experiment with SPARQL. In: Ivanova, V., Lambrix, P., Lohmann, S., Pesquita, C. (Eds.): Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA! 2015). CEUR Workshop Proceedings, Vol. 1456, 2015, pp. 103–108.

[5] Campinas, S.: Live SPARQL Auto-Completion. In: Horridge, M., Rospocher, M., van Ossenbruggen, J. (Eds.): Proceedings of the ISWC 2014 Posters and Demonstrations Track, a Track within the 13th International Semantic Web Conference (ISWC 2014). CEUR Workshop Proceedings, 2014, Vol. 1272, pp. 477–480.

[6] Campinas, S.—Perry, T. E.—Ceccarelli, D.—Delbru, R.—Tummarello, G.: Introducing RDF Graph Summary with Application to Assisted SPARQL Formulation. In: Hameurlain, Q., Min Tjoa, A., Wagner, R. (Eds.): 23rd International Workshop on Database and Expert Systems Applications (DEXA 2012). IEEE Computer Society, 2012, pp. 261–266, doi: 10.1109/dexa.2012.38.

[7] Carothers, G.—Prud'hommeaux, E.: RDF 1.1 Turtle. W3C Recommendation, W3C, February 2014, `http://www.w3.org/TR/2014/REC-turtle-20140225/`.

[8] Cohn, D.: Active Learning. In: Sammut, C., Webb, G. I. (Eds.): Encyclopedia of Machine Learning. Springer US, Boston, MA, 2010, pp. 10–14, doi: 10.1007/978-0-387-30164-8_6.

[9] Sana e Zainab, S.—Hasnain, A.—Saleem, M.—Mehmood, Q.—Zehra, D.—Decker, S.: FedViz: A Visual Interface for SPARQL Queries Formulation and Execution. In: Ivanova, V., Lambrix, P., Lohmann, S., Pesquita, C. (Eds.): Proceedings of the International Workshop on Visualizations and User Interfaces for Ontologies and Linked Data (VOILA! 2015). CEUR Workshop Proceedings, Vol. 1456, 2015, pp. 49–60.

[10] Fanizzi, N.—d'Amato, C.—Esposito, F.: DL-FOIL Concept Learning in Description Logics. In: Železný, F., Lavrač, N. (Eds.): Inductive Logic Programming (ILP 2008). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 5194, 2008, pp. 107–121, doi: 10.1007/978-3-540-85928-4_12.

[11] Ferré, S.: Sparklis: An Expressive Query Builder for SPARQL Endpoints with Guidance in Natural Language. Semantic Web, Vol. 8, 2017, No. 3, pp. 405–418, doi: 10.3233/sw-150208.

[12] Gennari, J. H.—Musen, M. A.—Fergerson, R. W.—Grosso, W. E.—Crubézy, M.—Eriksson, H.—Noy, N. F.—Tu, S. W.: The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. International Journal of Human-Computer Studies, Vol. 58, 2003, No. 1, pp. 89–123, doi: 10.1016/s1071-5819(02)00127-1.

[13] Gombos, G.—Kiss, A.: SPARQL Query Writing with Recommendations Based on Datasets. In: Yamamoto, S. (Ed.): Human Interface and the Management of Information. Information and Knowledge Design and Evaluation (HIMI 2014). Pro-

ceedings, Part I. Springer, Cham, Lecture Notes in Computer Science, Vol. 8521, 2014, pp. 310–319, doi: 10.1007/978-3-319-07731-4_32.

[14] Harris, S.—Seaborne, A.: SPARQL 1.1 Query Language. W3C Recommendation, W3C, March 2013, `http://www.w3.org/TR/2013/REC-sparql11-query-20130321/`.

[15] Höffner, K.—Lehmann, J.: Towards Question Answering on Statistical Linked Data. In: Sack, H., Filipowska, A., Lehmann, J., Hellmann, S. (Eds.): Proceedings of the 10th International Conference on Semantic Systems (SEMANTICS 2014). ACM, 2014, pp. 61–64, doi: 10.1145/2660517.2660521.

[16] Höfner, K.—Walter, S.—Marx, E.—Usbeck, R.—Lehmann, J.—Ngonga Ngomo, A.-C.: Survey on Challenges of Question Answering in the Semantic Web. Semantic Web, Vol. 8, 2017, No. 6, pp. 895–920, doi: 10.3233/sw-160247.

[17] Ławrynowicz, A.—Potoniec, J.: Pattern Based Feature Construction in Semantic Data Mining. International Journal on Semantic Web and Information Systems (IJSWIS), Vol. 10, 2014, No. 1, pp. 27–65, doi: 10.4018/ijswis.2014010102.

[18] Lehmann, J.: DL-Learner: Learning Concepts in Description Logics. Journal of Machine Learning Research, Vol. 10, 2009, pp. 2639–2642.

[19] Lehmann, J.—Bühmann, L.: AutoSPARQL: Let Users Query Your Knowledge Base. In: Antoniou, G., Grobelnik, M. et al. (Eds.): The Semantic Web: Research and Applications (ESWC 2011). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6643, 2011, pp. 63–79, doi: 10.1007/978-3-642-21034-1_5.

[20] Lehmann, J.—Isele, R.—Jakob, M.—Jentzsch, A.—Kontokostas, D.—Mendes, P. N.—Hellmann, S.—Morsey, M.—van Kleef, P.—Auer, S.—Bizer, C.: DBpedia – A Large-Scale, Multilingual Knowledge Base Extracted from Wikipedia. Semantic Web, Vol. 6, 2015, No. 2, pp. 167–195, doi: 10.3233/SW-140134.

[21] de Leon, A.—Wisniewki, F.—Villazón-Terrazas, B.—Corcho, O.: Map4rdf – Faceted Browser for Geospatial Datasets. PMOD Workshop, USING OPEN DATA: Policy Modeling, Citizen Empowerment, Data Journalism, 2012.

[22] Muñoz, S.—Pérez, J.—Gutierrez, C.: Minimal Deductive Systems for RDF. In: Franconi, E., Kifer, M., May, W. (Eds.): The Semantic Web: Research and Applications (ESWC 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4519, 2007, pp. 53–67, doi: 10.1007/978-3-540-72667-8_6.

[23] Oren, E.—Delbru, R.—Decker, S.: Extending Faceted Navigation for RDF Data. In: Cruz, I. F. et al. (Eds.): The Semantic Web – ISWC 2006. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4273, 2006, pp. 559–572, doi: 10.1007/11926078_40.

[24] Paulheim, H.—Fürnkranz, J.: Unsupervised Generation of Data Mining Features from Linked Open Data. In: Burdescu, D. D., Akerkar, R., Badica, C. (Eds.): 2nd International Conference on Web Intelligence, Mining and Semantics (WIMS '12). ACM, 2012, Art. No. 31, doi: 10.1145/2254129.2254168.

[25] Potoniec, J.: An On-Line Learning to Query System. In: Kawamura, T., Paulheim, H. (Eds.): Proceedings of the ISWC 2016 Posters and Demonstrations Track. CEUR Workshop Proceedings, Vol. 1690, 2016, 4 pp.

[26] Saleem, M.—Ali, M. I.—Hogan, A.—Mehmood, Q.—Ngonga Ngomo, A.-C.: LSQ: The Linked SPARQL Queries Dataset. In: Arenas, M. et al. (Eds.): The

Semantic Web – ISWC 2015. Proceedings, Part II. Springer, Cham, Lecture Notes in Computer Science, Vol. 9367, 2015, pp. 261–269, doi: 10.1007/978-3-319-25010-6_15.

[27] Stickler, P.: CBD – Concise Bounded Description. W3C Member Submission, W3C, June 2005, `https://www.w3.org/Submission/CBD/`.

[28] Welch, B. L.: The Generalization of 'Student's' Problem When Several Different Population Variances Are Involved. Biometrika, Vol. 34, 1947, No. 1-2, pp. 28–35, doi: 10.1093/biomet/34.1-2.28.

[29] Westphal, P.—Bühmann, L.—Bin, S.—Jabeen, H.—Lehmann, J.: SML-Bench – A Benchmarking Framework for Structured Machine Learning. Semantic Web, Vol. 10, 2019, No. 2, pp. 231–245, doi: 10.3233/sw-180308.

[30] Wikipedia. Gümüşhane Province – Wikipedia, the Free Encyclopedia, 2015. [Online, accessed 22-June-2016].

[31] Wikipedia. Area Code 406 – Wikipedia, the Free Encyclopedia, 2016. Online, accessed 24-June-2016.

[32] Williams, G.—Feigenbaum, L.—Clark, K.—Torres, E.: SPARQL 1.1 Protocol. W3C Recommendation, W3C, March 2013, `http://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/`.

[33] Wood, D.—Lanthaler, M.—Cyganiak, R.: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, W3C, February 2014, `http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/`.

[34] Xu, K.—Feng, Y.—Zhao, D.: Xser@QALD-4: Answering Natural Language Questions via Phrasal Semantic Parsing. In: Cappellato, L., Ferro, N., Halvey, M., Kraaij, W. (Eds.): Working Notes for CLEF 2014 Conference. CEUR Workshop Proceedings, Vol. 1180, 2014, pp. 1260–1274.

Jedrzej Potoniec is a postdoctoral researcher at Poznan University of Technology (PUT), Poland. He received his Ph.D. from PUT in 2018 for research on learning ontologies from Linked Data. His research interest concentrates on machine learning with structured data, especially with the Semantic Web data.

Computing and Informatics, Vol. 38, 2019, 701–727, doi: 10.31577/cai_2019_3_701

CHAOTIC ENCRYPTION AND PRIVILEGE BASED VISUAL SECRET SHARING MODEL FOR COLOR IMAGES

Aytekin Yildizhan

Computer Engineering Department
Hacettepe University
06800 Ankara, Turkey
e-mail: aytekinyildizhan@hacettepe.edu.tr

Nurettin Topaloglu

Computer Engineering Department
Gazi University
06500 Ankara, Turkey
e-mail: nurettin@gazi.edu.tr

Abstract. In the Privilege-based Visual Secret Sharing Model (PVSSM), each share has a unique privilege and a higher-privilege share contributes with more privilege to reveal the secret image. However, in PVSSM, when several shares with the higher priority are stacked, the secret image can be visibly displayed. This security problem is solved by applying a two-dimensional Logistic-Adjusted Sine Map (2D-LASM) to each share. This method is called Chaotic Encryption-based PVSSM. In this paper, we aim to present how Chaotic Encryption-based PVSSM is applied to color images. In order to assess the efficiency of this method, histogram analysis, data loss attack, salt-pepper noise attack, differential attack, chi-square analysis and correlation analysis tests were applied. The performance of this method has been evaluated according to NCPR, UACI, PSNR, SSIM and CQM. The proposed method achieved a good test values and showed better results compared to similar studies in literature.

Keywords: Cryptography, chaotic map, information security, visual secret sharing

Mathematics Subject Classification 2010: 94A60

1 INTRODUCTION

With the development of technology and the Internet network, digital images published and transmitted over the network have become extremely important. The transfer and storage of secret and private images such as military, intelligence, and medical images must be fast and secured. Thus, researchers have started to focus their attention on image security [1]. Traditional encryption algorithm such as Data Encryption Standard (DES) and Advanced Encryption Standard (AES) can be used to encrypt images.

However, image data is much greater than text data, so the traditional encryption algorithms require a lot of time to encrypt the image. In addition, the decrypted text must be equal to the original text, but decrypted image should not necessarily be equal to the original image [2]. Watermarking and steganography have been proposed instead of DES and AES for image encryption but these methods are suitable for single communication channel [3]. Therefore, Visual Secret Sharing Scheme (VSSS) methods could be one of the possible solutions.

Visual cryptography (VC), one of the VSSS methods, was first developed in 1994 by Naor and Shamir [4]. The approach of this system based fragmenting the key and distributing them to the different users. In key-based approach, if the key is lost or stolen, all secret information is inaccessible or revealed. In order to prevent crime, VC methods are used, especially cybercrime [5]. With this method, if any key is lost or stolen, other key pieces can be put together and generate the secret key. Also any user cannot get the secret information from any single key pieces.

In VC method, the secret image is divided into n images. These are called shares that do not contain any information about the secret image. If any r shares are stacked, the secret image is restored. The result of stacking shares in the VC is equivalent to that of the logical OR ($\vee$) operation. VC has the advantages that it eliminates complex algorithm to improve applicability and efficiency [6], and decryption can be done using the human visual system. However, if less than r shares are stacked, the content of the secret image remains hidden [7].

An increase in the number of shares causes more difficulties to reveal the secret image. However, if the number of the shares is too high, restoring the secret image will become difficult as well. To resolve this dilemma, Fang and Lin [8] presented a progressive VC (PVC) for binary image. While traditional VC needs all the necessary shares for the secret image to reveal, in PVC the secret image is gradually revealed as each share is added. Jin et al. [9] proposed PVC for color image. Fang's method [10] is about how to manage the shares more easily in PVC. On the other hand, with these methods, the shares have increased four times of the size of the original image and there is no order of privilege among the shares [8, 9, 10, 11].

In the process of revealing the secret image, it may be expected to be a privileged order among the shares in hierarchical systems such as military institutions, public institutions, and corporations. Therefore, the share with higher priority is distributed to people based on their positions or status. However, in most of the studies, this situation has not been considered. In order to overcome this problem, it is necessary to use a privilege-based sharing scheme. The privilege means the capability to reveal the secret image. A group-based weighted VC is proposed in [12]. The shares are divided into different groups and given a different weight number within the group. Lin et al. [13] performed a study to determine the size of the shares according to their weights. However, when the size of any share is small, the weight of that share is small, too. Therefore, it reveals which share is small. Li et al. [14] suggested that each of the shares on the PVC scheme has a certain privilege. In their study, the shares are divided into four groups as essential, non-essential, limitedly essential and limitedly non-essential, but there is no order of importance for each group. Hou et al. [15] proposed that the importance of the shares is arranged from the lowest to the highest order. This is called the Privilege-based Visual Secret Sharing Model (PVSSM), and it has several advantages that each share size is the same as the secret image, the restored image has a better contrast than the traditional VC scheme and each share has a unique privilege level according to their order.

In PVSSM, when several shares with the higher priority are stacked, the secret image can be revealed. To solve this problem, it would be appropriate to encrypt each share. Because of the high correlation among adjacent pixels and bulk data capacity, traditional encryption algorithms such as DES, AES, RSA, are not suitable for image encryption [16]. To prevent image information leakage, chaotic systems are suitable for image encryption. Chaotic systems are used in many areas [17]. Chaotic maps are the core of a chaos-based image encryption [18]. Moreover, permutation-diffusion mechanism can be used in chaos-based image encryption [17, 19]. Chaotic maps have an excellent character such as ergodicity, flexibility, speed, unpredictability, and high sensitivity to initial state of the system and control parameter [17, 20]. As a result, chaotic-based encryption has become popular for image encryption [19].

There are many studies on chaos-based image encryption methods in protecting and transferring digital images. The level of security in chaos-based encryption depends on the performance of the chaotic maps [17]. Chaotic maps can be classified into two groups; one-dimensional and high-dimensional. The initial states and orbits of one-dimensional chaotic maps such as Logistic, Gaussian, Sinus and Tent maps, can be estimated easily [20, 21]. Arroyo et al. [20] demonstrated that the estimation of control parameters and timing attacks applied to one-dimensional chaotic maps and some weaknesses about linearity were found. Tang and Guan [22] show that control parameters were estimated in time-lagged Logistic and Mackey-Glass map by using genetic algorithm. These weaknesses have caused many security vulnerabilities in one-dimensional chaotic map [15, 19, 20]. In high-dimensional chaotic maps, because the system and control parameters are more complex, it is more unpredictable [19, 20, 21, 22, 23]. Therefore, it would be more appropriate to encrypt the shares with a high-dimensional chaotic map. However, because of the large number of parameters in the high-dimensional chaotic map, the system requirements are more expensive and the computational complexity is high [16, 17]. At the same time, using more than one chaotic map can increase the calculation time [24]. Therefore, it is important that the high-dimensional chaotic map shows low calculation time and high degree of chaos feature [21]. A two-dimensional Logistic-adjusted Sine Map (2D-LASM) [17] is a high-dimensional chaotic map that provides these features.

In this paper, we aim to present the Chaotic Encryption-based PVSSM for color images. The secret color image is first separated into RGB channels. Then, RGB channels are transformed into binary images. PVSSM and 2D-LASM based image encryption are applied respectively to each channel. Even if the shares are captured and stacked, the secret image would not be seen. In decryption process, the shares of each channel are decrypted with 2D-LASM and stacked. Recreated RGB channels are merged. To obtain the original color image from the combined RGB channels, the pixel values of 1 are changed to 255. Finally, the secret image is revealed. Histogram analysis is applied for all R, G and B channels. In addition, for observing the proposed method's performance on different image distortions, data loss and salt-pepper noise attack are applied to the encrypted images. The results show that this method has high resistance against these attacks.

2 VISUAL CRYPTOGRAPHY

The VC was first proposed by Naor and Shamir [4], the secret image is divided into n shares. When r shares are stacked, the secret image is revealed. If $r-1$ shares stacked, the secret image cannot be visible. This is called (r,n) threshold mechanism [4, 15, 25, 26, 27, 28, 29]. In black-and-white VC scheme, black and white pixels are shared according to some rules. In $(2,2)$ VC scheme, one pixel in the secret image is divided into four pixels in the share images in Figure 1.

In Figure 1, if the pixel is black, one of six blocks is randomly selected for the first share. According to the first share's block, the other block is selected for the second share. If the pixel is white, the same rule is applied. Then the shares are stacked. Black pixel is logical 1 and white pixel is logical 0. Therefore, black pixels in the secret image are obtained full black and white pixels in the secret image are obtained half-black-and-white. The stacked image is four times as big as the original image, but aspect ratio remains the same. Although the contrast of the secret image is degraded by 50 %, human visual system can still detect the content of the secret image.

In Figure 2, a black and white secret image with the words of "123456" is decomposed into two shares and the stacked image is shown.

In some studies, the privilege of the shares is limited or does not exist [4, 7, 9, 13, 28]. If the shares have a privilege order, it will be better for hierarchical systems. This means that if a person in hierarchical systems has higher privilege such as the manager of the company, etc., then his/her share must be also a higher privilege. Thus, more privileged shares are stacked, more information will be reached or revealed.

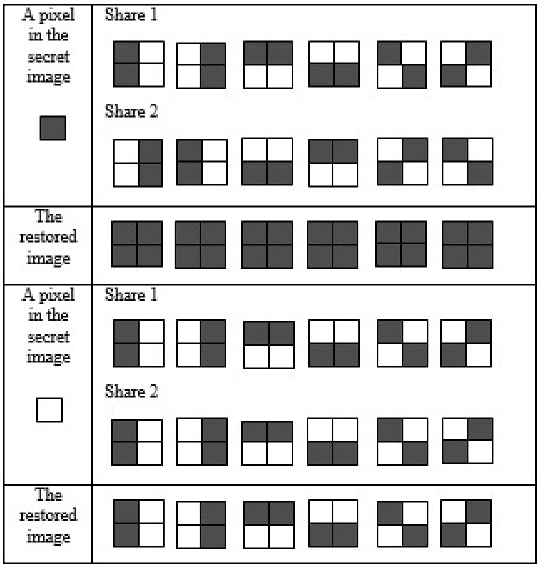
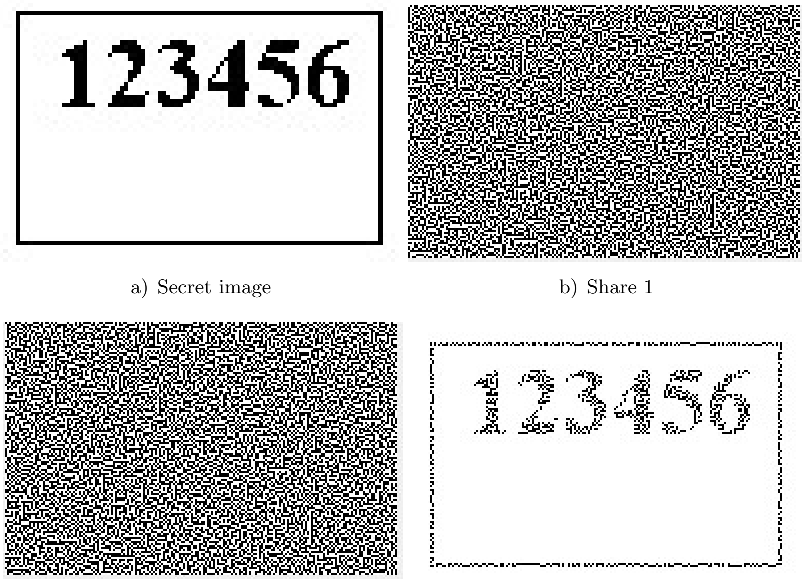


Figure 1. (2, 2) VC scheme



a) Secret image

b) Share 1

c) Share 2

d) Stacked image

Figure 2. Visual cryptography

Fang and Lin [8] first described a progressive VC on binary images. In their work, each pixel in the secret image was magnified to 2×2 blocks. If the black pixels are encrypted, 2×2 blocks are full black. If the white pixels are encrypted, one of randomly selected two pixels is white in 2×2 blocks. However, due to pixel enlargement, more memory and processing time are required.

Hou and Quan [29] reported that Fang and Lin's work [8] had pixel enlargement, unnecessary storage space and more transfer time, and also it had a low visual quality of the stacked images. Hou and Quan stated that they eliminated these disadvantages.

Fang [10] proposed a new progressive VC that stego images and shares are used together. The aim of this work is a more simplified management of shares. However, in Fang's method, the share has been increased in size by four times more than the secret image.

Hou [7] proposed a privileged-based approach. In Hou's study, to create the shares of color images, he uses three color shares (Cyan-Magenta-Yellow) and mask share. Thus, four shares are produced. The masks are randomly created to cover unwanted colors in the stacked image, while the three main colors represent the color share images. Although there is no privilege among the color shares, according to the mask, the stacked image is changed. In this case, the mask image is more privileged than the color share images. However, in this method, the size of the stacked image size has increased to four times as large as the original image and the encryption process is limited to only four shares.

Li et al. [14] posed the method that emphasizes the importance of the shares among themselves. The shares on the progressive VC scheme are classified into two groups: necessarily with higher importance and non-necessarily with lower importance. Later on, intermediate shares are also added with important and non-important shares, but it is difficult to manage the shares in this study. In addition, the size of non-important shares is larger than the size of the important ones. Because of the different size, the importance of shares is revealed.

PVSSM is proposed in [15]. With this work, each share has a unique level of privilege. The privilege level of the shares is set by the proportion of the black pixel in the secret image. This means the blacker pixels in the secret image, the higher the order of importance.

3 PRIVILEGED-BASED VISUAL SECRET SHARING MODEL

In this section, PVSSM is explained. The flow diagram of PVSSM is given in Figure 3.

The design of sharing matrices C^0 and C^1 is defined below. All rows and columns start from 1 and n is defined as the number of shares.

C^0 consists of two parts, the left-C^0 and the right-C^0. The column of left-C^0 is calculated as $m_1 = n(n-1)/2$. The size of left-C^1 is set to $n \times m_1$. In left-C^0, all 0s are placed to different columns. The number of 0s in i^{th} row is

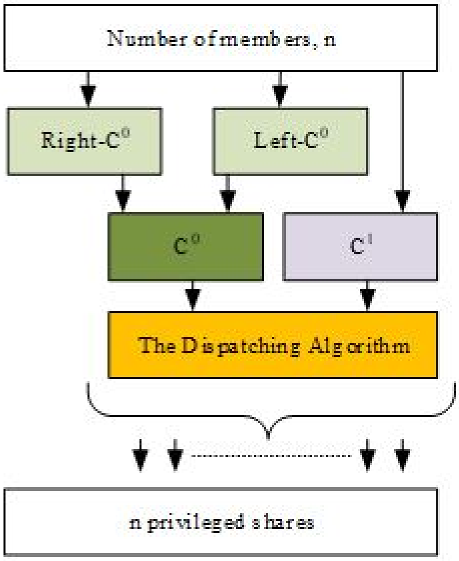


Figure 3. PVSSM flow diagram

equal to $n - i$. The remaining locations are filled with 1s. Right-C^0 and left-C^0 have the same number of rows. The column of right-C^0 is equal to $m_2 = n(n-1)(n-2)/2$. The size of right-C^0 is set to $n \times m_2$. All locations of right-C^0 are filled with 0s. After this design, right-C^0 and left-C^0 concatenate and the design of C^0 is completed.

The column of C^1 is calculated as $m = n(n-1)(n-1)/2$. Moreover, Equation (1) is satisfied:

$$m = m_1 + m_2. \tag{1}$$

The size of C^1 is set to $n \times m$. In C^1, all 1s are placed to different columns. The number of 1s in i^{th} row is calculated as $(n^2 - 3n + 2i)/2$. The remaining locations are filled with 0s. Therefore, the design of C^1 is completed.

After the design of the C^0 and C^1 sharing matrices, the dispatching algorithm is implemented to generate the shares in Figure 4.

Firstly, a random number "t", between 1 and m, is selected. If the pixel in the secret image is white (0), t^{th} column of C^0 is selected. The first element of this t^{th} column is distributed to the first share, the second element is distributed to the second share, the third element is distributed to the third share, etc. If the pixel in the secret image is black (1), t^{th} column of C^1 is selected. The first element of this t^{th} column is distributed to the first share, the second element is distributed to the second share, the third element is distributed to the third share, etc. The densities of the black pixels in the shares determine the level of its privilege in PVSSM because the human vision system more focuses on the black pixels in binary image. Output of

```
The Dispatching Algorithm:
Input: a halftone secret image SI, size of (i x j)
n, number of members
C^0 and C^1,sharing matrices
Output: n privileged shares, PS_k, k=1, 2, 3,…,n
Process:
  for a ← 1:i
      for b ← 1:j
          t ← Random Number (1 ≤ t ≤ m)
            if SI (a, b) is black
                 for z ← 1:n
                    PS_k(a, b) = C^1_{z,t}
            else
                 for z ← 1:n
                    PS_k(a, b) = C^0_{z,t}
```

Figure 4. The dispatching algorithm

this algorithm are privileged shares with increasing order and each share is a unique privilege [14].

The grayscale test image size of 256×256 pixels is used and it has produced 6 shares in PVSSM as an example (S1, S2, S3, S4, S5, S6). The grayscale test image is first converted with the Jarvis algorithm [30] into binary image in Figure 5 and it is shared by PVSSM.

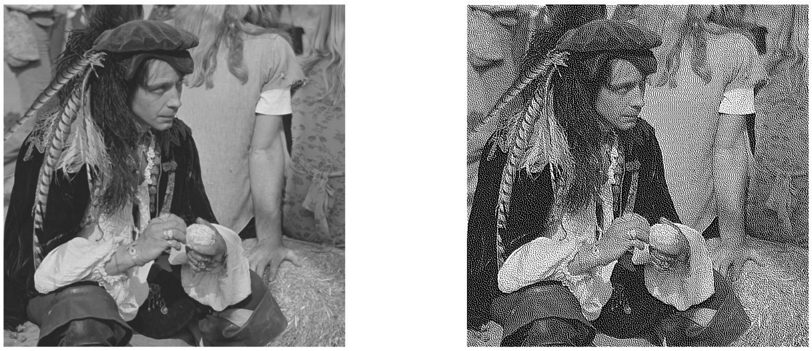
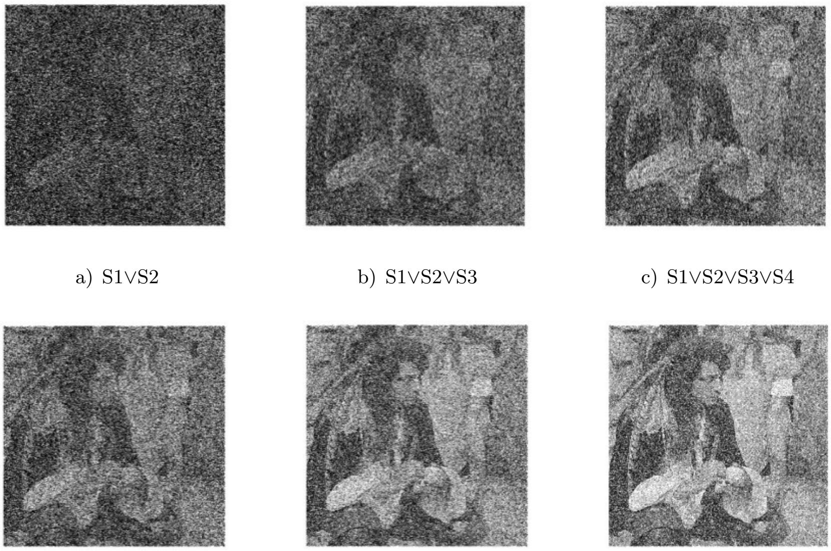

a) Grayscale image b) Binary image

Figure 5. Test image



a) S1∨S2 b) S1∨S2∨S3 c) S1∨S2∨S3∨S4

d) S1∨S2∨S4∨S5 e) S1∨S2∨S3∨S4∨S5 f) S1∨S2∨S3∨S4∨S5∨S6

Figure 6. The secret image gradually appears

As seen in Figure 6, the secret image becomes apparent while the shares are stacked one by one. The human vision system can detect the secret. Therefore, all the shares are not needed to restore the secret image. Table 1 shows the similarity ratio of the images in Figure 6 using PSNR (peak to noise signal ratio) and SSIM (structural similarity index). They have been seen to be similar both visually and mathematically.

Images	**a)**	**b)**	**c)**	**d)**	**e)**	**f)**
PSNR	52.3784	53.3662	54.5234	54.3006	55.9546	57.7802
SSIM	0.9868	0.9915	0.9944	0.9940	0.9958	0.9961

Table 1. PSNR and SSIM values

4 THE PROPOSED METHOD

Although PVSSM determines the order of importance among the shares, when the shares with the higher priority are stacked, the secret image appears visually in Figure 6. It also proves the similarities by using PSNR and SSIM in Table 1. In order to overcome this problem of PVSSM, PVSSM and 2D-LASM based image encryption are used together. The proposed method is called Chaotic Encryption-based PVSSM and we have explained how this method is applied to color images.

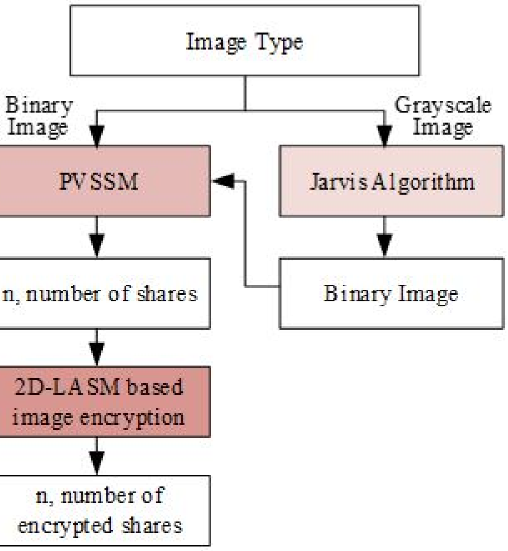


Figure 7. Chaotic Encryption-based PVSSM flow diagram for grayscale images

Firstly, the diagram of the Chaotic Encryption-based PVSSM for grayscale images is shown in Figure 7. In Figure 7, if the secret image is binary, PVSMM is applied to the secret image. n shares are obtained. Then, these n shares are encrypted by 2D-LASM based image encryption. If the secret image is grayscale, it is transformed into binary image by Jarvis algorithm. After that, this binary image is shared by PVSSM. Finally, n shares are encrypted by 2D-LASM based image encryption. Therefore, even if n encrypted shares are captured and stacked, no information about the secret image can be revealed.

In decryption process, firstly n encrypted shares are decrypted by 2D-LASM based image encryption. Then n shares are stacked and the secret image is restored.

The diagram of the Chaotic Encryption-based PVSSM for color images is shown in Figure 8. In Chaotic Encryption-based PVSSM for color images, the color secret image is first separated into RGB channels. After, RGB channels are transformed into binary images by the Jarvis algorithm. Then PVSSM is applied to binary images of RGB channel. n shares of R channel, n shares of G channel and n shares of B channel are obtained. After that, 2D-LASM based image encryption is applied to each channel shares. Finally, *3n* encrypted shares are obtained. Therefore, no information is revealed from the shares.

In decryption process for color image, the shares are first decrypted with 2D-LASM based image encryption. Then the shares are stacked for each color channel,

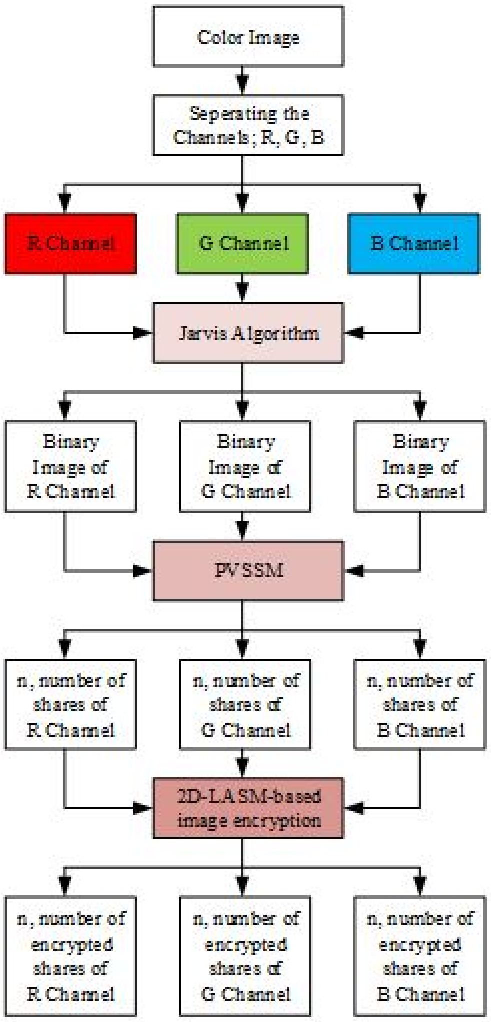

Color Image
Seperating the Channels; R, G, B
R Channel
G Channel
B Channel
Jarvis Algorithm
Binary Image of R Channel
Binary Image of G Channel
Binary Image of B Channel
PVSSM
n, number of shares of R Channel
n, number of shares of G Channel
n, number of shares of B Channel
2D-LASM-based image encryption
n, number of encrypted shares of R Channel
n, number of encrypted shares of G Channel
n, number of encrypted shares of B Channel

Figure 8. Chaotic encryption-based PVSSM flow diagram for color images

and the RGB channels are merged. Finally, the pixel values of 1 are changed into 255 to obtain the original color image in Table 2. In this way, 8 colors are obtained. The secret color image is restored.

This work is implemented with the MATLAB 2014b program on the AMD A10-4600M 2.30 GHz 8 GB computer. This method is applied to color Lena image size of 256×256. The color Lena image is divided into RGB channels in Figure 9. Then Jarvis algorithm is used and the RGB channels are converted into binary images in Figure 10.

The Resulting Pixel Values	**New Pixel Values of Color Images**	**The Obtained Color**
$(0,0,0)$	$(0,0,0)$	Black
$(0,0,1)$	$(0,0,255)$	Blue
$(0,1,0)$	$(0,255,0)$	Green
$(0,1,1)$	$(0,255,255)$	Cyan
$(1,0,0)$	$(255,0,0)$	Red
$(1,0,1)$	$(255,0,255)$	Magenta
$(1,1,0)$	$(255,255,0)$	Yellow
$(1,1,1)$	$(255,255,255)$	White

Table 2. Restored color

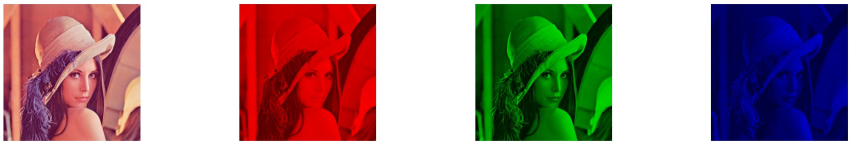

a) Secret image b) R channel c) G channel d) B channel

Figure 9. Color Lena image

PVSSM and 2D-LASM based image encryption is applied to the binary RGB channels respectively. 6 shares are produced as example. Figure 11 shows each of the encrypted shares and their histograms. In Figure 11, Chaotic Encryption-based PVSSM for color images is successfully applied. The encrypted shares of the histograms show a balanced distribution. Figures 11 a), c), and e) show the chaotic encrypted PVSSM implementation for all R, G and B channels, respectively. Figures 11 b), d), and f) represent histograms of the encrypted images for all R, G and B channels, respectively. The x-axis of these histograms shows the pixel values (0–255), and the y-axis shows the number of pixels. Thus, it will be difficult to obtain information with statistical methods [16]. The restored images are shown

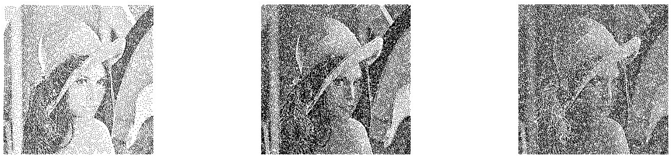
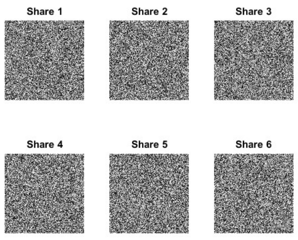
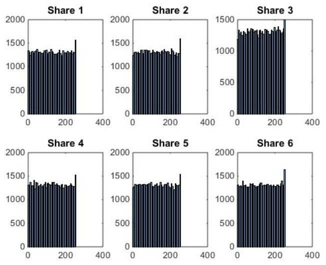

a) R channel b) G channel c) B channel

Figure 10. Binary channels

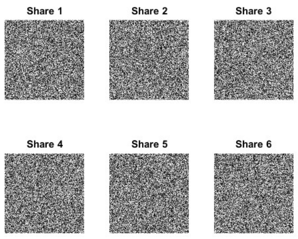
in Figure 12. Also PSNR and SSIM values of R, G, and B channels of color Lena image are shown in Table 3.



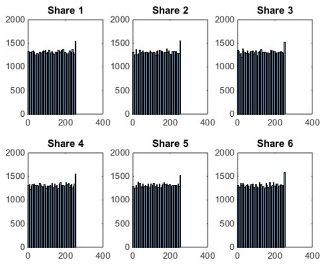
a) Encrypted R channel



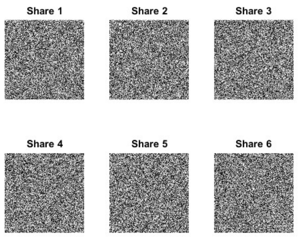
b) Histograms of Figure a)



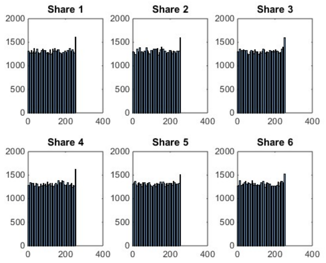
c) Encrypted G channel



d) Histograms of Figure c)



e) Encrypted B channel



f) Histograms of Figure e)

Figure 11. Chaotic Encryption-based PVSSM

5 EXPERIMENTAL RESULTS

In this section, we analyze the proposed method with some tests including histogram analysis, data loss attack, salt-pepper noise attack, differential attack, chi-square analysis and correlation analysis. NCPR, UACI, PSNR, SSIM and CQM

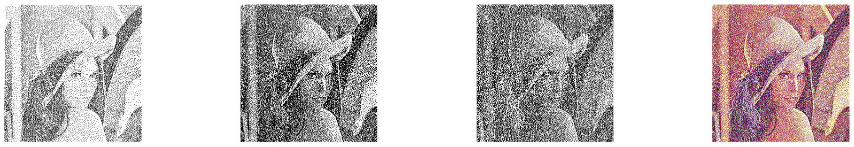


Figure 12. Restored images

	R Channel	**G Channel**	**B Channel**	**Average**
PSNR	60.5639	57.3671	57.5363	58.4891
SSIM	0.9983	0.9955	0.9958	0.9965

Table 3. PSNR and SSIM values of channels of color Lena image

are used for measurement. Experimental images are used from the USC-SIPI image database.

5.1 Histogram Analysis

The histograms of the encrypted shares are shown in Figures 11 b), d), and f). These histograms show a balanced distribution, so it is difficult to get information from the shares [16]. In addition, showing a balanced distribution, this will reduce the possibility of statistic attacks.

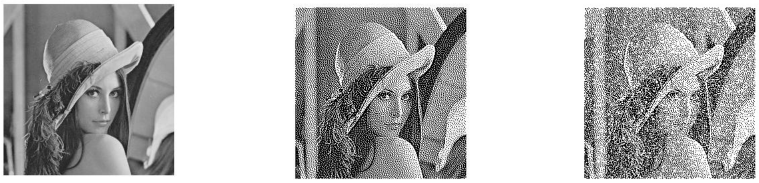
5.2 Data Loss Attack

Data loss attack means that some parts of the share lose their real value by adding noise to them. In this section, grayscale Lena image and color Lena image are used. Both images are encrypted with Chaotic Encryption-based PVSSM with 6 shares. Then data loss attacks of various types take place. Each encrypted share is attacked on the same region and the same noise ratio. Finally, the encrypted shares are decrypted.

The grayscale secret image and restored image is shown in Figure 13. Data loss attack is implemented to all shares. Only first share is shown for grayscale image in Figure 14 and only R channel share is shown for color image in Figure 15.

5.3 Salt-Pepper Noise Attack

In this section, grayscale Lena image and color Lena image are encrypted and then salt-pepper noise with different ratio is added to all shares. Finally, the encrypted shares are decrypted. Simulation results are shown in Figure 16.



a) Lena image b) Binary image of Figure a) c) Restored image

Figure 13. Grayscale test image

5.4 Differential Attack

NPCR (Number of pixels change rate) and UACI (Unified average changing intensity) metrics can measure the number of pixel changing rate with respect to differential attacks [31]. For secret image, randomly change one bit of a pixel and obtaining another secret image. NPCR and UACI are represented as Equations (2), (3) and (4), respectively [21]:

$$NPCR(C_1, C_2) = \sum_{i=1}^{M}\sum_{j=1}^{N} \frac{D(i,j)}{L} \times 100\,\%, \tag{2}$$

$$UACI(C_1, C_2) = \sum_{i=1}^{M}\sum_{j=1}^{N} \left(\frac{|(C_1(i,j) - (C_2(i,j)|}{T \times L} \right) \times 100\,\%, \tag{3}$$

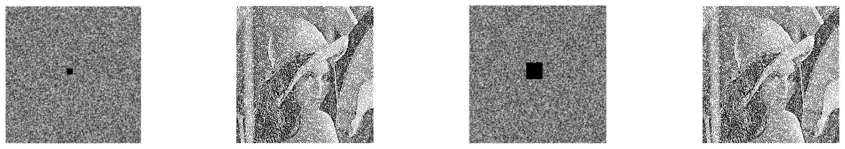
$$D(i,j) = \begin{cases} 0, & \text{if } (C_1(i,j) = (C_2(i,j), \\ 1, & \text{if } (C_1(i,j) \neq (C_2(i,j) \end{cases} \tag{4}$$

where C_1 and C_2 are encrypted images, M and N denote the size of images, i and j denote the pixels, T is the total number of pixels in the encrypted image, L is the largest allowed pixel value in the images and D is the bipolar array.

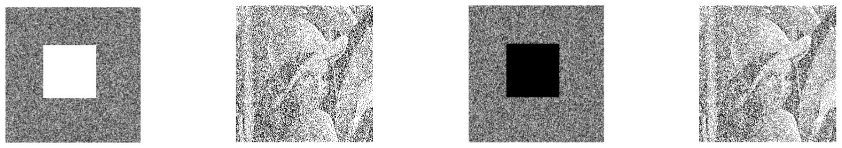
Experimental Image	**NCPR**	**UACI**
Lena Image	0.9963	0.3347
Barbara Image	0.9963	0.3346
Boat Image	0.9961	0.3346
Cameraman Image	0.9962	0.3342
Elanie Image	0.9961	0.3347
Peppers Image	0.9960	0.3344

Table 4. NCPR and UACI values

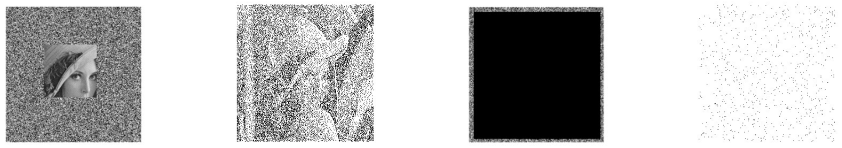
Table 4 shows NPCR and UACI values with respect to same secret key. When NPCR is equal to 0, it implies that all pixels remain the same between images.



a) 10 × 10 black square attack b) 10 × 10 black square result c) 30 × 30 black square attack d) 30 × 30 black square result



e) 100 × 100 white square attack f) 100×100 white square result g) 100×100 black square attack h) 100×100 black square result



i) 100 × 100 part of image attack j) 100×100 part of image attack result k) 90 % data loss attack l) 90 % data loss result

Figure 14. Data loss attack results for grayscale image

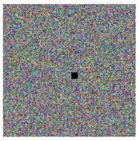
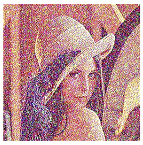
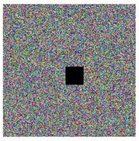
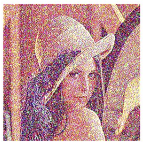
When NPCR is equal to 1, it implies that all pixel values are changed [31]. The ideal value of NCPR is 0.9961 and the ideal value of UACI is 0.3346 [32]. Table 4 shows that NPCR and UACI values are over 0.9960 and 0.3342, respectively, so the proposed method has good ability to resist differential attacks.

5.5 Chi-Square Analysis

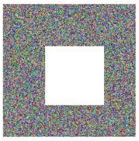
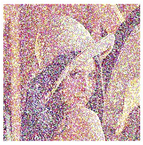
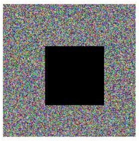
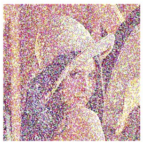
The chi-square parameter X^2 is defined as Equation (5):

$$X^2 = \sum_{i=1}^{256} \frac{(O_i - E_i)^2}{E_i} \tag{5}$$

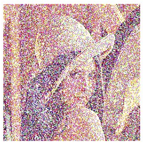
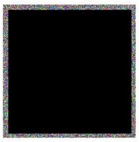
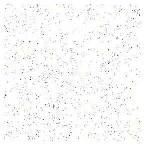
where i is the number of gray values, O_i and E_i are observed and expected occurrence of each gray value (0 to 255), respectively. In this experiment, we obtained 6 encrypted shares as an example, so it is obtained chi-square analysis between test image and its encrypted shares one by one. Then, the average is calculated. The

a) 10 × 10 black square attack b) 10 × 10 black square result c) 30×30 black square attack d) 30 × 30 black square result

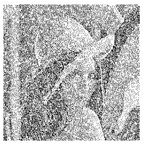
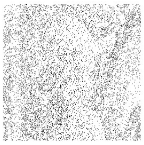
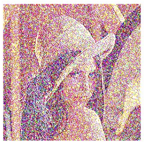
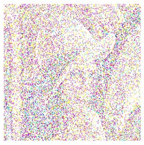
e) 100 × 100 white square attack f) 100 × 100 white square result g) 100 × 100 black square attack h) 100 × 100 black square result

i) 100 × 100 part of image attack j) 100 × 100 part of image attack result k) 90 % data loss attack l) 90 % data loss result

Figure 15. Data loss attack results for color image

values of chi-square for images under study are listed in Table 5. Table 5 shows that the histograms of the encrypted images are uniform.

a) 1 % salt-pepper noise for grayscale image b) 5 % salt-pepper noise for grayscale image c) 1 % salt-pepper noise for color image d) 5 % salt-pepper noise for color image

Figure 16. Salt-pepper noise attack result

Experimental Image	**Chi-Square Value**
Lena Image	255
Barbara Image	255
Boat Image	128
Cameraman Image	255
Elanie Image	255
Peppers Image	255

Table 5. Chi-square values

5.6 Correlation Analysis

The correlation coefficient $R_{x,y}$ between two grayscale adjacent pixels x and y are defined as Equation (6):

$$R_{x,y} = \frac{cov(x,y)}{\sqrt{D(x)D(y)}} \tag{6}$$

where $cov(x,y) = \frac{1}{T}\sum_{i=1}^{T}(x_i - E(x))(y_i - E(y))$, $E(x) = \frac{1}{T}\sum_{i=1}^{T} x_i$, $D(x) = \frac{1}{T}\sum_{i=1}^{T}(x_i - E(x))^2$ and T is the total number of pixels selected from the encrypted image.

In this experiment, the grayscale Lena image is used and 6 encrypted shares are obtained. 2 000 pairs of adjacent pixels are randomly selected. The horizontal, vertical and diagonal correlation coefficients of Lena image and original Lena image are shown in Table 6.

	1	2	3	4	5	6	Avg.	Lena
Horizontal	0.1356	0.1303	0.0821	0.0846	0.1210	0.1312	0.1141	0.9269
Vertical	0.1101	0.1007	0.1008	0.1317	0.0923	0.0775	0.1021	0.9699
Diagonal	0.1068	0.1089	0.1078	0.0815	0.1340	0.1013	0.1067	0.9129

Table 6. Correlation coefficients of encrypted shares of Lena image

As can be seen in Table 6, a small coefficient value means a weak correlation between two adjacent pairs [17]. There are no detectable correlations between the encypted share images and Lena image. Thus, the proposed method has good ability to resist statistical attacks.

5.7 Test Results

PSNR is a metric that measures the similarity between two images. PSNR can represent the image noise removal effects [33]. PSNR is represented as Equation (7):

$$PSNR = 10\log_{10}\frac{255^2}{\frac{1}{n\times m}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(x_{ij} - x'_{ij})^2} \tag{7}$$

where $n \times m$ denotes the size of the secret image, x_{ij} and (x'_{ij}) denote the pixel of secret image and restored image, respectively.

SSIM is the other metric of the similarity. If SSIM is equal to 1, two images are the same [34]. SSIM is of the following form [35]:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C1) + (2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C1) + (\sigma_x^2 + \sigma_y^2 + C2)} \tag{8}$$

where x and y are the secret image and the restored image, μ_x and μ_y denote the mean values, σ_{xy} denotes the covariance, σ_x and σ_y denote the variance of the secret image and restored image, respectively. $C1$ and $C2$ are the constants that inhibits division by 0.

CQM is the measurement of color image similarity [36]. In this method, first RGB channels are transformed to YUV channels:

$$Y = 0.257R + 0.504G + 0.098B + 16, \tag{9}$$

$$U = -0.148R - 0.291G + 0.439B + 128, \tag{10}$$

$$V = 0.439R - 0.368G - 0.071B + 128. \tag{11}$$

Then, C_W and R_W are calculated as $C_W = 0.0551$ and $R_W = 0.9449$. C_W is the weight on the human perception of the cones and R_W is the weight on the human perception of the rods.

Finally, CQM is calculated as Equation (12):

$$CQM = (PSNR_Y \times R_W) + \left(\frac{PSNR_U + PSNR_V}{2}\right) \times C_W. \tag{12}$$

Some test images are encrypted and their PSNR, SSIM and CQM values are calculated and shown in Table 7.

Experimental Image	**PSNR**	**SSIM**	**Experimental Image**	**CQM**
Binary Lena Image	58.3701	0.9958	Color Pepper Image	20.6227
Grayscale Lena Image	58.1951	0.9966	Color Lena Image	22.1913
Grayscale Barbara Image	57.8817	0.9962	Color House Image	22.4517
R Channel of Pepper Image	58.9144	0.9973	Color Jet plane Image	25.8311
G Channel of Pepper Image	57.7796	0.9958	Color Baboon Image	20.5208
B Channel of Pepper Image	56.5663	0.9941	–	–
Grayscale Building Image	59.8442	0.9978	–	–

Table 7. Test results of Chaotic Encryption-based PVSSM

As can be seen in Table 7, PSNR, SSIM and CQM values show that there is a similarity between the original secret image and the restored images which are both grayscale and color images. In addition, SSIM values are close to 1. Also the proposed method has not only worked with square image but also rectangular image because a building image with size of (232 × 311) is a rectangular.

Grayscale	PSNR	SSIM	Color	CQM
(10 × 10) black square	75.7988	1.000	(10 x 10) black square	25.4017
(30 × 30) black square	67.4142	0.9997	(30 x 30) black square	25.3883
(100 × 100) white square	57.6786	0.9959	(100 x 100) white square	25.3587
(100 × 100) black square	57.6786	0.9959	(100 x 100) black square	25.3587
(100 × 100) part of image	57.6762	0.9959	(100 x 100) part of image	25.3543
90 % data loss	52.0986	0.9728	90 % data loss	24.9894
1 % salt-pepper noise	58.7118	0.9971	1 % salt-pepper noise	25.1583
5 % salt-pepper noise	53.5124	0.9834	5 % salt-pepper noise	25.0015

Table 8. Attack type results

As can be seen in Table 8, when the same attack type rate increases, PSNR, SSIM and CQM values decrease. It is understood that the similarity ratio decreases as the attack rate increases. According to PSNR, SSIM and CQM values, the similarity ratio of the images is still high despite these attacks. SSIM values are also close to 1. In only 90 % data loss, the image is visually lost in Figures 14 and 15. As a result, it has been seen that the system is resistant to most attacks.

6 DISCUSSION

When experimental PSNR and SSIM values are investigated, high PSNR values represent the success of the proposed method. SSIM shows the proposed method's quality measure of one of the images being compared with original images. Tables 9 and 10 compare the proposed method with other methods.

Method	Image Type	PSNR
Wang et al. proposed method 1 [11]	Grayscale	51.13
Thien and Lin [37]	Grayscale	37.37
Yang et al. scheme 1 [38]	Grayscale	50.53
Yang et al. scheme 2 [38]	Grayscale	48.88
Pandey et al. [39]	Grayscale	57.6337
Goswami et al. [40]	Grayscale	37.24
Proposed Method	Grayscale	58.1951

Table 9. Encryption comparisons 1

In Table 9, Wang et al. [11] get PSNR value of 51.13 (proposed method 1), Thien and Lin [37] get PSNR value of 37.37, Yang et al. [38] get PSNR values of 50.53 and 48.88, Pandey et al. [39] get PSNR value of 57.6337 and Goswami et al. [40] get PSNR value of 37.24. A higher rate of PSNR value (58.1951) is obtained in the proposed method.

As can be seen in Table 10, Goswami et al. [40] get SSIM value of 0.9960 and, Weir et al. [41] get SSIM value of 0.9147. A higher rate of SSIM value of 0.9966 is obtained with the proposed method. SaiCandana and Anuradha [42] get SSIM values of 0.98 and 0.99 (color) and Huang et al. [43] get SSIM values of 0.9879 and 0.9925 (color). A higher rate of SSIM value (0.9965 – color) is obtained in the proposed method.

Method	**Image Type**	**SSIM**
Goswami et al. [40]	Grayscale	0.9960
Weir et al. [41]	Grayscale	0.9147
SaiCandana et al. [42] example 1	Color	0.98
SaiCandana et al. [42] example 2	Color	0.99
Huang et al. [43] example 1	Color	0.9879
Huang et al. [43] example 2	Color	0.9925
Proposed Method	Grayscale	0.9966
Proposed Method	Color	0.9965

Table 10. Encryption comparisons 2

Gorji et al. [44] get PSNR values of 53.6654 (1 % salt-pepper), 47.8638 (5 % salt-pepper) and 56.6781 ((10 × 10) data loss). A higher rate of PSNR values 58.7118 (1 % salt-pepper), 53.5124 (5 % salt-pepper) and 75.7988 6781 ((10 × 10) data loss) are obtained in the proposed method (see Table 11).

Method	**Attack Type**	**Image Type**	**SSIM**
Gorji et al. [44]	1 % salt-pepper	Grayscale	53.6654
Proposed method	1 % salt-pepper	Grayscale	58.7118
Gorji et al. [44]	5 % salt-pepper	Grayscale	47.8638
Proposed method	5 % salt-pepper	Grayscale	53.5124
Gorji et al. [44]	(10 × 10) data loss	Grayscale	56.6781
Proposed method	(10 × 10) data loss	Grayscale	75.7988

Table 11. Attack comparisons 1

In Lena image, Xu et al. [45] and Ye et al. [19] get NCPR values of 0.9962 and 0.9955, respectively. A higher rate of NCPR value 0.9963 is obtained in the proposed method. Xu et al. [45] and Ye et al. [19] get UACI values of 0.3351 and 0.3339, respectively. In the proposed method, UACI value 0.3347 is obtained.

In Boat image, Hua et al. [17] and Ye et al. [19] get NCPR values of 0.9963 and 0.9962, respectively. A lower rate of NCPR value 0.9961 is obtained in the proposed method. Hua et al. [17] and Ye et al. [19] get UACI values of 0.3353 and 0.3347, respectively. A lower rate of UACI value 0.3346 is obtained in the proposed method.

In Elanie image, Hua et al. [17] and Hua et al. [21] get NCPR values of 0.9962 and 0.9961, respectively. In the proposed method, UACI value 0.9961 is obtained. Hua et al. [17] and Hua et al. [21] get UACI values of 0.3342 and 0.3355, respectively. In the proposed method, UACI value 0.3347 is obtained.

Method	Image	NCPR	UACI
Xu et al. [45]	Lena	0.9962	0.3351
Ye et al. [19]	Lena	0.9955	0.3339
Proposed method	Lena	0.9963	0.3347
Hua et al. [17]	Boat	0.9962	0.3347
Ye et al. [19]	Boat	0.9963	0.3353
Proposed method	Boat	0.9961	0.3346
Hua et al. [17]	Elanie	0.9962	0.3342
Hua et al. [21]	Elanie	0.9961	0.3355
Proposed method	Elanie	0.9961	0.3347
Ye et al. [19]	Cameraman	0.9960	0.3353
Proposed method	Cameraman	0.9962	0.3342

Table 12. Attack comparisons 2

In Cameraman image, Ye et al. [19] get NCPR value of 0.9960 and UACI value of 0.3353. In the proposed method, NCPR value 0.9962 and UACI value 0.3342 are obtained (see Table 12).

The encrypted shares are tested by using many methods of histogram analysis, data loss attack and salt-pepper noise attack. These tests are applied to both grayscale and color images. The histograms of the encrypted shares are balanced and unbiased causing the statistical methods to fail in information extraction. The measurement results of data loss and salt-pepper noise attacks are obtained with PSNR and SSIM for binary and grayscale images whereas CQM is used for color images. When image resolutions are examined, the worst result is obtained with 90 % loss compared to (10 × 10), (30 × 30) and (100 × 100) data losses attacks. A better image resolution is obtained in 1 % salt-pepper noise attacks compared to 5 % salt-pepper noise attacks.

Moreover, the measurement results of differential attacks are obtained with NCPR and UACI. There is no big difference between these comparisons in Table 12. Chi-square analysis is also applied. The low values of chi-square confirm that the proposed method offers fairly high encryption effect [46].

Finally, all comparisons show that the proposed method has better PSNR, SSIM values and a good NPCR and UACI values. Thus it provides a better security.

7 CONCLUSION

In PVSSM, when the higher privileged shares are superimposed, the secret image is restored. Therefore, there is no need for all share images to reveal the secret image.

In this work, we have presented a solution to the security problem of PVSSM. We combined PVSSM with 2D-LASM based image encryption. This method was called Chaotic Encryption-based PVSSM. There are some advantages and novelty of the proposed method:

1. One of the VC and chaotic encryption methods are employed together.
2. Unlike traditional VC, this method has no pixel expansion.
3. When grayscale and color images are encrypted, Jarvis algorithm is applied.
4. Chaotic Encryption-based PVSSM is suitable for binary, grayscale, and color images.
5. For color images, CQM metric is used.

Finally, according to the test results, the proposed method is resistant against various attacks. Moreover, a better image quality has been obtained with the proposed method.

Since PVSSM is a lossy-method, certain loss may be possible while decrypting the encrypted image. Future works may ensure better image quality and compare CQM values for color images.

REFERENCES

[1] Hua, Z.—Zhou, Y.—Pun, C.-M.—Chen, C. L. P.: Image Encryption Using 2D Logistic-Sine Chaotic Map. IEEE International Conference on Systems, Man and Cybernetics, 2014, pp. 3229–3234, doi: 10.1109/smc.2014.6974425.

[2] Kapoor, D.—Keshari, S.—Gaur, S. K.: An Overview of Visual Cryptography. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 4, 2014, No. 2, pp. 103–110.

[3] Thien, C.-C.—Lin, J.-C.: Secret Image Sharing. Computers and Graphics, Vol. 26, 2002, No. 5, pp. 765–770, doi: 10.1016/s0097-8493(02)00131-0.

[4] Naor, M.—Shamir, A.: Visual Cryptography. In: De Santis, A. (Ed.): Advances in Cryptology – EUROCRYPT '94. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 950, 1995, pp. 1–12, doi: 10.1007/bfb0053419.

[5] Pandey, A.—Som, S.: Applications and Usage of Visual Cryptography: A Review. 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), 2016, pp. 375–381, doi: 10.1109/icrito.2016.7784984.

[6] Wei, K.-J.—Lee, J.-S.—Chen, S.-J.: Enhancing the Security of Credit Card Transaction Based on Visual DSC. KSII Transactions on Internet and Information Systems, Vol. 9, 2015, No. 3, pp. 1231–1245, doi: 10.3837/tiis.2015.03.022.

[7] Hou, Y.-C.: Visual Cryptography for Color Images. Pattern Recogniton, Vol. 36, 2015, No. 7, pp. 1619–1629, doi: 10.1016/s0031-3203(02)00258-3.

[8] Fang, W.-P.—Lin, J.-C.: Progressive Viewing and Sharing of Sensitive Images. Pattern Recognition and Image Analysis, Vol. 16, 2006, No. 4, pp. 632–636, doi: 10.1134/s1054661806040080.

[9] Jin, D.—Yan, W.-Q.—Kankanhalli, M. S.: Progressive Color Visual Cryptography. Journal of Electronic Imaging, Vol. 14, 2005, No. 3, Art. No. 033019, doi: 10.1117/1.1993625.

[10] Fang, W.-P.: Friendly Progressive Visual Secret Sharing. Pattern Recognition, Vol. 41, 2008, No. 4, pp. 1410–1414, doi: 10.1016/j.patcog.2007.09.004.

[11] Wang, R.-Z.—Chien, Y.-F.—Lin, Y.-Y.: Scalable User-Friendly Image Sharing. Journal of Visual Communication and Image Representation, Vol. 21, 2010, No. 7, pp. 751–761, doi: 10.1016/j.jvcir.2010.06.001.

[12] Chen, C.-C.—Chen, C.-C.—Lin, Y.-C.: Weighted Modulated Secret Image Sharing Method. Journal of Electronic Imaging, Vol. 18, 2009, No. 4, Art. No. 043011, doi: 10.1117/1.3268362.

[13] Lin, S.-J.—Chen, L. S.-T.—Lin, J.-C.: Fast-Weighted Secret Image Sharing. Optical Engineering, Vol. 48, 2009, No. 7, Art. No. 077008, doi: 10.1117/1.3168644.

[14] Li, P.—Yang, C.-N.—Wu, C.-C.—Kong, Q.—Ma, Y.: Essential Secret Image Sharing Scheme with Different Importance of Shadows. Journal of Visual Communication and Image Representation, Vol. 24, 2013, No. 7, pp. 1106–1114, doi: 10.1016/j.jvcir.2013.07.005.

[15] Hou, Y.-C.—Quan, Z.-Y.—Tsai, C.-F.: A Privilege-Based Visual Secret Sharing Model. Journal of Visual Communication and Image Representation, Vol. 33, 2015, pp. 358–367, doi: 10.1016/j.jvcir.2015.10.005.

[16] Mao, Y.—Chen, G.: Chaos-Based Image Encryption. Chapter 8. In: Bayro Corrochano, E. (Ed.): Handbook of Geometric Computing. Springer, 2005, pp. 231–265.

[17] Hua, Z.—Zhou, Y.: Image Encryption Using 2D Logistic-Adjusted-Sine Map. Information Sciences, Vol. 339, 2016, pp. 237–253, doi: 10.1016/j.ins.2016.01.017.

[18] Alvarez, G.—Li, S.: Cryptanalyzing a Nonlinear Chaotic Algorithm (NCA) for Image Encryption. Communications in Nonlinear Science and Numerical Simulations, Vol. 14, 2009, No. 11, pp. 3734–3749, doi: 10.1016/j.cnsns.2009.02.033.

[19] Ye, G.—Zhao, H.—Chai, H.: Chaotic Image Encryption Algorithm Using Wave-Line Permutation and Block Diffusion. Nonlinear Dynamics, Vol. 83, 2016, No. 4, pp. 2067–2077, doi: 10.1007/s11071-015-2465-7.

[20] Arroyo, D.—Rhouma, R.—Alvarez, G.—Li, S.—Fernandez, V.: On the Security of a New Image Encryption Scheme Based on Chaotic Map Lattices. Chaos: An Interdisciplinary Journal of Nonlinear Science, Vol. 18, 2008, No. 3, Art. No. 033112, doi: 10.1063/1.2959102.

[21] Hua, Z.—Zhou, Y.—Pun, C. M.—Chen, C. L. P.: 2D Sine Logistic Modulation Map for Image Encryption. Information Sciences, Vol. 297, 2015, pp. 80–94, doi: 10.1016/j.ins.2014.11.018.

[22] Tang, Y.—Guan, X.: Parameter Estimation of Chaotic System with Time-Delay: A Differential Evolution Approach. Chaos, Solitons and Fractals, Vol. 42, 2009, No. 5, pp. 3132–3139, doi: 10.1016/j.chaos.2009.04.045.

[23] Wu, Y.—Noonan, J. P.—Yang, G.—Jin, H.: Image Encryption Using the Two-Dimensional Logistic Chaotic Map. Journal of Electronic Imaging, Vol. 21, 2012, No. 1, Art. No. 013014, doi: 10.1117/1.jei.21.1.013014.

[24] Ye, G.: Image Scrambling Encryption Algorithm of Pixel Bit Based on Chaos Map. Pattern Recognition Letters, Vol. 31, 2010, No. 5, pp. 347–354, doi: 10.1016/j.patrec.2009.11.008.

[25] Liu, F.—Wu, C.: Embedded Extended Visual Cryptography Schemes. IEEE Transactions on Information Forensics and Security, Vol. 6, 2011, No. 2, pp. 307–322, doi: 10.1109/tifs.2011.2116782.

[26] Chang, C.-C.—Hsieh, Y.-P.—Liao, C.-C.: A Visual Secret Sharing Scheme for Progressively Restoring Secrets. Journal of Electronic Science and Technology, Vol. 9, 2011, No. 4, pp. 325–331.

[27] Soman, N.— Baby, S.: XOR-Based Visual Cryptography. International Journal on Cybernetics and Informatics (IJCI), Vol. 5, 2016, No. 2, pp. 253–264, doi: 10.5121/ijci.2016.5228.

[28] Padhmavathi B.—Kumar, P. N.: A Novel Mathematical Model for (t, n)-Threshold Visual Cryptography Scheme. International Journal of Computer Trends and Technology (IJCTT), Vol. 12, 2014, No. 3, pp. 126–129, doi: 10.14445/22312803/ijctt-v12p125.

[29] Hou, Y.-C.—Quan, Z.-Y.: Progressive Visual Cryptography with Unexpanded Shares. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 21, 2011, No. 11, pp. 1760–1764, doi: 10.1109/tcsvt.2011.2106291.

[30] Nikate, P. M.—Mujawar, I. I.: Performance Evaluation of Floyd Steinberg Halftoning and Jarvis Halftoning Algorithms in Visual Cryptography. International Journal of Innovations in Engineering and Technology, Vol. 5, 2015, No. 1, pp. 336–342.

[31] Wu, Y.—Noonan, J. P.—Agaian, S.: NPCR and UACI Randomness Tests for Image Encryption. Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications (JSAT), April Edition, 2011, pp. 31–38.

[32] Borujeni, S. E.—Eshghi, M.: Chaotic Image Encryption Design Using Tompkins-Paige Algorithm. Mathematical Problems in Engineering, Vol. 2009, 2009, Art. No. 762652, 22 pp., doi: 10.1155/2009/762652.

[33] Yu, J.—Tan, L.—Zhou, S.—Wang, L.—Wang, C.: Image Denoising Based on Adaptive Fractional Order Anisotropic Diffusion. KSII Transactions on Internet and Information Systems, Vol. 11, 2017, No. 1, pp. 436–450, doi: 10.3837/tiis.2017.01.023.

[34] Kocak, C.: CLSM: Couple Layered Security Model A High-Capacity Data Hiding Scheme Using with Steganography. Image Analysis and Stereology, Vol. 36, 2017, No. 1, pp. 15–23, doi: 10.5566/ias.1482.

[35] Wang, Z.—Bovik, A. C.—Sheikh, H. R.—Simoncelli, E. P.: Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, Vol. 13, 2004, No. 4, pp. 600–612, doi: 10.1109/tip.2003.819861.

[36] Yalman, Y.—Ertürk, İ.: A New Color Image Quality Measure Based on YUV Transformation and PSNR for Human Vision System. Turkish Journal of Electrical Engineering and Computer Sciences, Vol. 21, 2013, No. 2, pp. 603–612, doi: 10.3906/elk-1111-11.

[37] Thien, C.-C.—Lin, J.-C.: An Image Sharing Method with User-Friendly Shadow Images. IEEE Transactions on Circuits and Systems for Video Technology, Vol. 13, 2003, No. 12, pp. 1161–1169, doi: 10.1109/tcsvt.2003.819176.

[38] Yang, C.-N.—Yu, K.-H.—Lukac, R.: User-Friendly Image Sharing Using Polynomials with Different Primes. International Journal of Imaging Systems and Technology, Vol. 17, 2007, No. 1, pp. 40–47, doi: 10.1002/ima.20096.

[39] Pandey, D.—Rawat, U. S.—Kumar, A.: Robust Progressive Block Based Visual Cryptography with Chaotic Map. Journal of Discrete Mathematical Sciences and Cryptography, Vol. 19, 2016, No. 5–6, pp. 1025–1040, doi: 10.1080/09720529.2015.1132040.

[40] Goswami, A.—Mukherjee, R.—Ghoshal, N.: Chaotic Visual Cryptography Based Digitized Document Authentication. Wireless Personal Communications, Vol. 96, 2017, No. 3, pp. 3585–3605, doi: 10.1007/s11277-017-4088-4.

[41] Weir, J.—Yan, W.—Kankanhalli, M. S.: Image Hatching for Visual Cryptography. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) – Special Issue on Multimedia Security, Vol. 8, 2012, No. 2S, Art. No. 32, 15 pp., doi: 10.1145/2344436.2344438.

[42] SaiChandana, B.—Anuradha, S.: A New Visual Cryptography Scheme for Color Images. International Journal of Engineering Science and Technology, Vol. 2, 2010, No. 6, pp. 1997–2000.

[43] Huang, H.-C.—Lu, Y.-Y.—Liu, J.: Ownership Protection for Progressive Image Transmission with Reversible Data Hiding and Visual Secret Sharing. Optik, Vol. 127, 2016, No. 15, pp. 5950–5960, doi: 10.1016/j.ijleo.2016.04.011.

[44] Gorji, R. B.—Shirvani, M. H.—Mooziraji, F. R.: A New Image Encryption Method Using Chaotic Map. Journal of Multidisciplinary Engineering Science and Technology (JMEST), Vol. 2, 2015, No. 2, pp. 251–256.

[45] Xu, L.—Li, Z.—Li, J.—Hua, W.: A Novel Bit-Level Image Encryption Algorithm Based on Chaotic Maps. Optics and Lasers in Engineering, Vol. 78, 2016, pp. 17–25, doi: 10.1016/j.optlaseng.2015.09.007.

[46] Ahmad, M.—Alsharari, H. D.—Nizam, M.: Security Improvement of an Image Encryption Based on mPixel-Chaotic-Shuffle and Pixel-Chaotic-Diffusion. European Journal of Scientific Research, Vol. 98, 2013, No. 3, 14 pp.

Aytekin Yildizhan received his B.Sc. degree in computer engineering from the Izmir Institute of Technology, Izmir, Turkey in 2008. He received his M.Sc. degree in computer engineering from Gazi University, Ankara, Turkey in 2013. He is a Ph.D. candidate in computer engineering at Hacettepe University, Ankara, Turkey. He currently works as an engineer first lieutenant in Turkish Armed Forces. He does research in cognitive science, visual secret sharing and visual cryptography.

Nurettin Topaloglu is Professor of the Computer Engineering Department at the Technology Faculty of Gazi University in Turkey. He received his B.Sc. in electronics, M.Sc. in electronics and computer education and Ph.D. in electric education. His research interests are computer architecture and organization, informatics technologies and information security. He has been involved in research areas in deep and machine learning. He developed the educational software Visual 6502 microprocessor simulator for teaching computer architecture. He is a writer of microprocessors and assembly language, and x86 microprocessor architecture and assembly language in Turkish.

Computing and Informatics, Vol. 38, 2019, 728–764, doi: 10.31577/cai_2019_3_728

PETRI NETS AT MODELLING AND CONTROL OF DISCRETE-EVENT SYSTEMS WITH NONDETERMINISM – PART 2

František Čapkovič

Institute of Informatics
Slovak Academy of Sciences
Dúbravská cesta 9
845 07 Bratislava, Slovakia
e-mail: `Frantisek.Capkovic@savba.sk`

Abstract. Discrete-Event Systems (DES) are discrete in nature. Petri Nets (PN) are one of the most widespread tools for DES modelling, analyzing and control. Different kinds of PN can be used for such purposes. Some of them were described in [3], being the first part of this paper. Here, the applicability of Labelled PN (LbPN) and Interpreted PN (IPN) for modelling and control of nondeterministic DES, especially with uncontrollable and/or unobservable transitions in the models, will be pointed out. Moreover, another kinds of nondeterminism in DES (errors, failures) will be modelled, and the possibilities of the error recovery of failed system will be presented.

Keywords: Analyzing, control synthesis, discrete-event systems, error recovery, interpreted Petri nets, modelling, labelled Petri nets, place/transition Petri nets, uncertainty, uncontrollable transitions, unmeasurable places, unobservable transitions

Mathematics Subject Classification 2010: 93-C65, 93-C30

1 INTRODUCTION AND PRELIMINARIES

This paper is the Part 2 of the the paper which had started by the Part 1 published as [3]. In the Part 1 the basic background about several crucial kinds of Petri nets

(PN), including Place/Transition PN (P/T PN), Timed PN (TPN), Controlled PN (CtPN), Interpreted PN (IPN) and Labelled PN (LbPN), was presented as well as a simple application of them for modelling and control of flexible manufacturing systems (FMS). In this paper, especially the IPN and LbPN will be used in modelling and control of discrete-event systems (DES), namely some kinds of FMS and a segment of transport systems, to tend towards the applications in practice. Two principled kinds of uncertainties in PN models of DES will be analyzed here:

1. the effect of uncontrollable and unobservable transitions;
2. the occurrence of different kinds of errors/failures.

While resolving the former problem requires the usage of special kinds of PNs (IPN or LbPN), resolving the latter one requires the usage of an error recovery procedure. Both kinds of uncertainties as well as their combination will be analyzed here by PN-based approach.

1.1 Mathematical Model of Petri Nets

At the beginning let us recall the principle definition of the basal PN – P/T PN. As it was introduced in the Part 1 of this paper – see [3] – mathematical expression of P/T PN consists of the (i) expression of the PN structure – PN is a bipartite directed graph $\langle P, T, F, G\rangle$ with $P = \{p_1, \dots, p_n\}$, $|P| = n$ being the set of places p_i, $i = 1, \dots, n$; $T = \{t_1, \dots, t_m, |T| = m\}$ being the set of transitions t_j, $j = 1, \dots, m$; F being the set of directed arcs from places to transitions; G being the set of directed arcs from transitions to places; (ii) description of PN dynamics (the marking development) in the form of the limited discrete linear equation

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{B}.\mathbf{u}_k, \quad k = 0, 1, \dots, N, \tag{1}$$

$$\mathbf{F}.\mathbf{u}_k \leq \mathbf{x}_k, \tag{2}$$

with the structural matrix $\mathbf{B} = \mathbf{G}^T - \mathbf{F}$ ($\mathbf{F}$, $\mathbf{G}^T$ correspond to F, G). Here, $\mathbf{x}_k = (\sigma^k_{p_1}, \dots, \sigma^k_{p_n})^T$ with entries $\sigma^k_{p_i} \in \{0, 1, \dots, \infty\}$, representing the states of particular places, is the PN state vector in the k^{th} step of the dynamics development; $\mathbf{u}_k = (\gamma^k_{t_1}, \dots, \gamma^k_{t_m})^T$ with entries $\gamma^k_{p_i} \in \{0, 1\}$, representing the states of particular transitions (either enable – when 1, or disable – when 0) is the control vector; $\mathbf{F}$, $\mathbf{G}^T$ are, respectively, incidence matrices of arcs from places to transitions and contrariwise.

Particulars about P/T PN as well as about other kinds of Petri nets which will be used here (IPN, LbPN) are introduced and explained in detail in the Part 1 of the paper – i.e. in [3].

1.2 Place Invariants in Supervisor Synthesis

In order to control the PN model of a plant, a supervisor has to be proposed. To synthesize the supervisor based on place invariants (P-invariants), where the invariants are defined as columns of the matrix $\mathbf{W}$ given as follows:

$$\mathbf{W}^T.\mathbf{B} = \mathbf{0}, \tag{3}$$

it is necessary to enforce the suitable restrictive condition on the state vectors of the system (1) as follows:

$$\mathbf{L}.\mathbf{x} \leq \mathbf{b}. \tag{4}$$

Here, $\mathbf{L}$ is a matrix of integers and $\mathbf{b}$ is a vector of integers representing some restrictions on the linear combination of corresponding entries of the state vector. To eliminate the inequality in (4) it is needful to insert there the $(n_s \times 1)$ vector $\mathbf{x}_s$ consisting of *slack* variables. Thus,

$$\mathbf{L}.\mathbf{x} + \mathbf{x}_s = \mathbf{L}.\mathbf{x} + \mathbf{I}_s.\mathbf{x}_s = (\mathbf{L}\mathbf{I}_s).\begin{pmatrix} \mathbf{x} \\ \mathbf{x}_s \end{pmatrix} = \mathbf{b} \tag{5}$$

where $\mathbf{I}_s$ is the $(s \times s)$-dimensional identity matrix. To synthesize the supervisor with a structure $\mathbf{B}_s$ (so far unknown), force the matrix $(\mathbf{L}, \mathbf{I}_s)$ into (3) instead of $\mathbf{W}^T$ and the matrix $(\mathbf{B}^T, \mathbf{B}_s^T)^T$ instead of $\mathbf{B}$. Hence, the structure of the supervisor was found as

$$(\mathbf{L}\mathbf{I}_s).\begin{pmatrix} \mathbf{B} \\ \mathbf{B}_s \end{pmatrix} = \mathbf{0}; \quad \mathbf{B}_s = -\mathbf{L}.\mathbf{B}; \quad \mathbf{B}_s = \mathbf{G}_s^T - \mathbf{F}_s. \tag{6}$$

Here, $\mathbf{B}_s$ represents the interconnections of n_s additional places (called *monitors*) with the original PN. $\mathbf{F}_s, \mathbf{G}_s$ obtained by the decomposition of $\mathbf{B}_s$ are the incidence matrices of the directed arcs. The monitors together with the arcs create the supervisor. The directed arcs realize interconnections of the supervisor with the PN model of a plant in both directions:

1. from the supervisor places to a part of the PN model transitions (a set of the model inputs) and
2. from another part of the PN model transitions (a set of the model outputs) to the places of the supervisor.

The initial state of the supervisor follows from (5) in the form

$$\mathbf{x}_s^0 = \mathbf{b} - \mathbf{L}.\mathbf{x}_0. \tag{7}$$

1.3 Uncontrollable and Unobservable Transitions

When no uncontrollable and unobservable transitions occur in the PN model (it is the ideal case), the supervisor is able to observe all transitions. Consequently, it can either prevent a transition from firing or to fire it – i.e. to enforce a desired behavior on it. However, in practice usually such an assumption is not valid. There are two possibilities for transitions in PN models of realistic systems: (i) a transition $t_j \in T$

may be uncontrollable; (ii) a transition $t_j \in T$ may be unobservable. In the former case the supervisor is not able to prevent the transition from firing – i.e. there is no arc from any supervisor place to t_j. In the latter case the supervisor is not able to detect the firing of the transition – i.e. there is no arc from any transition of the PN model of the plant to the supervisor place. In other words, the feedback from the PN model of the plant to the supervisor is missing. It means that there are neither arcs from an unobservable transition t_j to any controller place p_i, $i = 1, \dots s$, nor the arc from any controller place p_i, $i = 1, \dots s$ to t_j.

It appears that a transition being unobservable is also uncontrollable. Therefore, controllability of a transition implies its observability. Consequently, three different kinds of transitions are distinguished:

1. controllable;
2. uncontrollable, but observable; and
3. uncontrollable and unobservable.

Accordingly, the set T of transitions has three following subsets $T = T_{o,c} \cup T_{o,uc} \cup T_{uo,uc}$. Hence, it can be written that $T_{uc} = T_{o,uc} \cup T_{uo,uc}$. Here, $T_{o,c}$ and T_{uc} are, respectively, the subsets of controllable and uncontrollable transitions. $T_{o,uc}$ represents the set of uncontrollable but observable transitions, and $T_{uo,uc}$ consists of transitions that are both uncontrollable and unobservable.

1.4 Errors and Error Recovery

Many times errors of different kinds occur during DES operation. They bring another kind of nondeterminism into DES performance. For instance, in a specific kind of FMS like robotic cells a part may drop out (i.e. fall down) from the robot gripper, in a simple railroad crossing the control system of crossing gate may fail (e.g. the premature gate raising), etc. After the occurrence of the fault the system development is different than the standard one. In such a case the system has to detect what was wrong and recover the normal behaviour, i.e. to eliminate the influence of the error on the system behaviour. Namely, the error recovery is the set of actions that must be performed in order to return the system to its normal state. To do this, it is necessary

1. to synthesize the recovery sequence, and
2. to extend the scope of the controller activity in order to deal with the fault.

The problem is directly connected with reachability.

It is necessary to emphasize that errors of the mentioned kinds cannot be excluded, either in deterministic PN models or in PN models with uncontrollable/unobservable transitions.

1.5 The Paper Organization

The paper is organized as follows:

1. the discussion about the nondeterminism caused by unobservable/uncontrollable transitions and unmeasurable (unobservable) places in PN models of DES and about how to deal with it;
2. the discussion about how to model errors/failures and the dealing with the matter of the error recovery in DES;
3. the introduction of three case studies, the first one based on the IPN model, the second one on the LbPN model and the third one (specific one) based on the PN model of RAS (resource allocation systems) where a special kind of nondeterminism occurs.

2 DEALING WITH UNCONTROLLABLE AND UNOBSERVABLE TRANSITIONS

In case when solely controllable and observable transitions occur, the PN model (1) may be used, and the supervisor may be synthesized by means of (6). In the opposite case, i.e. when uncontrollable and/or unobservable transitions occur, the situation is much more complicated.

2.1 Presence of Uncontrollable and/or Unobservable Transitions

Taking into account the previous relations concerning the PN model, the supervisor synthesis, and the indexing the particular part of transitions described in the Section 1.3 we have to cogitate about how to express the PN model and the supervisor synthesis in this case. We can assume that the incidence matrix **B** of the PN model of plant has the form

$$\mathbf{B} = [\mathbf{B}_{o,c}\mathbf{B}_{uc}] = [\mathbf{B}_{o,c}\mathbf{B}_{o,uc}\mathbf{B}_{uo,uc}] \tag{8}$$

where $\mathbf{B}_{uc} = [\mathbf{B}_{o,uc}\mathbf{B}_{uo,uc}]$. Such a configuration of submatrices of **B** corresponds to ordering the transitions:

1. $t_1, \dots, t_{mc}$ (for controllable and observable transitions);
2. $t_{mc+1}, \dots, t_{mc+mo}$ (for uncontrollable, but observable, transitions);
3. and $t_{mc+mo+1}, \dots, t_m$ (for uncontrollable and unobservable transitions).

2.1.1 Ideal Enforceability

The ideal enforceability of control interferences occurs if there are no arcs from controller places to transitions $t \in T_{uc}$ and no arcs from transitions $t \in T_{uo,uc}$ to controller places. It can be easily checked. The controller incidence matrix:

$$\mathbf{B}_s = -\mathbf{L}\mathbf{B} = -\mathbf{L}\left[\mathbf{B}_{o,c}\mathbf{B}_{uc}\right] = -\mathbf{L}\left[\mathbf{B}_{o,c}\mathbf{B}_{o,uc}\mathbf{B}_{uo,uc}\right]. \tag{9}$$

In such a case the following inequality has to be valid:

$$-\mathbf{L}\mathbf{B}_{o,uc} \geq \mathbf{0} \tag{10}$$

where the signs of the inequality are performed element by element. This relation expresses the fact that the firing of any uncontrollable, but observable, transition does not depend on the number of tokens in a controller place, but may increase this number. Moreover, the following relation has to be valid:

$$\mathbf{L}\mathbf{B}_{uo,uc} = \mathbf{0}. \tag{11}$$

This equation expresses the fact that the firing of any uncontrollable and unobservable transition will not affect the number of tokens in a controller place. Finally, for the initial state of the supervisor, the following relation has to hold:

$$\mathbf{x}_\mathbf{0}^\mathbf{s} = \mathbf{b} - \mathbf{L}\mathbf{x}_0 \geq \mathbf{0}. \tag{12}$$

This inequality expresses that the initial state vector (i.e. initial marking) $\mathbf{x}_0^s$ of the supervisor is a nonzero vector. Note, that it is the same relation as $\mathbf{L}\mathbf{x}_0 \leq \mathbf{b}$.

Supervisory control is a procedure of enforcing the external constraints on a system to be controlled. If a desired control specification is ideally enforceable, the supervisor respects the observability and controllability constraints. When the supervisor respects the uncontrollability and unobservability constraints of the plant, it is marked as admissible. Consequently, the presence of uncontrollable and/or unobservable transitions does not pose any problem. However, in real conditions the enforceability may not exist. Even, the ideal enforceability does not exist.

The specification (4) or (5) is said to be ideally enforceable, if the (ideal) supervisor/controller represented by $\mathbf{B}_s$ and $\mathbf{x}_0^s$ can be realized (i.e., if it is feasible) – i.e., if in case of unobservable and uncontrollable transitions there are no arcs from controller places to transitions in T_{uc} and no arcs from transitions in $T_{uo,uc}$ to controller places. In models of real systems it is not always possible.

2.1.2 Real Enforceability

In real conditions the ideal enforceability does not exist. In general, the solution will not be possible in terms of the original specification. In such a case it is necessary to find modified constrains (control specifications) in the form $\mathbf{L}'.\mathbf{x} \leq \mathbf{b}'$ and compute the controller $\mathbf{B}_s = -\mathbf{L}'\mathbf{B}$ for the new specifications. Here, $\mathbf{x}_\mathbf{0}^\mathbf{s} = \mathbf{b}' - \mathbf{L}'\mathbf{x}_0$. The problem is solved when we succeed in finding a suitable $\mathbf{L}'$ an $\mathbf{b}'$.

As it was proved in [15], the specifications can have the following form:

$$\mathbf{L}' = (\mathbf{R}_1 + \mathbf{R}_2\mathbf{L}), \tag{13}$$

$$\mathbf{b}' = \mathbf{R}_2(\mathbf{b} + \mathbf{1}^{n_s\times 1}) - \mathbf{1}^{n_s\times 1} \tag{14}$$

where $\mathbf{1}^{n_s\times 1} = (1, \dots, 1)^T$, $\mathbf{R}_1 \in \mathbb{Z}^{n_s\times n}$ satisfies $\mathbf{R}_1.\mathbf{x} \geq \mathbf{0}$, $\forall\mathbf{x}$, $\mathbf{R}_2 \in \mathbb{Z}^{n_s\times n_s}$ be a diagonal matrix of natural numbers (i.e. positive-definite diagonal matrix of integers).

Choose the entries for $\mathbf{R}_1$ and $\mathbf{R}_2$ to ensure the ideal enforceability. According to conditions (10)–(12) concerning the ideal enforceability introduced and described above, the following relations have to hold:

$$(\mathbf{R}_1 + \mathbf{R}_2\mathbf{L})\mathbf{B}_{o,uc} \leq \mathbf{0}, \tag{15}$$

$$(\mathbf{R}_1 + \mathbf{R}_2\mathbf{L})\mathbf{B}_{uo,uc} = \mathbf{0}, \tag{16}$$

$$(\mathbf{R}_1 + \mathbf{R}_2\mathbf{L})\mathbf{x}_0 \leq \mathbf{R}_2(\mathbf{b} + \mathbf{1}^{n_s\times 1}) - \mathbf{1}^{n_s\times 1}. \tag{17}$$

In [15] it was proved that when

$$[\mathbf{R}_1\mathbf{R}_2]\,.\left[\begin{array}{cccc} \mathbf{B}_{uc} & \mathbf{B}_{uo} & -\mathbf{B}_{uo} & \mathbf{x}_0 \\ \mathbf{L}.\mathbf{B}_{uc} & \mathbf{L}.\mathbf{B}_{uo} & -\mathbf{L}.\mathbf{B}_{uo} & \mathbf{L}.\mathbf{x}_0 - \mathbf{b} - \mathbf{1}^{n_c\times 1} \end{array}\right] \leq \left[\begin{array}{cccc} \mathbf{0} & \mathbf{0} & \mathbf{0} & -\mathbf{1}^{n_c\times 1} \end{array}\right], \tag{18}$$

then the controller

$$\mathbf{B}_s = -(\mathbf{R}_1 + \mathbf{R}_2\mathbf{L})\mathbf{B} = -\mathbf{L}'\mathbf{B}, \tag{19}$$

$$\mathbf{x}_s^0 = \mathbf{R}_2(\mathbf{b} + \mathbf{1}^{n_c\times 1}) - \mathbf{1}^{n_c\times 1} - (\mathbf{R}_1 + \mathbf{R}_2\mathbf{L})\mathbf{x}_0 = \mathbf{b}' - \mathbf{L}'\mathbf{x}_0 \tag{20}$$

exists. Moreover, it causes that all subsequent markings of the closed loop system satisfy the constraint $\mathbf{L}\mathbf{x} \leq \mathbf{b}$. This is achieved without any attempt to inhibit uncontrollable transitions and without detecting unobservable transitions. The advantage of such an approach is that the matrices $\mathbf{R}_1$, $\mathbf{R}_2$ can be generated.

2.1.3 Example – Comparison of Ideal and Real Enforceability

To illustrate the difference between the ideal and real enforceability of control interferences let us introduce the simple PN given in Figure 1 where the transition t_4 is uncontrollable. Namely, the matrix $\mathbf{B}$ consists of two submatrices $(\mathbf{B}_{o,c}, \mathbf{B}_{o,uc})$ as follows:

$$\mathbf{B} = (\mathbf{B}_{o,c}, \mathbf{B}_{o,uc}) = \left(\begin{array}{ccc|c} 0 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 1 \end{array}\right). \tag{21}$$

Suppose, that in each step k of the system evolution the condition

$$\sigma_{p_2} + 3.\sigma_{p_4} \leq 3, \quad k = 0, 1, \dots \tag{22}$$

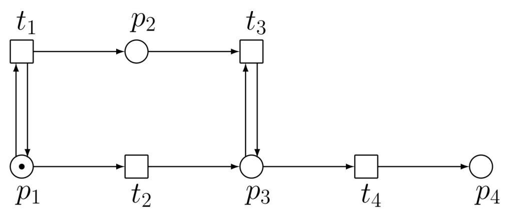


Figure 1. The given uncontrolled PN model

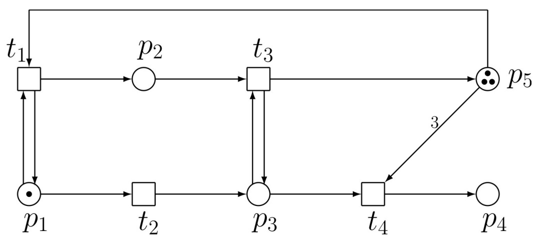


Figure 2. The trial to apply the ideal enforceability

has to be met. Consequently,

$$\mathbf{L} = (0,1,0,3), \quad \mathbf{b} = 3. \tag{23}$$

Because $\mathbf{x}_0 = (1,0,0,0)^T$, with respect to (4)–(7) we can obtain the supervisor (corresponding to the ideal enforcing) given in Figure 2 where the structure of the supervisor and its initial state are the following:

$$\mathbf{B}_s = -\mathbf{L}.\mathbf{B} = (-1,0,1,-3); \quad \mathbf{x}_s^0 = \mathbf{b} - \mathbf{L}.\mathbf{x}_0 = 3. \tag{24}$$

However, the condition (10) is not satisfied, because

$$-\mathbf{L}.\mathbf{B}_{o,uc} = -(0,1,0,3).(0,0,-1,1)^T = -3 \ngeq 0. \tag{25}$$

It means that the ideal enforceability is impossible. Consequently, the real enforceability approach has to be applied. Therefore, consider

$$\mathbf{R}_1 = (0,0,3,0); \quad R_2 = 1. \tag{26}$$

Then, because of (13), (14),

$$\mathbf{L}' = (0,1,3,3); \quad \mathbf{b}' = 3, \tag{27}$$

$$\mathbf{B}'_s = -\mathbf{L}'.\mathbf{B} = (-1,-3,1,0); \quad \mathbf{x}_s^0 = \mathbf{b}' - \mathbf{L}'.\mathbf{x}_0 = 3. \tag{28}$$

The supervised system is given in Figure 3. Here, the inequality (15) is fulfilled because

$$(\mathbf{R}_1 + \mathbf{R}_2\mathbf{L})\mathbf{B}_{o,uc} = \mathbf{L}'.\mathbf{B}_{o,uc} = (0,1,3,3).(0,0,-1,1)^T = 0 \qquad (29)$$

and it should be ≤ 0.

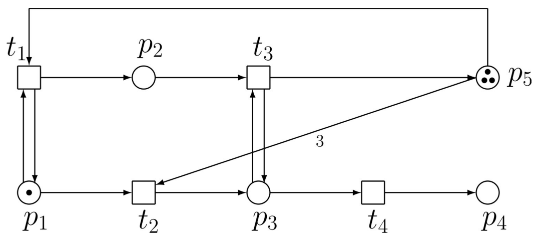


Figure 3. The real enforceability

The reachability trees of the three versions of the PN model structure are given in Figure 4.

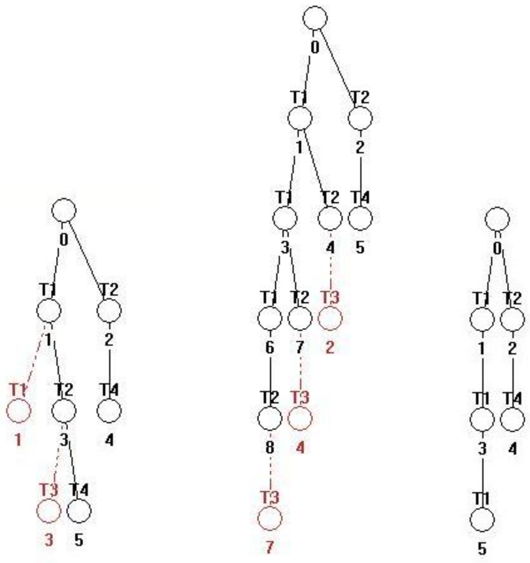


Figure 4. The reachability trees corresponding (from the left to the right) with the PN models in Figures 1, 2 and 3, respectively, with the corresponding reachability matrices $\mathbf{X}_a$, $\mathbf{X}_b$, $\mathbf{X}_c$

Corresponding nodes of these trees are represented by the columns of the reachability matrices as follows:

$$\mathbf{X}_a = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & \omega & 0 & \omega & 0 & \omega \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \tag{30}$$

where ω means unlimited number of tokens in the place p_2 in three reachable states, namely $\mathbf{x}_1$, $\mathbf{x}_3$ and $\mathbf{x}_5$, because of the self-loops in the RT nodes $\mathbf{x}_1$ and $\mathbf{x}_3$, and

$$\mathbf{X}_b = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 & 3 & 2 & 3 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 3 & 2 & 3 & 1 & 2 & 0 & 0 & 1 & 0 \end{pmatrix}, \tag{31}$$

$$\mathbf{X}_c = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 2 & 0 & 3 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 3 & 2 & 0 & 1 & 0 & 0 \end{pmatrix}. \tag{32}$$

2.1.4 Local Summary

In case where no uncontrollable and unobservable transitions occur, the controller can directly observe and prevent all transitions of PN model of the plant. However, it is much more realistic to abandon such an assumption in PN models of DES working in practice. Namely:

1. Transition t_j may be uncontrollable. It means, that the controller will not be able to directly prevent the transition from firing. In such a case, there will be no arc from any controller place to $t_j \in T$.
2. Transition $t_j \in T$ may be unobservable. It means, that the controller will not be able to directly detect the firing of the transition. Thus, the firing of t_j cannot affect the number of tokens in any controller place (i.e. the *feedback* from the plant to the controller is missing).

This implies that there are neither arcs from an unobservable transition t_j to any controller place p_i, $i = 1, \dots s$, $j = 1, \dots m$ nor the arc from any controller place p_i to t_j.

Hence, a transition being unobservable implies that it is also uncontrollable. Therefore, controllability of a transition implies its observability. Three different kinds of transitions are distinguished:

1. controllable;
2. uncontrollable but observable; and
3. uncontrollable and unobservable.

In such a case the enforceability of control interferences needs not always be ideal and the supervisor synthesis has to by modified – compare the pair (6), (7) with the pair (19), (20).

3 ERROR RECOVERY

As it was mentioned in the Section 1.4, many times errors (failures) occur during DES operation. For example in FMS a part may drop out from the robot gripper. The system has to detect what was wrong and recover the normal behaviour – i.e., to eliminate the influence of the error on the system behaviour. The error recovery is a set of actions that must be performed in order to return the system to its normal state. It is necessary

1. to synthesize the recovery sequence, and
2. to extend the scope of the controller activity in order to deal with the fault.

The problem is directly connected with reachability.

After the occurrence of the fault the system development is different than the standard one. Consider the situation after the occurrence of the fault $\mathbf{f}_j = (0,\dots,0,1,0,\dots,0)^T$ being the unit vector with dimensionality m (the number of transitions). Here $\mathbf{x}_k = \mathbf{x}_j + \mathbf{B}_f.\mathbf{f}_j$, where $\mathbf{B}_f$ is the structural matrix of the faulty system submodel.

In practice, there are many areas where it comes toward errors of different kinds. Let us illustrate two such areas – FMS and a transport system – namely, the robotized cell and the railroad crossing.

3.1 Error Recovery of a Kind of FMS

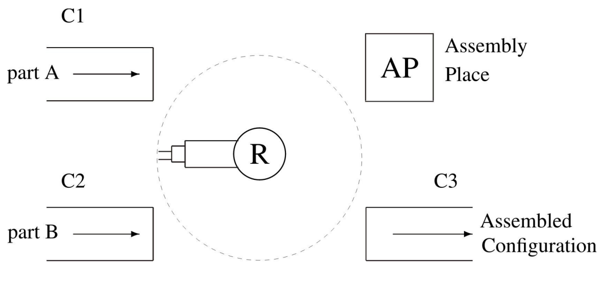


Figure 5. The scheme of the robotized assembly cell

Consider the robotized cell schematically displayed in Figure 5. Here, C1 represents a conveyor feeding parts of a kind A into the cell, C2 expresses a conveyor feeding parts of a kind B into the cell, and C3 pictures a conveyor carrying away the final product – i.e. the assembled part C (i.e. A + B) prepared in the assembly place AP – from the cell. The robot R plays the central role in the cell, because it serves all other devices inside the cell (i.e. C1–C3 and AP).

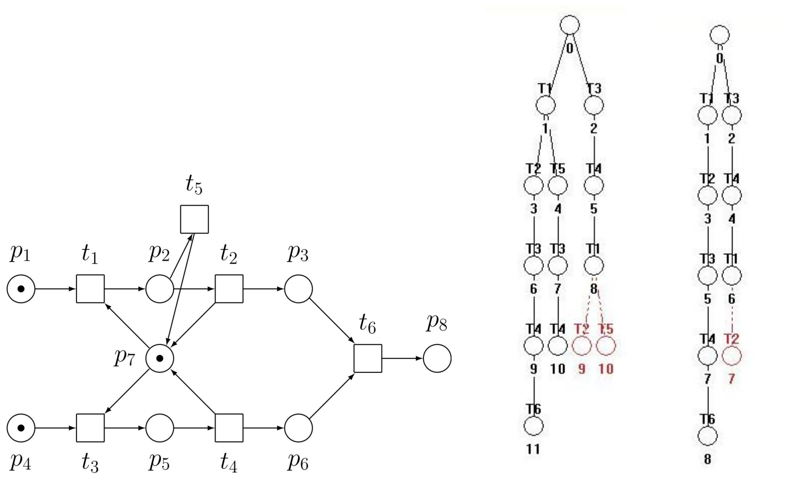


Figure 6. The PN model of the fragment of the plant (left), its full RT including firing of t_5 (middle) and the RT of the model without firing of t_5, i.e. without the occurrence of the error event (right)

The fragment of the global PN model of the robotized cell, displayed in Figure 6 left, models the handling of first two belts by R. The robot consecutively takes away parts A, B from two transport belts C1 and C2, respectively. In the PN model the supplying is realized by means of the places p_1 and p_4, respectively. The operations of taking parts are modelled by p_2, p_5, respectively. The parts A, B prepared to be inserted into the assembly place are modelled by places p_3, p_6, respectively. The robot puts them subsequently into AP modelled by p_8. Although the process is deterministic a fault can occur. The transition t_5 represents the error event – i.e. the fault when a part A drops out from the robot gripper. The fault itself is modelled by firing the transition t_5. The corresponding RT of the model segment, where the fault occurs, is displayed in the middle of the Figure 6. The RT corresponding to the normal situation, when no fault occur, is given in Figure 6 right. The state space of the system (the nodes of the full RT) are represented by the columns of the matrix (33):

$$\mathbf{X}_r = \begin{pmatrix} 1&0&1&0&0&1&0&0&0&0&0&0\\ 0&1&0&0&0&0&0&0&1&0&0&0\\ 0&0&0&1&0&0&1&0&0&1&0&0\\ 1&1&0&1&1&0&0&0&0&0&0&0\\ 0&0&1&0&0&0&1&1&0&0&0&0\\ 0&0&0&0&0&1&0&0&1&1&1&0\\ 1&0&0&1&1&1&0&0&0&1&1&1\\ 0&0&0&0&0&0&0&0&0&0&0&1 \end{pmatrix}. \tag{33}$$

Because the fault event is spontaneous, it cannot be prevented. There is only a possibility (if any) for trying to recover the system operation after occurring the fault, i.e. to remove the effects of the fault on the system operation. We must watch the consecutive states of the system after occurring the fault – i.e. the states $\mathbf{x}_4 = (0\,0\,0\,1\,0\,0\,1\,0)^T$ and $\mathbf{x}_{10} = (0\,0\,0\,0\,0\,1\,1\,0)^T$. From these states the recovery procedure has to be evolved. Note, that the first column of the matrix (33) is $\mathbf{x}_0$. Consequenly, $\mathbf{x}_4$ and $\mathbf{x}_{10}$ are displayed, respectively, as 5th and 11th columns of the matrix). However, the state $\mathbf{x}_{10}$ is inadmissible. Moreover, no further development is allowed from it, i.e. $\mathbf{x}_{10}$ has a form of a deadlock. Instead of $\mathbf{x}_{10}$ the recovered state $\mathbf{x}_9 = (0\,0\,1\,0\,0\,1\,1\,0)^T$ is expected. Also the states of PN without the fault – i.e. without firing t_5 – has to be taken into account in order to see the correct development. The form of the PN is the same like that on the left side of Figure 6, only the transition t_5 is not firing (i.e. as if missing). RT of such a PN model is given on the right side of the Figure 6. Its nodes are the columns of the matrix

$$\mathbf{X}_r = \begin{pmatrix} 1&0&1&0&1&0&0&0&0\\ 0&1&0&0&0&0&1&0&0\\ 0&0&0&1&0&1&0&1&0\\ 1&1&0&1&0&0&0&0&0\\ 0&0&1&0&0&1&0&0&0\\ 0&0&0&0&1&0&1&1&0\\ 1&0&0&1&1&0&0&1&1\\ 0&0&0&0&0&0&0&0&1 \end{pmatrix}. \tag{34}$$

A trial how the fault can be recovered in order to normalize the system behaviour is expressed by amending the PN model. The PN model able to recover the fault is given in Figure 7. The model covers both the normal behaviour and the recovery of the fault behaviour. Its RT is given in Figure 8. The trajectory of the system development when the fault occurred (i.e. when t_5 was fired) and then its entail was recovered is the following: $\mathbf{x}_0 \xrightarrow{t_1} \mathbf{x}_1 \xrightarrow{t_5} \mathbf{x}_5 \xrightarrow{t_3} \mathbf{x}_{11} \xrightarrow{t_4} \mathbf{x}_{18} \xrightarrow{t_7} \mathbf{x}_{25} \xrightarrow{t_8} \mathbf{x}_{31} \xrightarrow{t_1} \mathbf{x}_{35} \xrightarrow{t_2} \mathbf{x}_{38} \xrightarrow{t_6} \mathbf{x}_{39}$. Of course, also the sideways branch starting from $\mathbf{x}_5$: $\mathbf{x}_5 \xrightarrow{t_7} \mathbf{x}_{12} \ldots \xrightarrow{t_4} \mathbf{x}_{38}$ may be taken into account. However, the RT expresses also the trajectories of the system development where the firing of t_5 does not occur (i.e. when no fault occurs),

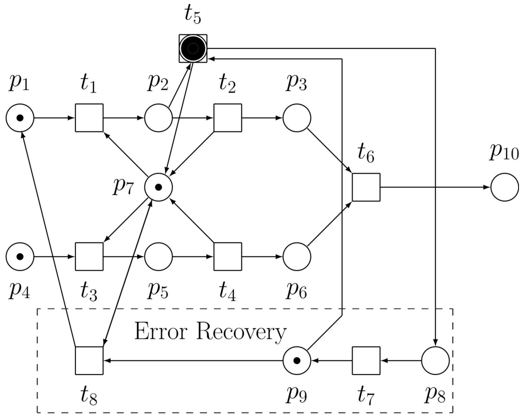


Figure 7. The PN model of the recovered system

e.g.: $\mathbf{x}_0 \xrightarrow{t_1} \mathbf{x}_1 \xrightarrow{t_2} \mathbf{x}_4 \xrightarrow{t_3} \mathbf{x}_9 \xrightarrow{t_4} \mathbf{x}_{15} \xrightarrow{t_6} \mathbf{x}_{22} \dots \xrightarrow{t_2} \mathbf{x}_{37}$ (together with its sideways branches). The RT nodes are represented by the columns of the matrix

$$\mathbf{X}_r = \begin{pmatrix}
1012001120100020010011010001100100000000 \\
0100000100000100100001000010010001010000 \\
0000100001100001110000001200001201020111 0 \\
1101110100101000100010001010000010000000 \\
0010000011010000010100000000100100000100 0 \\
0000001000000011100100101010001010011001 0 \\
1001111000101011001010111100100110100111 \\
0000010000001000000010000000000000000000 \\
1110101001001101000100100100000000000000 \\
0000000000000000000000100000010000100010 1
\end{pmatrix} . \qquad (35)$$

3.2 Error Recovery in a Segment of Transport Systems

Consider an example of the simple railroad crossing (RC). The RC gate prevents a direct contact of trains with vehicles on the road. The global PN model of RC given in Figure 9 consists of three cooperating sub-models expressing the behaviour of the

1. train,
2. crossing gate, and
3. control system.

The firing of the transition t_{f_5} models the occurrence of the failure(s). In general, the failure can occur more times. Thus, the marking of the place p_{14} (i.e. the

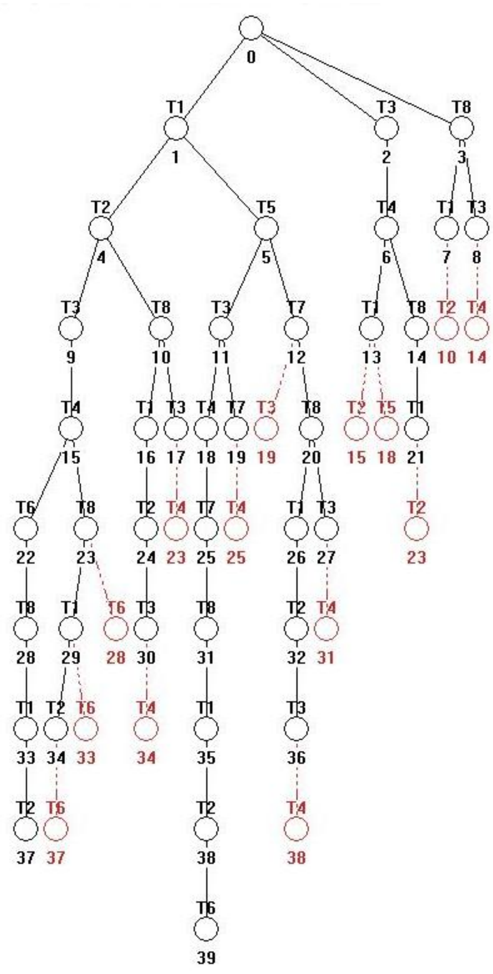


Figure 8. The RT of the PN model of the recovered system

number of its tokens) represents the number of the failure occurrences. What is very dangerous in doing so is that the firing of t_{f_5} involves an erroneous generation of a token in p_{l0} which directly influences the position of the barrier. Such an error may cause accidents. Many accidents allover the world has been caused due to such errors.

The following depicts the purport of places in the failure-free cases: As to the train, its states regarding the crossing are: p_1 = approaching; p_2 = being before; p_3 = being within; p_4 = being after. The states of the barrier are: p_{11} = up; p_{12} = down. The transitions t_6 and t_7 model, respectively, the events of raising and

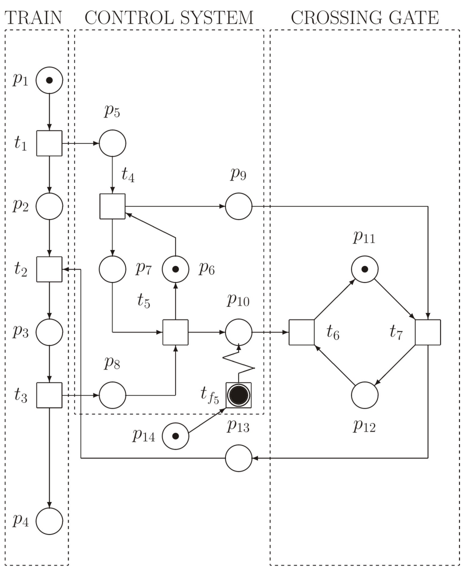


Figure 9. The PN model of the RC with the failure expressed by firing t_{f_5}

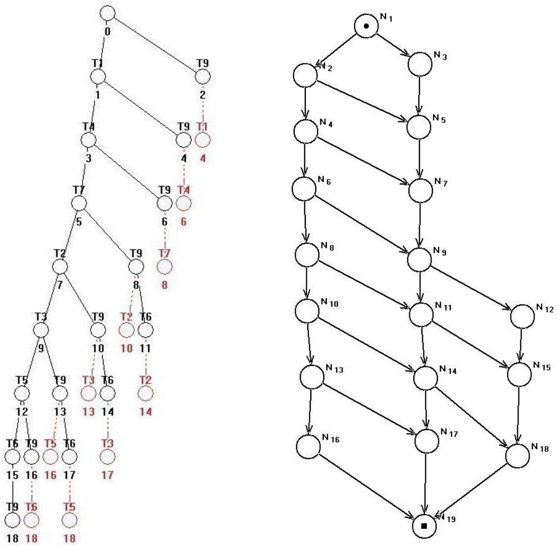


Figure 10. The RT of the PN model of RC with only one possible occurrence of the failure represented by t_{f_5} (left), and the corresponding RG (right)

lowering the barrier. The states of the control system are: $p_5 - p_{10}$. The place p_{13} models the interlock. Being active (having token) it gives the warning signal for the train – the alert that the barrier is still up.

Very dangerous (critical as to safety reasons) is the situation when the barrier is going up, and simultaneously, the failure t_{f_5} occurs. For detection of such situation the redundant information is needed. The control system must issue such information. It can be seen that the places p_6 and p_7 in the control system correspond to p_{11} and p_{12} in the real crossing gate. If p_7 and p_{11} are active simultaneously, a contradiction between the fault situation (real) and the standard situation (normal, error free) is detected. For the error recovery it is necessary to set what state is accepted to be the true one. Supposing that the barrier is up and drops down, the recovery is realized by means of the transition t_{r_1}.

Considerably simpler situation occurs when the barrier is up and none train is approaching. Namely, by means of the transition t_{r_2} the fail signal p_{10} from the control system in a close touch with the activity of the place p_{11} ensures that the fail signal can be practically ignored.

The RT and RG of the failed system are given in Figure 10. The particular nodes of RT/RG are the columns of the reachability matrix

$$\mathbf{X}_{reach} = \begin{pmatrix} 1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,1\,0\,1\,1\,1\,1\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,1\,1\,1 \\ 0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,1\,0\,1\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1 \\ 0\,0\,0\,1\,0\,1\,1\,1\,1\,1\,1\,1\,0\,1\,1\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,1\,0\,0\,0\,1\,0 \\ 0\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0 \\ 0\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,1\,0\,0\,2\,0\,1 \\ 1\,1\,1\,1\,1\,0\,1\,0\,0\,0\,0\,1\,0\,0\,1\,1\,0\,1\,1 \\ 0\,0\,0\,0\,0\,1\,0\,1\,1\,1\,1\,0\,1\,1\,0\,0\,1\,0\,0 \\ 0\,0\,0\,0\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0 \\ 1\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,0\,1\,0\,0\,1\,0\,0\,0 \end{pmatrix}. \tag{36}$$

The PN model of the system with the recovered error is given in Figure 11, while the RT and RG are displayed in Figure 12.

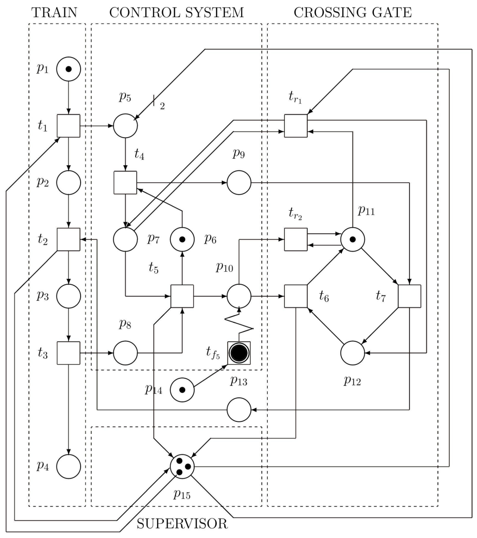


Figure 11. The PN model of the system with the recovered failure and with the supervisor removing the deadlock occurring during the process of the failure recovery

The RT/RG nodes corresponding to such model are represented by the columns of the reachability matrix

$$\mathbf{X}^s_{reach} = \begin{pmatrix}
1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&1&1&0&1&1&1&0&1&1&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&1&0&0&0&0&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&1&1&1&1&1&1&1\\
0&1&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&0&1&1&0&0&1&0&0&0&0&0&0&0&1&0&0&0&1&1&0&0&1&1&1\\
0&0&0&1&0&0&1&1&0&1&1&1&1&1&1&1&0&1&1&1&0&0&1&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&1&1&0&0&0\\
0&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&1&0&1&0&0&1&0&0&1&0&0&1&0&0&1&1&0&0&0&2&0&0&1&1&0\\
1&1&1&1&1&1&0&1&1&0&0&1&0&0&1&0&0&0&1&0&1&0&1&0&1&0&1\\
0&0&0&0&0&0&1&0&0&1&1&0&1&1&0&1&1&1&0&1&0&1&0&1&0&1&0\\
0&0&0&0&0&0&1&0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&1&0&0&1&0&0&1&0&0&1&0&0&0&1&0&0&0&1&0&0&0&0&0&0\\
-&-\\
3&2&3&0&2&3&0&0&2&1&0&0&1&1&1&0&2&1&2&1&3&2&2&1&3&2&3
\end{pmatrix} \tag{37}$$

where the last row represents the states (marking) of the supervisor in particular RT/RG nodes. More details concerning the multiplicity of the error occurrence are discussed in [4].

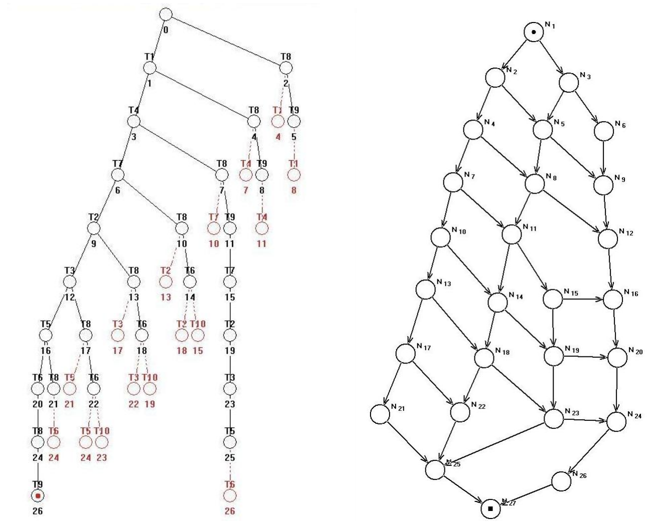


Figure 12. The RT of the recovered system with removed deadlock by means of the supervisor (left) and the corresponding RG (right)

4 CASE STUDY ON CONTROL OF COMPLETE FMS BY MEANS OF IPN

Having resolved the error recovery of the FMS fragment in Section 3.1, let us apply the IPN-based approach to control the complete FMS including the error. The uncontrolled PN model of the plant is the expanded form of the fragment given in Figure 6 with the same fault, of course. The IPN-based control of it (before the error recovering) is presented in Figure 13.

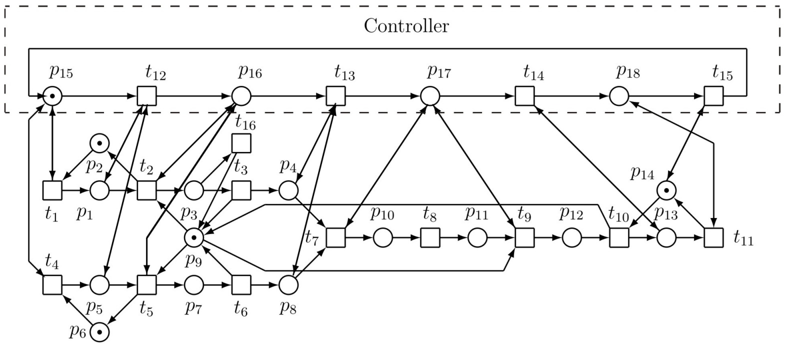


Figure 13. The PN model of the controlled system

The RT of the PN model is given in Figure 14. To save a space, it is turned by 90 degrees to the left.

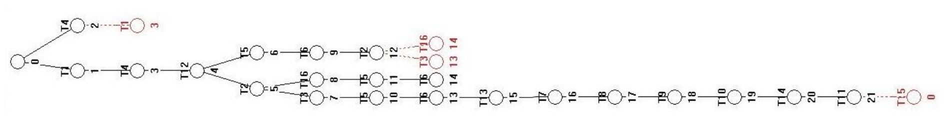


Figure 14. RT of the PN model of the controlled system

The reachability set, i.e. the particular nodes of the RT in the form of the state vectors are represented by means of the columns of the following matrix

$$\mathbf{X}_r = \begin{pmatrix}
0&1&0&1&1&0&1&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&1&0&1&1&0&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&1&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&1&0&0&1&0&0&1&0&1&0&0&0&0&0&0\\
0&0&1&1&1&1&0&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&1&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1\\
0&0&0&0&0&0&1&0&0&0&1&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&1&1&1&1&0&0&0&0&0&0\\
1&1&1&1&1&0&0&1&1&1&0&0&0&1&1&1&1&1&0&1&1&1\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&1\\
1&1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&1
\end{pmatrix}. \tag{38}$$

4.1 Error Recovery

Let us recover the fault analogically to the approach presented in Figure 7. As we can see in Figure 14, the occurrence of the fault represented by the transition t_{16} (when it is firing) tends to the deadlock state $\mathbf{x}_{14}$ (see the column 15 of the matrix (38) because RT node N1 corresponds to $\mathbf{x}_0$) presented by active places p_2, p_6, p_8, p_9, p_{14}, p_{16}, all of them with marking equal to 1. To prevent such situation the recovery must tend to the elimination of the deadlock state $\mathbf{x}_{14}$ (caused by firing t_{16}) and simultaneously recover the system operation.

A trial directed towards the error recovery proposal is given in Figure 15. The RT of the system is given in Figure 16 while the corresponding complete RG of the recovered system representing all system trajectories is displayed in Figure 17.

There exist four production cycles – i.e. the trajectories/paths starting from $\mathbf{x}_0$ and ending also in $\mathbf{x}_0$ – in the RG of the recovered system displayed in Figure 17. Their lengths are 15, 22, 29, and 30 steps (a step represents the firing of a transition). They can be analyzed separately. These cycles include the internal sub-cycles (sub-trajectories).

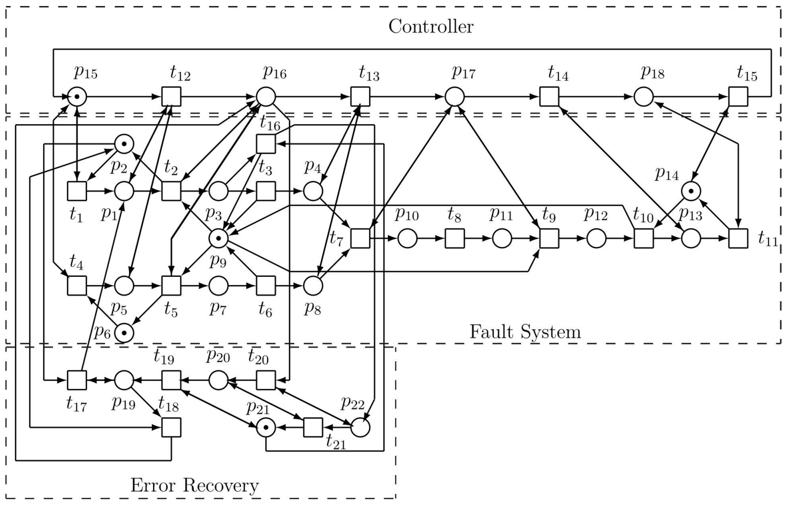


Figure 15. The PN model of the controlled system able to recover the failure

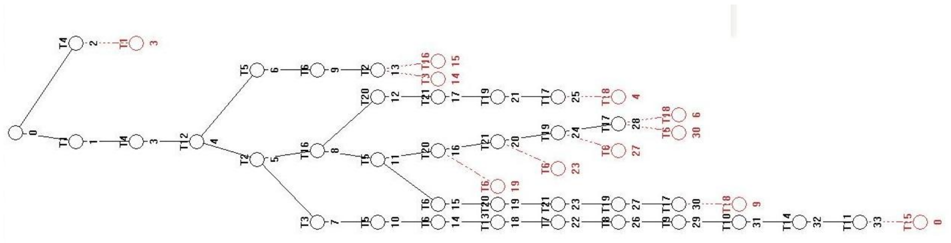


Figure 16. The RT of the PN model of the controlled system with the recovered failure

Let us analyze in detail the shortest cycle with 15 steps – it is given in Figure 18. It contains four sub-cycles (sub-trajectories). None of them contains a deadlock. Particular steps in the trajectories are realized by successive firing of transitions in corresponding sequences. They realize the transition from a state to another (next)

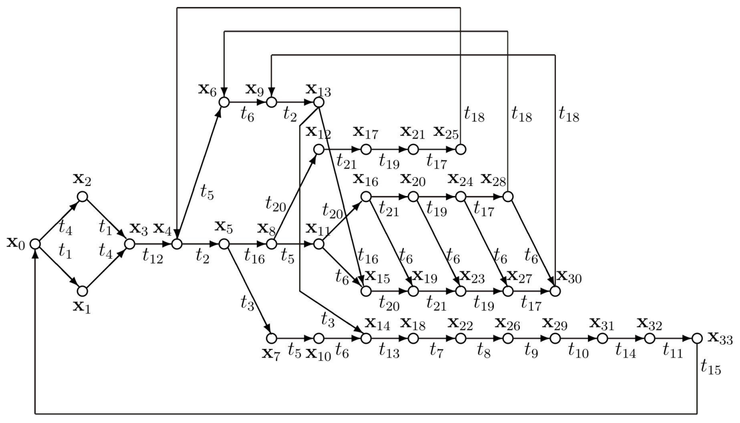

x0 x1 x2 x3 x4 x5 x6 x7 x8 x9 x10 x11 x12 x13 x14 x15 x16 x17 x18 x19 x20 x21 x22 x23 x24 x25 x26 x27 x28 x29 x30 x31 x32 x33
t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16 t17 t18 t19 t20 t21

Figure 17. The corresponding RG of the recovered system

state in the trajectories.

$$\mathbf{X}_r = \begin{pmatrix}
0101101001000000000000000100101000 \\
1010010110111111111111111011010111 \\
0000010000000100000000000000000000 \\
0000000100100010001000000000000000 \\
0011110110001000010001000100000000 \\
1100001001110111101110111011111111 \\
0000001000110000100010001000100000 \\
0000000001000111001100010001001000 \\
1111100111001011011101110111001111 \\
0000000000000000000000100000000000 \\
0000000000000000000000000010000000 \\
0000000000000000000000000000010000 \\
0000000000000000000000000000000110 \\
1111111111111111111111111111111001 \\
1111000000000000000000000000000000 \\
0000111111110111000000000000000000 \\
0000000000000000001000100010010100 \\
0000000000000000000000000000000011 \\
0000000000000000000001001101101000 \\
0000000000001000110110010000000000 \\
1111111101100110011011111111111111 \\
0000000010011001100100000000000000
\end{pmatrix} \tag{39}$$

Let as analyze the shortest path in details. As we can see from the RT/RG, as a matter of fact the trajectory (path) aggregates four sub-trajectories:

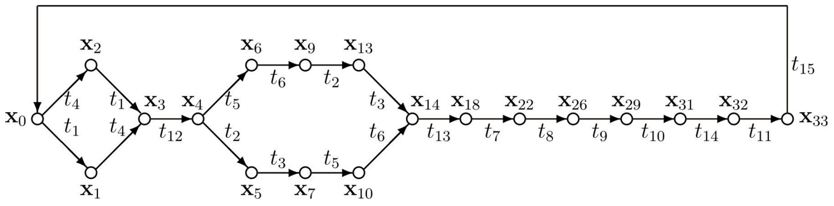


Figure 18. The 15 steps cycle in the RG of the recovered system

1. $\mathbf{x}_0 \xrightarrow{t_4} \mathbf{x}_2 \xrightarrow{t_1} \mathbf{x}_3 \xrightarrow{t_{12}} \mathbf{x}_4 \xrightarrow{t_5} \mathbf{x}_6 \xrightarrow{t_6} \mathbf{x}_9 \xrightarrow{t_2} \mathbf{x}_{13} \xrightarrow{t_3} \mathbf{x}_{14} \xrightarrow{t_{13}} \mathbf{x}_{18} \xrightarrow{t_7} \mathbf{x}_{22} \xrightarrow{t_8}$ $\mathbf{x}_{26} \xrightarrow{t_9} \mathbf{x}_{29} \xrightarrow{t_{10}} \mathbf{x}_{31} \xrightarrow{t_{14}} \mathbf{x}_{32} \xrightarrow{t_{11}} \mathbf{x}_{33} \xrightarrow{t_{15}} \mathbf{x}_0$;
2. $\mathbf{x}_0 \xrightarrow{t_4} \mathbf{x}_2 \xrightarrow{t_1} \mathbf{x}_3 \xrightarrow{t_{12}} \mathbf{x}_4 \xrightarrow{t_2} \mathbf{x}_5 \xrightarrow{t_3} \mathbf{x}_7 \xrightarrow{t_5} \mathbf{x}_{10} \xrightarrow{t_4} \mathbf{x}_{14} \xrightarrow{t_{13}} \mathbf{x}_{18} \xrightarrow{t_7} \mathbf{x}_{22} \xrightarrow{t_8}$ $\mathbf{x}_{26} \xrightarrow{t_9} \mathbf{x}_{29} \xrightarrow{t_{10}} \mathbf{x}_{31} \xrightarrow{t_{14}} \mathbf{x}_{32} \xrightarrow{t_{11}} \mathbf{x}_{33} \xrightarrow{t_{15}} \mathbf{x}_0$;
3. $\mathbf{x}_0 \xrightarrow{t_1} \mathbf{x}_1 \xrightarrow{t_4} \mathbf{x}_3 \xrightarrow{t_{12}} \mathbf{x}_4 \xrightarrow{t_5} \mathbf{x}_6 \xrightarrow{t_6} \mathbf{x}_9 \xrightarrow{t_2} \mathbf{x}_{13} \xrightarrow{t_3} \mathbf{x}_{14} \xrightarrow{t_{13}} \mathbf{x}_{18} \xrightarrow{t_7} \mathbf{x}_{22} \xrightarrow{t_8}$ $\mathbf{x}_{26} \xrightarrow{t_9} \mathbf{x}_{29} \xrightarrow{t_{10}} \mathbf{x}_{31} \xrightarrow{t_{14}} \mathbf{x}_{32} \xrightarrow{t_{11}} \mathbf{x}_{33} \xrightarrow{t_{15}} \mathbf{x}_0$;
4. $\mathbf{x}_0 \xrightarrow{t_1} \mathbf{x}_1 \xrightarrow{t_4} \mathbf{x}_3 \xrightarrow{t_{12}} \mathbf{x}_4 \xrightarrow{t_2} \mathbf{x}_5 \xrightarrow{t_3} \mathbf{x}_7 \xrightarrow{t_5} \mathbf{x}_{10} \xrightarrow{t_4} \mathbf{x}_{14} \xrightarrow{t_{13}} \mathbf{x}_{18} \xrightarrow{t_7} \mathbf{x}_{22} \xrightarrow{t_8}$ $\mathbf{x}_{26} \xrightarrow{t_9} \mathbf{x}_{29} \xrightarrow{t_{10}} \mathbf{x}_{31} \xrightarrow{t_{14}} \mathbf{x}_{32} \xrightarrow{t_{11}} \mathbf{x}_{33} \xrightarrow{t_{15}} \mathbf{x}_0$.

Other longer cycles have 22 (with 18 sub-trajectories), 29 (with 32 sub-trajectories), and 30 (with 16 sub-trajectories) steps. The last of them represents practically two consecutive shortest trajectories (i.e. two successive cycles without a failure). While any of the sub-trajectories of the shortest cycle introduced above does not contain the transition t_{16} representing the failure, the sub-trajectories of the longer cycles (namely with 22 steps and 29 steps) do. In spite of this they operate correctly, because of the successful error recovery. It means that the cycle consisting of 15 steps represents the normal operation of the controlled plant, i.e., the situations when no failure occurs. However, the longer cycles are able to deal with the failure (occurring event represented by firing t_{16}) successfully.

Unfortunately, there is not a sufficient space here for a graphical presentation of the trajectories, or all their sub-trajectories. Nevertheless, they can be traced from Figure 17. For illustration, let us introduce them at least in the aggregated form in Figure 19 (up) and Figure 19 (down). They show that the system is able to deal with the failure – i.e., the error recovery was successful.

4.2 Local Summary

The application of IPN for modelling and control of DES with nondeterminism represented by uncontrollable and/or unobservable transitions was presented in this section. IPN seems to be a suitable, sound and relatively simple approach for resolving such tasks, especially technical ones. In Section 5 the application of LbPN will be presented pointing out the difference from IPN.

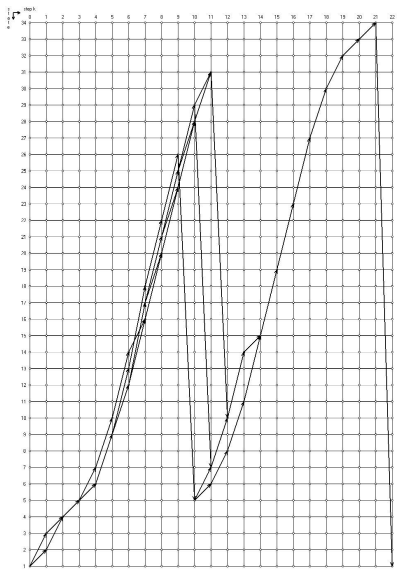
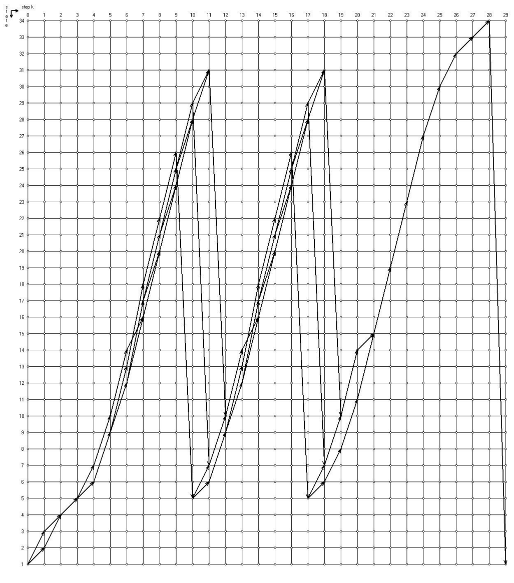



Figure 19. The aggregated form of the path with 22 steps (up) and 29 steps (down). In both cases on the horizontal axes the numbers of steps are displayed, while on the vertical axes the numbers of state vectors (shifted by +1) are displayed – namely, the RG nodes N_{i+1} corresponds to $\mathbf{x}_i$, $i = 0, 1 \ldots$

5 CASE STUDY ON CONTROL OF FMS BY MEANS OF LBPN

Consider the plant (a robotic cell) schematically displayed in Figure 20. There are two production lines producing two different kinds of final products. The plant consists of four machines (M1–M4), four robots (R1–R4), one automatically guided vehicle (AGV) system, one buffer (B) with the finite capacity, two input transport belts I1 and I2 feeding the cell, respectively, by parts of a kind Pa1 and of a kind Pa2. Finally, the output transport belts O1 and O2 export, respectively, the processed final parts Fp1, Fp2 from the cell. A similar production plant was used for another reason in [5], based on [26].

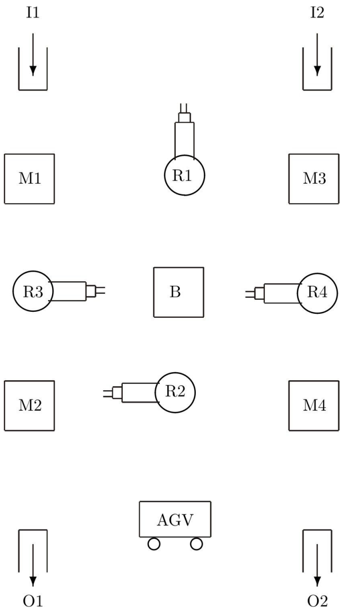


Figure 20. The scheme of the plant

While in an ideal P/T PN model all transitions are considered to be controllable and no faults occur in the plant, in the model of a real plant none of those presumptions holds true. However, when there occur uncontrollable transitions and even some errors, the situation changes. Therefore, let us use the LbPN model given in Figure 21.

The particular sections of the model are denoted by framed badges corresponding to individual devices.

The marking of the place p_1 represents the number of parts (in general α) of the type Pa1 entering the production line PL1 (the left column of devices in Figure 20), while the marking of the place p_{16} represents the number of parts (in general β) of the type Pa2 entering the production line PL2 (the right column of devices in

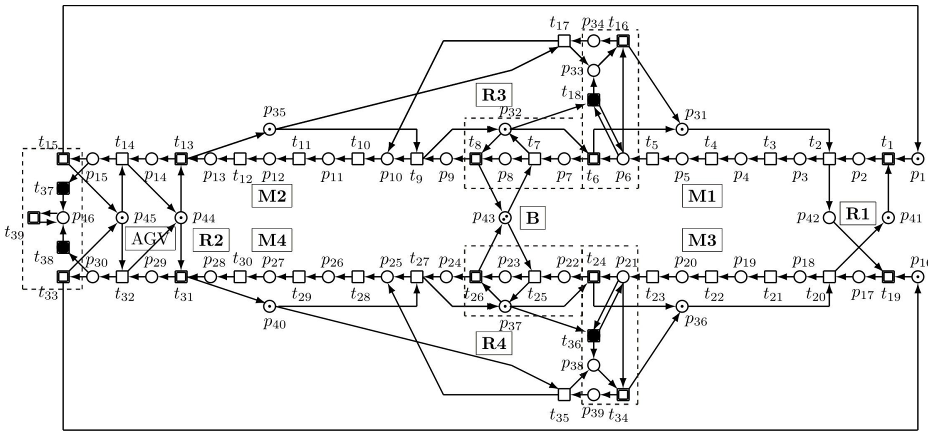


Figure 21. The LbPN model of the real plant with unobservable/uncontrollable transitions

Figure 20). Devices in the central column in Figure 20 – R1, R2 as well as B and AGV – are common for both production lines. The robot R3 belongs solely to PL1 and R4 solely to PL2.

Formally said, p_1 and p_{16} are fed, respectively, by means of I1 and I2. On the other hand t_{15} and t_{33} feed, respectively, O1 with final products Fp1 and O2 with final products Fp2. The marking of the place p_{43} expresses the number of free slots in B (they should be two minimally).

Places from the set $\{p_{33}, p_{34}, p_{38}, p_{39}, p_{46}\}$ represent the *faulty behavior*, i.e. errors. They are marked only if a fault has occurred. Transitions $t_{18}, t_{36}, t_{37}, t_{38}$, filled in the black color, model (by means of their firing) occurrence of the failures. Segments of the PN model containing those faults are demarcated by means of the dashed rectangles with the vertical dimension being greater than the horizontal dimension. The transition t_{18} models a fault of the robot R3 that moves a part from the output buffer of the machine M1 to the input buffer of the machine M2 instead of putting it into the buffer B. In the like manner t_{36} models a fault in the robot R4 that moves a part from the output buffer of the machine M3 to the input buffer of the machine M4 instead of to insert it into the buffer B. Finally, t_{37}, t_{38} model a fault in the AGV. When AGV is working correctly, a completely finished part exits from the robotic cell and a new part enters AGV. If a fault occurs in AGV, parts are not exiting the production lines. Hence, they cannot be replaced by new input parts.

In the model we can see three kinds of transitions:

1. *observable transitions* t_1, t_6, t_8, t_{13}, t_{15}, t_{19}, t_{24}, t_{26}, t_{31}, t_{33}, t_{16}, t_{34}, t_{39}. They are drawn by means of thick lines. The dashed rectangles with the greater width than their height are added in order to cover also t_8 along with t_7 and t_{26} along with t_{25};

2. *unobservable* but *regular transitions* t_2, t_3, t_4, t_5, t_7, t_9, t_{10}, t_{11}, t_{12}, t_{14}, t_{20}, t_{21}, t_{22}, t_{23}, t_{25}, t_{27}, t_{28}, t_{29}, t_{30}, t_{32}. Using the mark ε established in [3] (the Part 1 of this paper), the transitions may be renamed as ε_2, ε_3, ε_4, ε_5, ε_7, ε_9, ε_{10}, ε_{11}, ε_{12}, ε_{14}, ε_{20}, ε_{21}, ε_{22}, ε_{23}, ε_{25}, ε_{27}, ε_{28}, ε_{29}, ε_{30}, ε_{32}, respectively. They are drawn normally, i.e. by means of thin lines;
3. *fault transitions* t_{18}, t_{36}, t_{37}, t_{38}. They are filled in by black colour.

The LbPN model has 46 places and 39 transitions. Because of the great number of places and transitions, and rather complicated structure, the RT/RG may have the extensive number of nodes (i.e. reachable states of the model from a given initial state) depending on the initial state $\mathbf{x}_0$. At the initial state displayed in Figure 21 – i.e., when each of the places p_1, p_{41}, p_{16}, p_{31}, p_{36}, p_{32}, p_{37}, p_{35}, p_{40}, p_{44}, p_{45} possesses one token and p_{43} possesses two tokens – the number of nodes is 1640. The RT was computed in Matlab using the toolbox SPNBOX elaborated in [21]. Consequently, neither RT nor RG can be displayed here because of insufficient space.

It is not easy to compute the RT/RG of such large dimensionality because of long computational time, sometimes also for the limited capacity of the computer memory, especially in case of the initial state containing more entering parts – represented by number of tokens α, β stationed, respectively, in p_1, p_{16} – and/or the higher capacity of the buffer B represented by the number of tokens stationed in p_{43}.

Suppose that each robot is equipped with a sensor. Thus, it always can be observed whether the robot grasps a part (e.g. from a belt) and/or inserts it (e.g. into a machine or B) or not. Taking [3] into account, in LbPN the term $\mathcal{L} = L \cup \varepsilon$ is an alphabet representing a finite set of events, where L represents observable events and ε represents unobservable events. In our case in particular, we can set for observable transitions the following:

1. for R1 $\mathcal{L}(t_1) = a$ and $\mathcal{L}(t_{19}) = e$;
2. for R2 $\mathcal{L}(t_{13}) = c$ and $\mathcal{L}(t_{31}) = l$;
3. for R3 $\mathcal{L}(t_6) = \mathcal{L}(t_8) = \mathcal{L}(t_{16}) = b$;
4. for R4 $\mathcal{L}(t_{24}) = \mathcal{L}(t_{26}) = \mathcal{L}(t_{34}) = g$;
5. providing that it is possible to observe each time when a part is moved by the AGV, also $\mathcal{L}(t_{15}) = \mathcal{L}(t_{33}) = \mathcal{L}(t_{39}) = d$.

The robot R1 always starts taking one part from the first production line. Having denoted (see above) the number of tokens in p_1 as α, while the number of tokens in p_{16} as β, we can analyze the LbPN model behaviour in detail.

All cases in which $\alpha = 0$ present only one node corresponding to M_0. Moreover, all cases in which $\beta = 0$ present 19 nodes in RG.

For $\alpha \neq 0$ and $\beta \neq 0$ many of states occur in RG. For example for initial state where $p_1 = 1$, $p_{16} = 1$, $p_{31} = 1$, $p_{32} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{39} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$, 592 nodes occur in RG. However, it is valid for cases without failures.

When the failures occur in the model, the situation is changed. For example when $p_1 = 1$, $p_{16} = 0$, $p_{31} = 1$, $p_{32} = 1$, $p_{33} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{38} = 1$, $p_{39} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$, $p_{46} = 1$ the number of RG nodes is 1266. When $p_1 = 0$, $p_{16} = 1$, $p_{31} = 1$, $p_{32} = 1$, $p_{33} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{38} = 1$, $p_{39} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$, $p_{46} = 1$ the number of RG nodes is 19.

However, in case when $p_1 = 0$, $p_{16} = 1$, $p_{31} = 1$, $p_{32} = 1$, $p_{33} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{37} = 1$, $p_{38} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$ the number of RG nodes is 10, for $p_1 = 1$, $p_{16} = 0$, $p_{31} = 1$, $p_{32} = 1$, $p_{33} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{37} = 1$, $p_{38} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$ the number of RG nodes is 516.

But, for $p_1 = 1$, $p_{16} = 1$, $p_{31} = 1$, $p_{32} = 1$, $p_{33} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{37} = 1$, $p_{38} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$ the number of RG nodes is 2 904, for $p_1 = 2$, $p_{16} = 1$, $p_{31} = 1$, $p_{32} = 1$, $p_{33} = 1$, $p_{35} = 1$, $p_{36} = 1$, $p_{37} = 1$, $p_{38} = 1$, $p_{40} = 1$, $p_{41} = 1$, $p_{43} = 2$, $p_{44} = 1$, $p_{45} = 1$ the number of RG nodes is 20 356.

For greater α, β the numbers of RG nodes reach tens thousands, even hundred thousands.

Because PL1 is perfectly symmetric with PL2, the number of RG nodes does not change by altering α with β.

Local Summary. In [22] the observation of events is considered as outputs in most problem settings, such as state estimation and fault diagnosis. Some authors, e.g. [18, 20], consider an extended LbPN model enriched with state observations. It is assumed there that the token content in some places of the net is measurable. Such understanding of the extension of LbPN is named as Interpreted PN (IPN). It is suitable especially for technical applications, as shown in the Section 4.

However, in [20] we can see that such a type of LbPN can always be converted into an equivalent standard LbPN by a suitable re-definition of the transition labels, and hence the LbPN-based models and the IPN-based models have the same modeling power. In spite of this it is interesting and useful to utilize the IPN models along with the LbPN models. In general, the scope of LbPN abilities seems to be a little wider than that of IPN because LbPN are suitable also for the fault diagnosis and state estimation (as it was pointed out in this Section). Although both of the approaches are very suitable for analysing and modelling DES containing nondeterminism (as it was demonstrated above), the successful applicability of both kinds of PN (i.e. IPN and LbPN) in the supervisor synthesis may be limited by the complexity of the modelled plant (and especially by the consecutive complexity of its PN model). Namely, the so called curse of dimensionality well known in respect of the Bellman's dynamic programming [2] is, unfortunately, in force also in many other cases including the computation and analysis of RT/RG in PN at the supervisor synthesis. This is generally true for the PN-based control synthesis utilizing RT/RG and it has nothing to do with the convenience of the IPN and LbPN applicability to DES with nondeterminism.

In case of the estimation of the LbPN marking, the *representative markings* and corresponding *representative marking graphs* are defined [12] in order to find the *consistent markings*. The consistent marking set consists of all markings that are reachable from initial marking by firing some sequences whose observation is a word (a sequence of transitions being consecutively fired). In [13] the PN reachability problem is analyzed by means of defining the basis marking. There, the set T of PN transitions is partitioned into two subsets – the explicit and implicit transitions. The subset of implicit transitions does not contain directed cycles. The reachability set obtained by firing of implicit transitions is created by a subset of reachable markings called basis makings. Consequently, the basis reachability graph (BRG) can be obtained by means of the efficient algorithm presented just in [13]. It is suitable for bounded PN and after an extension also for unbounded PN. For unbounded PN the newest approach to computing basis coverability graph (BCG) is presented in [10]. Such approaches help to reduce an enormity (hugeness) of RT/RG sometimes also exponentially – see [10].

To avoid the problem with the huge RT/RG at the DES control synthesis by means of PN, also the *theory of regions* – see e.g. [16, 8, 9, 1, 23, 17, 14] – can be used. Consider the labelled transition system given as a tuple $(S, \mathcal{L}, T_{\rightarrow})$, where S is a set of states, $\mathcal{L}$ is a set of labels and $T_{\rightarrow}$ is a set of labelled transitions. A region r of such transition system is a mapping assigning to each state $s \in S$ a number $\sigma(s)$ (natural number for P/T PN, binary number for some event structures) and to each transition label ℓ a number $\tau(\ell)$ such that consistency condition $\sigma(s') = \sigma(s) + \tau(\ell)$ holds whenever $(s, \ell, s') \in T_{\rightarrow}$. So, a region r corresponds to a place of the Petri net which we would like to associate with a given step transition system. Consequently, the PN structure is, of course, enriched in such a way. This makes possible to find a more favourable structure in comparison with the original one. However, this topic is out of the scope of this paper.

6 ILLUSTRATIVE EXAMPLE OF ANOTHER KIND OF NONDETERMINISM IN FMS

Let us introduce, exclusively for illustration, another situation in complicated FMS leading to the nondeterminism. It concerns the problems occurring at sharing resources in a plant during cooperation of its subsystems. In FMS such systems are named as Resource Allocation Systems (RAS).

There are frequently used the $\underline{s}$ystems of $\underline{s}$imple $\underline{s}$equential $\underline{p}$rocesses with $\underline{r}$esources (S^{3}PRs) in FMS. In such systems the part being produced uses only one copy of one resource at each processing step. Such systems create a subclass of a higher (upper) class S*PRs [24, 7] where more copies of one resource are allowed. Consider the production system with the principal scheme given in Figure 22 performing the production routing with the scheme given in Figure 23.

Here, M1–M4 are four machine tools each using some of the four sets of cutting/surfacing tools h1–h4. Three robots R1–R3, being the crucial devices, serve

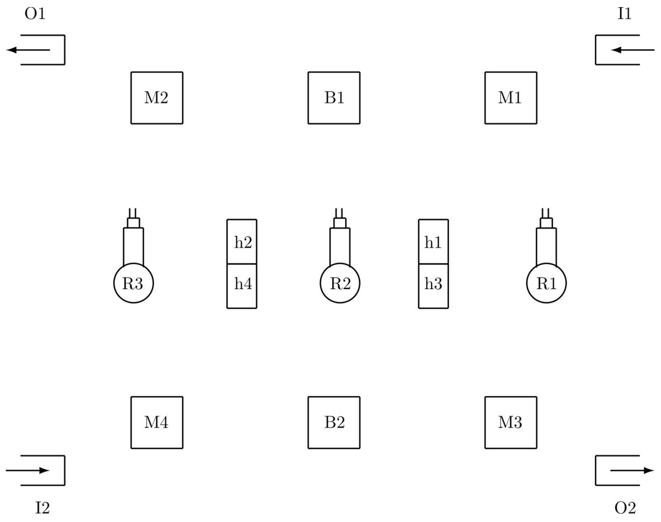


Figure 22. The scheme of the RAS plant

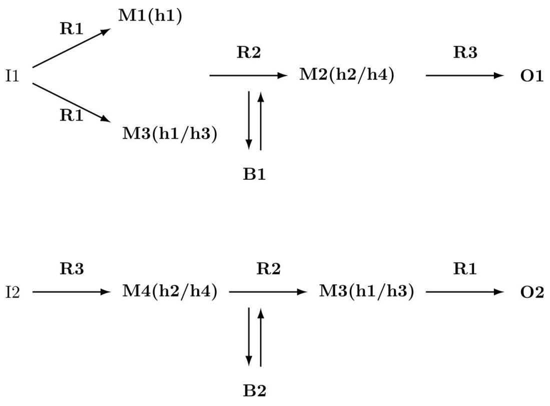


Figure 23. The production routing scheme of the RAS plant

these machines as well as buffers B1, B2, the input and output belts I1, I2 and O1, O2. The symbol Mi(hj/hk) in the routing scheme in Figure 23 means that the machine Mi uses the tools from the sets hj and hk, namely at first the tool from hj and then the tool from hk.

In the PN terminology, the resource(s) can be understood in a wider sense – e.g., in our case the instantaneous availability of a machine (because it can be either idle, i.e. available for an interested device, or not), cutting/surfacing tool, robot, buffer, etc.

The principle of PN model of a resource can be illustrated by means of Figure 24. In general, two or more production lines working as parallel processes sharing common resources, bring problems of different kinds to be solved, especially deadlocks because of a lack of resources in one or more of the processes. The simplest case of a resource is illustrated in Figure 24 left, while two simple parallel processes with common resource(s) are given in Figure 24 right. The parallel running of these processes is impossible when $n = 1$ while it is possible when $n > 1$.

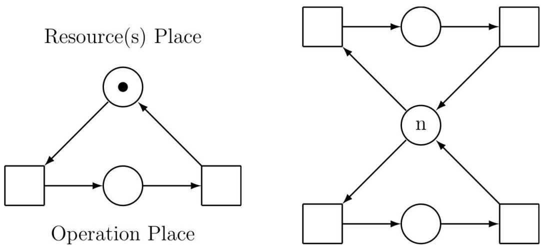


Figure 24. Resorce(s). In case of S^{3}PRs $n = 1$ (left). In case of S*PRs $n > 1$ (right). Here, two parallel processes share the common resource(s). When $n = 1$ the simultaneous using of the resource by the upper process and lower one is mutually excluded. However, for $n > 1$ both processes can run simultaneously.

The PN model of the RAS plant is displayed in Figure 25.

In general, the symbol $p_x \models$ Dy means that a place p_x models the activity of a device Dy. Then, in the PN model the following relations between PN places and the FMS devices are actual: $p_6 \models$ M1, $p_9 \models$ B1, $p_{10} \models$ M2, $p_{17} \models$ B2, $p_{19} \models$ M4, $p_{21} \models$ R1, $p_{22} \models$ h1, $p_{23} \models$ h3, $p_{24} \models$ M3, $p_{25} \models$ R2, $p_{26} \models$ h4, $p_{27} \models$ h2, $p_{28} \models$ R3.

This model corresponds to the situation when all resources are reliable. At the given initial state $\mathbf{x}_0 = (0,0,0,0,0,0,0,0,0,0,2,4,0,2,10,4,0,2,10,1,2,2,2,0,1,2,2,1,0)^T$ there exist 63 469 reachable states – e.g. the last of them is $\mathbf{x}_{63439} = (0,0,2,1,1,1,0,0,2,0,1,3,0,2,1,3,2,3,0,0,0,0,0,0,0,0,0,0,2)^T$. For that reason, neither RT/RG nor the reachability matrix (containing state vectors being RT/RG nodes as its columns) cannot be introduced here.

In case of the unreliable resources the situation is more complicated. The PN model of such a case is given in Figure 27. Here, in comparison with Figure 25 five sub-nets represented by the dashed boxes are added. Namely, any unreliable resource has to be equipped by an additional place which represents waiting for the recovery of the resource failure. The scheme of such a sub-net is given in Figure 26. The dashed boxes in Figure 27 represent just such sub-nets. Of course, the model itself is not able to deal with such a nondeterminism. The recovery consists in the renewal of the state of resources.

As it follows from Figure 26 and especially from the dashed sub-nets in Figure 27, no transition having p_u as its input place (i.e. $t_9, t_{12}, t_{13}, t_{25}, t_{27}$) cannot be fired during the occurrence of the rest of the corresponding breakdown. Any supervisor being synthesized for such a PN model has to respect this rule. Namely, the token in p_w (i.e. p_2, p_6, p_{12}, p_{20}, p_{29}) cannot be returned to p_u (i.e. p_3, p_5, p_{13}, p_{16}, p_{28})

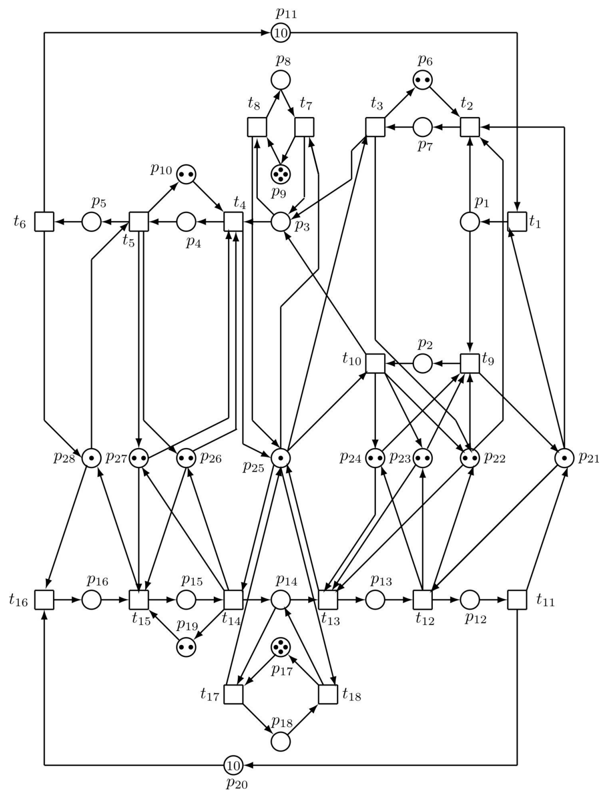


Figure 25. The PN model of the RAS plant with multiple resources equipped with the waiting places P_w, $w = 2, 6, 12, 20, 29$ waiting for the error recovery of unreliable resources

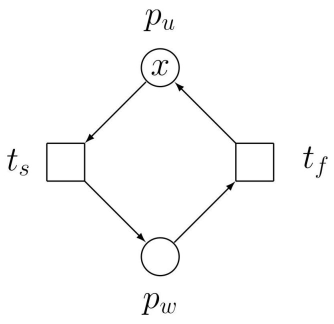


Figure 26. Unreliable resource x is represented by the place p_u, while the place p_w represents the process of waiting for the recovery. Transitions t_s and t_f represent, respectively, the start and finish of the waiting.

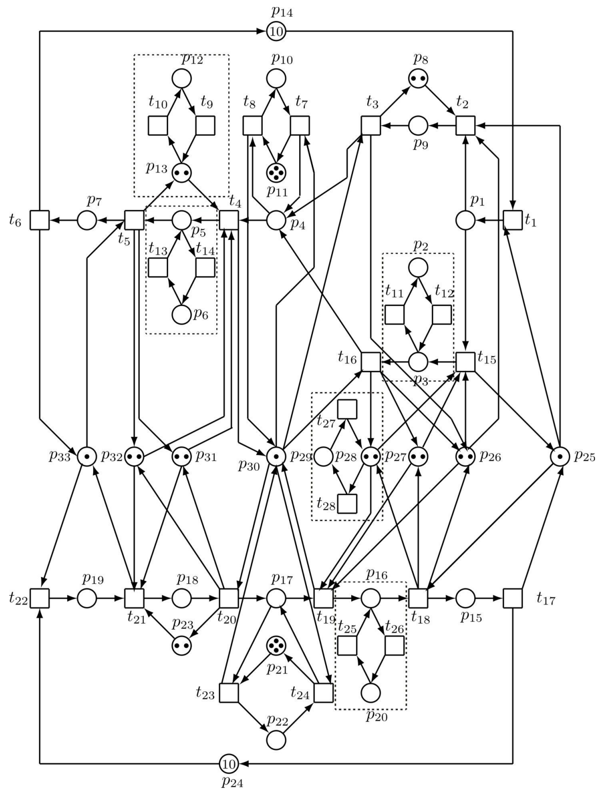


Figure 27. The PN model of the plant with multiple resources

until all faults related to p_u are corrected. The corrections have to be performed by error recovery subsystems.

Local Summary. The plant (FMS of a kind RAS) being the DES consists of a set of versatile resources i.e. machine tools, robots, buffers, cutting/surfacing tools, etc. Usually, there is a limited number of such resources shared by the plant sub-processes. Consequently, a lot of deadlocks can originate for this reason. Of course, deadlocks in FMS are undesirable, and highly unfavorable. Namely, the entire plant or at least a part of it remains stagnate and the primal intention of the production cannot be achieved. Such a situation can also be understood to be a form of nondeterminism, which is very unsafe. The recovery of resource failures is very complicated in this case. A fault-tolerant supervisor able to handle resource failures has to be used in order to resolve deadlocks, e.g. the Banker's algorithm [6] or its newer modifications. Another approaches to removal of deadlocks can be used as well – e.g. the supervisor based on PN siphons [25, 11, 19].

The error recovery of this kind of FMS (i.e. ARS) is out of the focus of this paper. However, dealing with such a problem may be an idea for further research in the DES containing nondeterminism.

7 CONCLUSION

This paper is a continuation of the paper [3]. It represents the Part 2 of the paper [3] being the Part 1. Both parts are inseparable components of one topic. While in [3] the different kinds of PN (especially P/T PN, timed PN, controlled PN, labelled PN, interpreted PN) were described and their applicability for DES modelling, analysis and control were indicated, the Part 2 brings more examples and case studies from PN-based modelling FMS and transport systems with nondeterminism.

The main aim of the article was to apply the results of article [3] particularly on more kinds of DES with the nondeterminism of different types. Some essential preliminaries were introduced at the beginning. Then, the particular topics were discussed. At first, the nondeterminism resulting from unobservable/uncontrollable transitions and/or unobservable (unmeasurable) places of PN-based models was dealt with. The ideal and real enforceability of control interferences were confronted in the example. After that, the problem of the error recovery was analyzed and two case studies on error recovery in real systems were introduced – namely, the case of the segment of the robotic cell of FMS and the segment of the transport system (the railroad crossing). Next, the case study on the error recovery of the complete robotic cell by means of IPN-based model was introduced. Afterward, the case study on the error recovery of another complex FMS by means of the LbPN-based model was presented. After all, the example of the specific kind of nondeterminism in the special kind of FMS – RAS (Resource Allocation Systems) was introduced for illustration.

The case studies and examples bear witness to applicability of PN in general, and especially IPN and LbPN, on dealing with nondeterminism in PN models of DES.

Because the local summaries were introduced at the end of the particular sections, they need not to be repeated here.

Acknowledgement

The author thanks for the partial support of the VEGA Agency (under Grant No. 2/0029/17).

REFERENCES

[1] Badouel, E.—Darondeau, P.: Theory of Regions. In: Reisig, W., Rozenberg, G. (Eds.): Lectures on Petri Nets I: Basic Models. Advanced Course on Petri Nets

(ACPN 1996). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 1491, 1998, pp. 529–586, doi: 10.1007/3-540-65306-6_22.

[2] Bellman, R. E.: Dynamic Programming. Princeton University Press, 1957. Republished 2003.

[3] Čapkovič, F.: Petri Nets at Modelling and Control of Discrete-Event Systems Containing Nondeterminism – Part 1. Computing and Informatics, Vol. 37, 2018, No. 5, pp. 1258–1292, doi: 10.4149/cai_2018_5_1258.

[4] Čapkovič, F.: Failures in Discrete-Event Systems and Dealing with Them by Means of Petri Nets. Vietnam Journal of Computer Science. Vol. 5, 2018, No. 2, pp. 143–155, doi: 10.1007/s40595-018-0110-3.

[5] Cabasino, M. P.—Giua, A.—Seatzu, C.: Fault Detection for Discrete Event Systems Using Petri Nets with Unobservable Transitions. Automatica, Vol. 46, 2010, No. 9, pp. 1531–1539, doi: 10.1016/j.automatica.2010.06.013.

[6] Dijkstra, E. W.: Selected Writings on Computing: A Personal Perspective. Springer Verlag, 1982, doi: 10.1007/978-1-4612-5695-3.

[7] Farooq, A.—Huang, H.—Wang, X.-L.: Petri Net Modeling and Deadlock Analysis of Parallel Manufacturing Processes with Shared-Resources. Journal of Systems and Software, Vol. 83, 2010, No. 4, pp. 675–688, doi: 10.1016/j.jss.2009.11.705.

[8] Ghaffari, A.—Rezg, N.—Xie, X.: Net Transformation and Theory of Regions for Optimal Supervisory Control of Petri Nets. IFAC Proceedings Volumes, Vol. 35, 2002, No. 1, pp. 443–448, doi: 10.3182/20020721-6-es-1901.00076.

[9] Ghaffari, A.—Rezg, N.—Xie, X.: Design of a Live and Maximally Permissive Petri Net Controller Using the Theory of Regions. IEEE Transactions on Robotics and Automation, Vol. 19, 2003, No. 1, pp. 137–141, doi: 10.1109/tra.2002.807555.

[10] Lefaucheux, E.—Giua, A.—Seatzu, C.: Basis Coverability Graph for Partially Observable Petri Nets with Application to Diagnosability Analysis. In: Khomenko, V., Roux, O. (Eds.): Applications and Theory of Petri Nets and Concurrency (PETRI NETS 2018). Springer, Cham, Lecture Notes in Computer Science, Vol. 10877, 2018, pp. 164–183, doi: 10.1007/978-3-319-91268-4_9.

[11] Liu, G. Y.—Barkaoui, K.: A Survey of Siphons in Petri Nets. Information Sciences, Vol. 363, 2016, pp. 198–220, doi: 10.1016/j.ins.2015.08.037.

[12] Ma, Z.—Tong, Y.—Li, Z.—Giua, A.: Marking Estimation in Labelled Petri Nets by the Representative Marking Graph. 20th IFAC World Congress, Toulouse, France, July 2017. IFAC PapersOnLine, Vol. 50, 2017, No. 1, pp. 11175–11181, doi: 10.1016/j.ifacol.2017.08.1240.

[13] Ma, Z. Y.—Tong, Y.—Li, Z.—Giua, A.: Basis Marking Representation of Petri Net Reachability Spaces and Its Application to the Reachability Problem. IEEE Transactions on Automatic Control, Vol. 62, 2017, No. 3, pp. 1078–1093, doi: 10.1109/TAC.2016.2574120.

[14] Meis, B.—Bergenthum, R.—Desel, J.: Synthesis of Elementary Net Systems with Final Configurations. Proceedings of the 16th International Conference on Application of Concurrency to System Design (ACSD 2016), Torun, Poland, 2016. CEUR Workshop Proceedings, Vol. 1592, 2016, pp. 47–57, available online at: `http://ceur-ws.org/Vol-1592/paper04.pdf`.

[15] Moody, J. O.—Antsaklis, P. J.: Petri Net Supervisors for DES with Uncontrollable and Unobservable Transitions. IEEE Transactions on Automatic Control, Vol. 45, 2000, No. 3, pp. 462–476, doi: 10.1109/9.847725.

[16] Mukund, M.: Petri Nets and Step Transition Systems. International Journal of Foundations of Computer Science, Vol. 3, 1992, No. 4, pp. 443–478, doi: 10.1142/s0129054192000231.

[17] Pan, Y.-L.—Huang, Y.-S.—Jeng, M.—Chung, S.-L.: Enhancement of an Efficient Control Policy for FMSs Using the Theory of Regions and Selective Siphon Method. The International Journal of Advanced Manufacturing Technology, Vol. 66, 2013, No. 9–12, pp. 1805–1815, doi: 10.1007/s00170-012-4460-1.

[18] Ramirez-Trevino, A.—Rivera-Rangel, I.—Lopez-Mellado, E.: Observability of Discrete Event Systems Modeled by Interpreted Petri Nets. IEEE Transactions on Robotics and Automation, Vol. 19, 2003, No. 4, pp. 557–565, doi: 10.1109/tra.2003.814503.

[19] Row, T.-C.—Pan, Y.-L.: Maximally Permissive Deadlock Prevention Policies for Flexible Manufacturing Systems Using Control Transition. Advances in Mechanical Engineering, Vol. 10, 2018, No. 7, pp. 1–10, doi: 10.1177/1687814018787406.

[20] Ru, Y.—Hadjicostis, C. N.: Fault Diagnosis in Discrete Event Systems Modeled by Partially Observed Petri Nets. Discrete Event Dynamic Systems, Vol. 19, 2009, No. 4, pp. 551–575, doi: 10.1007/s10626-009-0074-7.

[21] SPNBOX: A Toolbox for the Supervisory Control of Petri Nets. Available at: `http://mviordache.name/abs/spnbox/`.

[22] Tong, Y.—Li, Z.—Giua, A.: Observation Equivalence of Petri Net Generators. IFAC Proceedings Volumes, Vol. 47, 2014, No. 2, pp. 338–343, doi: 10.3182/20140514-3-fr-4046.00060.

[23] Uzam, M.: An Optimal Deadlock Prevention Policy for Flexible Manufacturing Systems Using Petri Net Models with Resources and the Theory of Regions. The International Journal of Advanced Manufacturing Technology, Vol. 19, 2002, No. 3, pp. 192–208, doi: 10.1007/s001700200014.

[24] Yue, H.—Xing, K.—Hu, H.—Wu, W.—Su, H.: Petri-Net-Based Robust Supervisory Control of Automated Manufacturing Systems. Control Engineering Practice, Vol. 54, 2016, pp. 176–189, doi: 10.1016/j.conengprac.2016.05.009.

[25] Li, Z. W.—Zhou, M. C.: Elementary Siphons of Petri Nets and Their Application to Deadlock Prevention in Flexible Manufacturing Systems. IEEE Transactions on System, Man, and Cybernetics – Part A: System and Humans, Vol. 34, 2004, No. 1, pp. 38–51, doi: 10.1109/tsmca.2003.820576.

[26] Zhou, M. C.—DiCesare, F.: Petri Net Synthesis for Discrete Event Control of Manufacturing Systems. Kluwer Academic Publishers, 1993, doi: 10.1007/978-1-4615-3126-5.

František Čapkovič received his Master degree in 1972 from the Faculty of Electrical Engineering of the Slovak Technical University, Bratislava, Slovakia. Since 1972 he has been working at the Slovak Academy of Sciences (SAS), Bratislava, in 1972–1991 at the Institute of Technical Cybernetics, in 1991–2001 at the Institute of Control Theory and Robotics and since 2001 till now he has worked at the Institute of Informatics. In 1980 he received the Ph.D. degree from SAS. In 1998 he was appointed Associate Professor. His interests are in the area of modelling, analysing and intelligent control of discrete-event systems (DES) and hybrid systems. He is the author of more than 240 publications.