Computing and Informatics, Vol. 37, 2018, 269–290, doi: 10.4149/cai_2018_2_269

META-PROGRAMMING AND POLICY-BASED DESIGN AS A TECHNIQUE OF ARCHITECTING MODULAR AND EFFICIENT DSP ALGORITHM IMPLEMENTATIONS

Ireneusz Gawlik

Department of Electronics, AGH University of Science and Technology
Kraków, Poland
e-mail: `igawlik@agh.edu.pl`

Szymon Pałka, Tomasz Pędzimąż, Bartosz Ziółko

Department of Electronics, AGH University of Science and Technology
Kraków, Poland
&
Techmo, Kraków, Poland
e-mail: {`pszymon, pedzimaz, bziolko`}`@agh.edu.pl`

Abstract. Meta-programming paradigm and policy-based design are less known programming techniques in Digital Signal Processing (DSP) community, used to coding in pure C or assembly language. Major software components, like C++ STL, have proven usefulness of such paradigms in providing top performance of highly optimised native code, along with abstraction and modularity necessary in complex software projects. This paper describes composition of DSP code using these techniques, bringing as an example implementation of Feedback Delay Network (FDN) artificial reverberation algorithm. The proposed approach was proven to be practical, especially in case of prototyping computationally intense algorithms. To provide further performance insight, we discuss the techniques in context of other optimisation methods, like Single Instruction Multiple Data (SIMD) instruction sets usage and exploitation of superscalar architecture capabilities.

Keywords: C++, low level optimisations, policy-based design, template meta-programming, SIMD, FDN

1 INTRODUCTION

Adjusting signal processing algorithms to achieve desired results, while conforming to strict performance restrictions, is a challenge of many DSP designers. We have faced similar issues during our work on acoustical signal processing code, that had to satisfy both top performance requirement (real-time processing of hundreds of audio streams using consumer grade hardware) and high level of subjective sound quality [11].

Most of the high performance DSP programming is done in assembly or C language (especially in the embedded/low-energy systems). Along with project size, using procedural paradigm leads to unmanageable code, which lacks composability and is error prone. While code complexity problem is mostly addressed by composing program structure in object oriented manner (using languages like C++, C#, Java, etc.), such solution may lead to severe performance impact, particularly in case of abstracting small, loosely-coupled code blocks. In some cases of DSP code, it means only few arithmetic operations per function call. Similarly, architecting modular code in C language involves function pointer management, that, in fact, is similar to what happens behind the scenes during virtual method calls in the object oriented languages like C++ or Java. Negative performance impact of virtual method calls is directly connected to the modern pipelined CPU architectures [12], memory access bottleneck [15] and function call overhead [6] (that cannot be *inlined* by compiler).

Common approaches to DSP programming may be divided into following classes:

- Assembly programming, usually targeted for dedicated DSP processors.
- C/C++ programming, with the use of popular DSP libraries [8, 27].
- Dedicated, usually functional domain specific languages (DSL), such as Faust [19], SuperCollider [16], ChucK [33, 32] or Kronos [18]. These languages are usually targeted for audio processing and sound synthesis.

As domain specific languages are not targeted for general purpose DSP programming, and assembly coding is error prone, nonportable, and unsuitable for complex problems, we focus on comparing our approach to the traditional C/C++ programming with use of optimised DSP libraries.

High performance DSP libraries, like Intel IPP [27] or FFTW [8] are widely available. Most of them provide implementations for some of the frequently used DSP building blocks, like discrete transforms, Finite Impulse Response (FIR) and Infinite Impulse Response (IIR) filtering, etc. There are also a specialised solutions, developed to automatically generate highly optimised machine code for many integral transforms [22]. While such primitives cover plethora of applications, not all problems may be solved efficiently using these functionalities, as shown in the following sections.

In this paper, we propose an approach to solve the problem of contradictory requirements, namely code *performance* and *composability*. Both of these requirements can be efficiently satisfied using template based meta-programming, implemented in C++ language. We also address *readability*, which is often a concern in the case of template meta-programming. This work is organised as follows: Section 2 provides an introduction to programming concepts used in proposed solution, namely template-metaprogramming and policy-based design. Section 3 motivates the use of proposed technique in DSP programming, describes general concepts and provides examples of simple algorithm implementations, along with considerations about SIMD instruction set usage. Section 4 describes an example of the modular implementation of a complex algorithm on the case of Feedback Delay Network reverberator. Discussed implementation has been applied in beam-tracing audio engine for computer games [35], proving its applicability in a real world multimedia problems. Section 5 reviews related work and possible further enhancements. Section 6 concludes.

Although all examples shown in this paper were tested on current x86-64 processor architectures [12], described concepts apply to any processor architecture, including specialised DSP cores and GPGPU units [5].

2 BACKGROUND

2.1 Template Metaprogramming

As C++ language evolved [28, 20], the standard committee introduced generic programming mechanisms to allow creating efficient and reusable code in fully type-safe manner. The C++ template system has been proven to be Turing-complete [31]. Therefore, any computation, that is computable in principle, may be expressed in form of templates and executed during compile time [1].

Template metaprogramming techniques have been presented in numerous publications [1, 3, 30, 26]. When used properly, they become very powerful tools for highly optimised code generation during compile time [30]. As an example, we present an implementation of fast Walsh-Hadamard transform based on metaprogramming. Typically, highly optimised transform libraries like FFTW provide dedicated, assembly coded routines for small transform sizes [9]. As many fast transform algorithms express a recursive nature, they are fairly easy to implement using template meta-programming, which results in assembly closely matching optimised code written by a skilled programmer.

```
template<size_t stride,
         size_t offset,
         typename value_type,
         size_t order>
inline typename
  std::enable_if<stride != 0, void>::type
fwht(std::array<value_type, order>& vector)
{
  for (size_t i = 0; i < stride; ++i)
    butterfly(vector[offset + i],
      vector[offset + i + stride]);
  fwht<stride / 2, offset, value_type,
    order>(vector);
  fwht<stride / 2, offset + stride, value_type,
    order>(vector);
}

template<size_t stride,
         size_t offset,
         typename value_type,
         size_t order>
inline typename
  std::enable_if<stride == 0, void>::type
fwht(std::array<value_type, order>& vector)
{} // terminate template recursion
```

Listing 1. Template meta-programming based FWHT implementation

Code listing 1 presents template meta-programming based implementation of fast Walsh-Hadamard transform [2]. Algorithm is implemented in straightforward, recursive manner, using divide and conquer methodology. Such approach results in highly expressive code, that is readable despite meta-programming specific code parts. The advantage of meta-programming implementation lies in fact, that compiler is able to inline all recursive calls, because the recursion depth is known at compile time. There is no need to apply more complex, loop based solutions. The compiler is also able to unroll *for* loop inside *fwht* template instances, as the number of iterations is also defined at compile time. As this example shows, guiding the compiler to generate efficient code is as simple as providing appropriate parameters in template class instantiation. Presented technique is applicable whenever transform size can be determined during compile time (which is usually the case).

Loop unrolling may be also defined *explicitly* using meta-programming, as shown in code listing 2. Depending on compiler used, such technique may yield better results. However, its usefulness needs to be validated by measuring execution time, as latest code optimisations manuals point out that loop unrolling may lead to micro-op cache issues [7]. In such case, partial unrolling (e.g. groups of 2 or 4 consecutive iterations) may be applied.

```
  template <size_t index,
            size_t offset,
            size_t stride,
            typename value_type,
            size_t order>
  inline typename
  std::enable_if<index != 0, void>::type
unroll(std::array<value_type, order>& vector)
{
  unroll<index - 1, offset, stride,
    value_type, order>(vector);
  butterfly(vector[offset + index - 1],
    vector[offset + index - 1 + stride]);
}

template <size_t index,
          size_t offset,
          size_t stride,
          typename value_type,
          size_t order>
inline typename
  std::enable_if<index == 0, void>::type
unroll(std::array<value_type, order>& vector)
{} // terminate unrolling
```

Listing 2. Template meta-programming based loop unrolling

In presented implementation, coupled with explicit loop unrolling, single call of specialised *fwht* function results in assembly code free of any function calls as well as jump instructions.

Code listing 3 shows assembly for 4-point fast Walsh-Hadamard assembly code, generated by MSVC 11.0 (Microsoft Visual Studio 2012 Update 4) from discussed function templates. Such implementation may leverage superscalar capabilities through Out Of Order (OOO) execution, present in all modern processors.

```
vmovss      xmm1,dword ptr [rbx]
vaddss      xmm0,xmm1,dword ptr [rbx+8]
vmovss      dword ptr [rbx],xmm0
vsubss      xmm1,xmm1,dword ptr [rbx+8]
vmovss      dword ptr [rbx+8],xmm1
vmovss      xmm2,dword ptr [rbx+4]
vaddss      xmm0,xmm2,dword ptr [rbx+0Ch]
vmovss      dword ptr [rbx+4],xmm0
vsubss      xmm1,xmm2,dword ptr [rbx+0Ch]
vmovss      dword ptr [rbx+0Ch],xmm1
vmovss      xmm3,dword ptr [rbx]
vaddss      xmm0,xmm3,dword ptr [rbx+4]
vmovss      dword ptr [rbx],xmm0
vsubss      xmm1,xmm3,dword ptr [rbx+4]
vmovss      dword ptr [rbx+4],xmm1
vmovss      xmm2,dword ptr [rbx+8]
vaddss      xmm0,xmm2,dword ptr [rbx+0Ch]
```

```
    vmovss          dword ptr [rbx+8],xmm0
    vsubss          xmm1,xmm2,dword ptr [rbx+0Ch]
    vmovss          dword ptr [rbx+0Ch],xmm1
```

Listing 3. Generated 4-point FWHT machine code

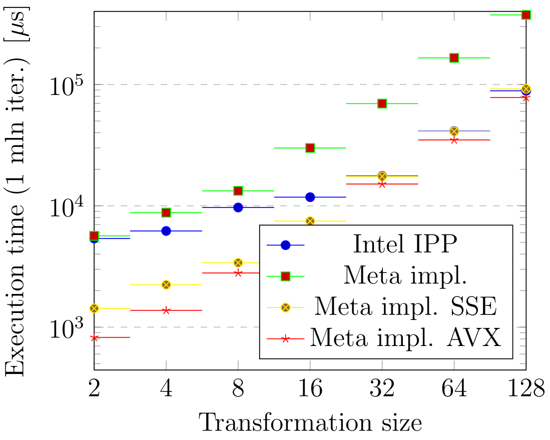


Figure 1. Comparison of fast Walsh-Hadamard transform execution time, depending on implementation used. Meta-programming based implementation is described in code listing 1. All compiler optimisations were enabled, including general optimisation (02 flag), inlining all suitable functions (Ob2 flag) and favouring fast code over code size (Ot flag). Test was performed on hardware featuring Intel Core i7-3770K 3.5 GHz processor.

As shown in Figure 1, code presented in listing 2 provides performance which is on a par with optimised Intel IPP routines in case of AVX vectorised implementation (IPP also uses AVX instruction set). In fact, proposed solution exhibits smaller execution overhead per each FWHT iteration.

2.2 Policy-Based Design

Policy-based design is a programming paradigm introduced by Andrei Alexandrescu in [3]. Its goal is to mimic the behaviour of the strategy design pattern [10], but using only static, compile time meta-programming techniques. That means that the only restriction added to strategy pattern requirements, is the need to fully determine the behaviour of the configurable component at compile time. This requirement enables policies to be an effective method of class customisation, as long as behaviour of this class may be defined at compile time.

The use of policy-based design involves providing a set of policies by passing them to the class template as template arguments. Such policies may provide static methods, type definitions or be a super-classes of the specialised class template.

Policy-based design, however, may require some experience to be implemented correctly. Most importantly, all of the policies need to be designed mutually orthogonal, so that replacing one does not affect behaviour of any of the others. Moreover, the designer may be unable to use template specialisations with different policies interchangeably (without any user code modifications), as policy-based design may involve class interface changes, i.e., when the specialised template class inherits from the policy class.

3 ENCODING DSP BLOCK DIAGRAMS USING POLICY BASED DESIGN

This section explains how metaprogramming techniques and policy-based design can be employed to aid DSP programming. We describe how to encode algorithms based on block diagrams with predefined components, using specific examples.

3.1 Motivation

Ease of experimentation with different DSP designs is the primary motivation to apply policy-based design and meta-programming methods. Because code generated using these techniques can be highly optimised during compilation phase, programmers are enabled to test both correctness and performance of the given design.

Modular design involves decomposition of the DSP diagrams into small, interchangeable components, thus embracing code reusability. As mentioned in previous sections, typical object oriented design entails virtual method calls. Such methods, when used frequently, can have negative performance impact. This is visible particularly in case of functions that contain only a few numerical operations. In contrast to classical object oriented desing, C++ template based approach allows to achieve similar level of modularity, if only polymorphic behaviour may be determined during compile time. Static polymorphism allows for code decomposition into small, individual pieces, with negligible performance impact, because compiler has much more freedom to perform code optimisations. Therefore, DSP applications, that usually struggle for top performance, benefit from the proposed approach.

3.2 General Concepts

Code composition based on DSP diagrams requires direct mapping of block elements and connections between them into template classes. Primarily, consistent naming convention needs to be enforced, as appropriate naming acts as an interface between all elements in DSP diagram. For example, in languages like C++, that allow for operator overloading, DSP blocks may be represented as *functors* (listing 4).

```
template<typename input_block>
class processing_block
```

```
{
public:
  output_type operator()(void)
  {
    output_type out = _input();
    // signal processing code
    return out;
  }
private:
  input_block _input;
};
```

Listing 4. Static variant of decorator pattern applied in DSP processing block

Each DSP block may perform on different pairs of input/output types. Therefore, heterogenic fixed-point/floating-point processing is enabled and various numerical precision representations may be used. Also, arrays of samples may be processed, if only it is necessary. Ability of combining arbitrary signal processing primitives, as long as connections between blocks match the input/output type pair, is the key feature that makes the proposed technique suitable for encoding DSP diagrams. Moreover, type-safety checks performed during compilation help to ensure correctness of built processing pipelines.

3.2.1 Sequential/Parallel DSP Blocks as Heterogenic Containers

As each DSP block needs to be represented in form of separate type, collections of blocks (connected either sequentially or in parallel) may be represented in following ways:

- Pairs of blocks, analogous to pair containers in most languages.
- Lists of blocks, implemented similarly to tuples.
- Type policy based containers, discussed later in this section.

Representing connections by pairs provides the same level of expressiveness as other solutions. Motivation to provide other ways of connecting blocks lies strictly in *ease of programming* and *readability*.

All DSP block connectors need to exhibit consistent interface in order to act itself as DSP building block (e.g. need to be implemented as functors). Code listing 5 provides example implementations of pair connectors, both for serial and parallel sum connection.

```
template<class first_block, class second_block>
class parallel_sum_pair
{
 public:
 output_t operator()(input_t input)
 {
   return _first(input) + _second(input);
 }
private:
  first_block _first;
  second_block _second;
};

template<class first_block, class second_block>
class sequence_pair
{
public:
 output_t operator()(input_t input)
 {
    return _second(_first(input));
  }
private:
  first_block _first;
  second_block _second;
};
```

Listing 5. Parallel and sequential block connectors

Larger structures could become unreadable when using nested pair connectors. Solution based on statically defined, heterogeneous container is more readable and easier to use alternative.

In the case of fixed number of blocks, instead of nested pair connectors, it is simpler to use variadic templates provided by modern revision of C++ or D language standard. *Typelists* [3] can be used as an alternative for compilers that lack support for this language feature.

If there is a need to scale number of connected elements, or the type of each block needs to be chosen algorithmically, type policies are a suitable solution. Type of each block may be determined by an index in the sequence. Such template class, passed as a template template parameter to heterogeneous container, guide compiler in memory offsetting and choosing methods that need to be applied. Logic, that determines the type for the given index needs to be defined in meta-code and evaluated during compile time. Here, template meta-programming acts as a solution. If supported, C++11 *constexpr* language feature may be used to simplify syntax. An example of type policy is provided in code listing 6.

```
template<size_t index>
class dsp_block_policy
{
public:
  typedef sequential_pair<
```

```
    blocks::delay_line<100 * index /*delay in samples*/>,
    blocks::iir_filter<2 /*poles*/, 1/*zeros*/>> type
}
```

Listing 6. Example of type policy. With use of static conditional statements and meta-programming, highly complex type choosing logic can be implemented.

As usage of such policy may not be clear, we propose a generic (not limited to DSP applications) implementation of heterogeneous container (code listing 7) that supports algorithmically defined block types. Container based on variadic template may be implemented in similar way. Having implementation of such container on hand, any type of connector can be created in a straightforward manner.

```
template<
  size_t size,
  template <size_t> class types_policy
> class meta_container
{
  // make root element accessors a friends
  template<size_t index> friend class get;
  template<
    template <size_t> class types_policy,
    size_t size,
    typename functor
  > friend void for_each(meta_container<size,
        types_policy>& container, functor& functor);

public:
  MetaContainer()
  {
    static_assert(
        size != 0,
        "meta_container of size 0 is not allowed."
    );
  }

private:
  meta_node<types_policy, 0, size> _root;
};

template<
  template <size_t> class types_policy,
  size_t idx,
  size_t size
> class meta_node
{
```

```
public:
  typedef std::integral_constant<size_t, idx> index;

  // type of element of this node
  typedef typename types_policy<idx>::type head;
  // tail node type
  typedef typename std::conditional<
    (size - 1 > idx),
    meta_node<types_policy, idx + 1, size>,
    null_type
  >::type tail;

public:
  head value;
  tail next;
};
```

Listing 7. Heterogenous container implementation, with type definitions provided by type policy. Instance of such container is contiguous in memory, despite linked-list like implementation, as iteration over elements takes place during compile time.

```
  template<size_t index>
  class get_node
  {
  public:
    template<
      template <size_t> class types_policy,
      size_t index,
      size_t size
    > static inline
    typename types_policy<index + typeIndex>::type&
    from(meta_node<types_policy, typeIndex, size>& container)
    {
      return get_node<index - 1>::from(container.next);
    }
  };

  // terminates template recursion
  template<>
  class get_node<0>
  {
  public:
    template<
      template <size_t> class types_policy,
      size_t index,
      size_t size
    > static inline
    typename types_policy<index >::type&
    from(meta_node<types_policy , index , size>& container)
    {
      return container.value;
    }
  };
```

Listing 8. Accessor method of meta-programming based container. Friend *get* class of meta_container delegates iteration to *get_node* classes shown in this listing, starting at the root element. Iteration is analogous to iterating over linked-lists, yet is performed during compile time. Foreach iteration has been implemented similarly to loop-unrolling code presented in listing 2 in Section 2, therefore we omit its implementation.

In the provided implementation, we can see that heterogeneous containers, such as tuples, are metaprogramming equivalent of linked-lists. Because iterating over elements is performed during compile time, there is no overhead of pointer dereferencing and cache issues typical to regular linked-list implementation. Also, type of each element is inferred by accessor methods, and many errors that would normally raise an exception in object oriented implementations occur as compilation errors, not runtime errors.

3.2.2 Composing Signal Flow via Template-Based Variant of Decorator Pattern

Template metaprogramming is enabled by *duck-typing* (naming based) paradigm enabled by template system [29], which still keeps the benefit of a fully type-safe language (types are still validated during compile time, in contrast to typical duck-typing languages like Python and Ruby). This behaviour is considered to be a case of static polymorphism [4]. Therefore, many design patterns can be mapped directly into their static equivalents [3].

In the object oriented approach, the usage of decorator pattern is considered to be an out of the box solution for pipelining computations. Templates allow to achieve the same effect without the need of using dynamic polymorphism. Therefore, in cases of sequential signal flow, simpler solution, that is based on metaprogramming implementation of decorator pattern, may be applied.

Single processing block may be defined as shown in listing 9. Such blocks compositions mimic the semantics of decorator pattern and allow to build processing pipelines.

```
template<typename prev_block>
class seq_dsp_block
{
public:
  signal_type operator()(signal_type input)
  {
    signal_type out = _prev(input);
    // signal processing code
    return out;
  }
private:
  prev_block _prev;
};
```

Listing 9. Static variant of decorator pattern applied in DSP processing block

3.3 Example of Schroeder Reverberator Implementation

As an example, we show how proposed techniques are useful in implementation of one of the first and simplest artificial reverberation algorithms, the Schroeder reverberator. Its structure is shown in Figure 2.

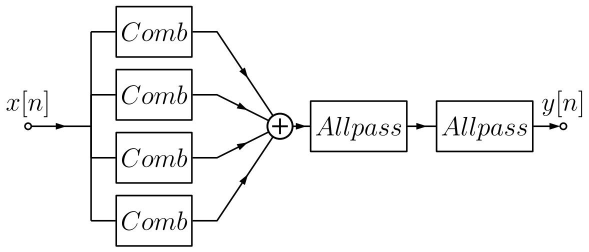


Figure 2. Schroeder reverberator block diagram

Having implementations of comb and allpass filters provided, Schroeder reverberator may be expressed as appears in code listing 10. In the provided example, blocks act as signal processing policies, defining specific behavior of generic structure (in this case, parallel and sequential groups). Since unified interface is defined as input/output type pairs, instantiated structure may act as building block for further composition, providing high level of modularity.

```
  typedef sequence<
      parallel_sum<
        comb<delay<unsigned int>>,
        comb<delay<unsigned int>>,
        comb<delay<unsigned int>>,
        comb<delay<unsigned int>>
      >,
      all_pass<delay<unsigned int>>,
      all_pass<delay<unsigned int>>
```

```
> schroeder_rev;
```

Listing 10. Schroeder reverberator implementation

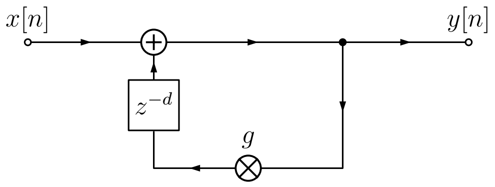


Figure 3. Comb filter block diagram

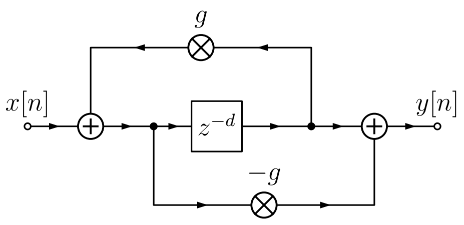


Figure 4. All-pass filter block diagram

Building blocks of Schroeder reverberator, namely comb (Figure 3) and all-pass (Figure 4) filters are simple forms of recursive structures, that due to their specifics (substantial delay of few thousand samples in typical applications) could benefit from custom implementation. Idea of block diagram decomposition may be developed further - as an example we present code listing 11, showing comb filter implementation.

```
typedef recursive<
    transparent, // forward
    sequence<scale, delay<unsigned int>> // recursive
  > comb;
```

Listing 11. Comb filter implementation

Even though such level of decomposition is possible with no impact of performance, it is generally better to provide predefined block implementations at this level of granularity.

3.4 Implementing Singe Instruction, Multiple Data (SIMD) Optimised DSP Components

SIMD instruction set usage is necessary to maximise performance of all current CPU architectures. Code vectorisation is not always a trivial task, mostly because of data dependencies and the requirement of properly aligned memory. Even though all current compilers offer auto-vectorisation feature, non-trivial cases must still be vectorised by a skilled programmer. Fortunately, as mentioned in the beginning of this section, approach described in this paper allows for heterogenous structure composition. The same principle, that allows for mixing floating/fixed point arithmetic, enables consistent SIMD processing.

Implementing DSP blocks with use of Streaming SIMD Extensions (SSE), Advanced Vector Extensions (AVX) or Advanced SIMD (NEON) instruction sets have been a subject of many studies [17, 14, 18, 23, 25]. Even vectorisation of, recursive in nature, infinite impulse response (IIR) filtration has been tacked by some [23]. For example, ITU G729C codec post processing formula transforms IIR difference equation for the samples vector (consisting of 4 or 8 consecutive samples) into the form of matrix multiplication. Techniques of DSP code vectorisation are not in scope of this paper. We focus mainly on SIMD vectorisation in context of proposed metaprogramming based implementations.

In order to create vectorised structures, each DSP block needs to operate on SIMD vector types, such as __m128 or __m256 keeping concise and elegant interface via propagation of the given type throughout series of connected blocks. The only limitation that constrains usage of SIMD instructions, results from recursive connections with delay lines shorter than number of samples processed on operational type used (e.g. 3 samples of delay when using SSE with single precision arithmetic). To avoid runtime errors, delay introduced by delay line should be fixed with statically defined value (via template parameter). Moreover, static_assert keyword should be used in order to detect errors during compilation (e.g. listing 12).

```
template <size_t delay>
class delay_line_SSE
{
  delay_line_SSE()
  {
    static_assert(
        delay >= 4,
        "delay_line_SSE: delay parameter needs to be >=4."
    );
  }

  inline __m128 operator()(__m128)
  {
    // delay line logic
  }
};
```

Listing 12. Using static asserts to detect errors during compilation

4 COMPLEX USE CASE: FEEDBACK DELAY NETWORK IMPLEMENTATION

Feedback Delay Network (FDN), being one of the most naturally sounding artificial reverberators [13, 24], became widely implemented in many sound processing software products. Although FDN is a very potent tool, achieving proper perceptual quality of acoustic simulation demands additional modifications to its structure. This applies to each of processing lines, as well as to the input/output filters. FDN needs to be fine-tuned to achieve proper acoustic properties, but having quite complex structure it demands proper level of implementation configurability. Implementation described in this section presents benefits of proposed technique, providing high level of code composability while retaining high efficiency via elimination of unnecessary polymorphic calls, heap memory allocations, SIMD vectorisation and other low level optimisations.

4.1 Brief Description FDN Reverberator Structure

The structure of FDN consists of N delay lines $DL_m = z^{(-m)}$, each resulting in a signal being delayed by $t_i = \frac{m_i}{f_s}$ seconds, where f_s is the sampling frequency. Output of these lines acts as an input to the orthogonal diffusion matrix producing output signals as well as the feedback mixed into input signal.

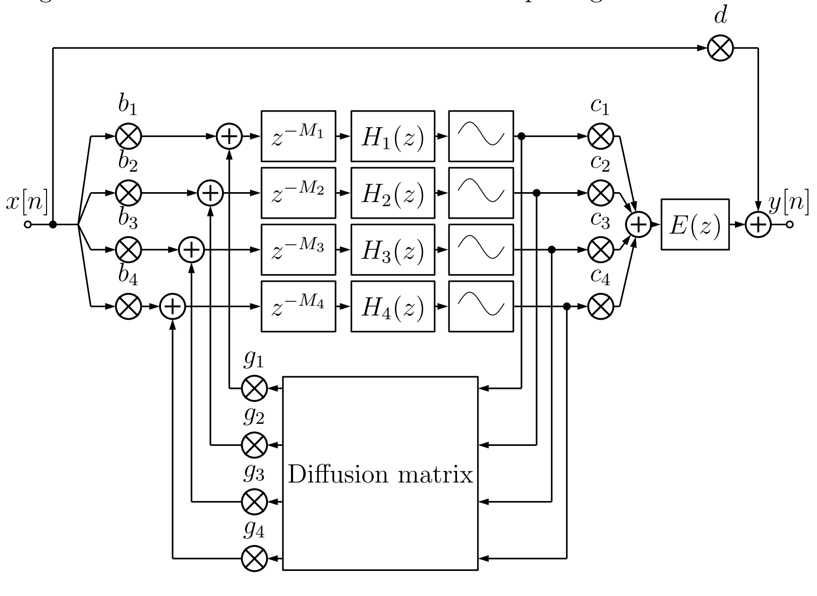


Figure 5. Feedback delay network block diagram

In the FDN block diagram (pictured in Figure 5), we can see various blocks that shall be abstracted, namely blocks within processing lines and diffusion matrix.

More advanced implementations introduce elements such as IIR filters and tone correction filters in addition to delay lines.

FDN aims to approximate acoustical environments, which may be modelled as a complex audio system with thousands of poles and zeros. Real time simulation of such system would be unacceptable in terms of computational complexity. Even though FDN is less computationally expensive, higher order networks still consume considerable amount of CPU time [13].

4.2 Achieving Modularity of FDN Implementation via Metaprogramming and Policy-Based Design

Primary motivation to use policy-based design in FDN implementation was the need to efficiently experiment with different configurations of DSP blocks within each of the processing lines, while being still able to match, and therefore assess, performance of highly optimised system. Application of object oriented design would have severe runtime impact, mostly due to virtual method calls. All the primitives described in previous section allow us to encode DSP diagram in form of nested templates. Proposed implementation is described in code listing 13.

```
template <size_t index>
class line_policy
{
public:
  typedef sequential<
    nth_prime<unsigned int /* start after */,
      index * 10 /* take each prime in strid of 10 */>::value,
    iir_filter<3 /*poles*/, 2/*zeros*/>,
    modulator<unsigned int>> type;
}

typedef parallel_sum<
  scale, // d scaling factor
  sequence<
    sum<
      parallel_vector<
        scale_vector, // b scaling factors
        recursive_vector<
          vector_of<
            line_policy,
            4
          >, // forward
          sequence_vector<
            fwht<4>, // diffusion matrix
            scale_vector, // g scaling factors
          > // backward
```

```
            >,
            scale_vector , // c scaling factors
         >,
         4
      >,
      iir_filter <3/*poles*/,2/*zeros*/>
   >> fdn_rev;
```

Listing 13. Encoded FDN diagram, according to previously discussed techniques. Notice vector processing blocks, that allow for recursive mixing of signals by the diffusion matrix (here implemented in form of fast Walsh-Hadamard transform).

5 RELATED WORK AND FURTHER DEVELOPMENT

Common DSP optimisations have been described by authors of widely used libraries. Frigo et al. [8] provided optimisation guidelines for FFT, also applicable to other *divide and conquer* derived methods. Authors focused on widely available CPU features, including OOO, super-scalar capabilities and SIMD instruction sets. SPIRAL software, described by Puschel at al. [22] addresses transforms code generation, including FFT, DWT and other.

Our approach, being more generic, is applicable to broader range of DSP algorithms. Although some authors consider the idea of DSP diagrams based automatic code generation [21, 34], we have not managed to find any recent publications that stay up to date with advancements in processor architectures.

Proposed technique efficiently solves the problem of providing fast, yet configurable DSP algorithm implementations. When used with simplified, easy to understand API, which hides the complexities of template metaprogramming, proposed techniques create a powerful framework for signal processing applications. Regardless of an exact use case, programmers can easily take an advantage of all the optimisations provided by current, highly advanced compilers.

The proposed solution may be applied in:

- Automatic translation of DSP diagrams into C++ code that is ready to compile.
- Functional paradigm domain specific languages, that are translated directly to C++.
- Visual tools, that allow for easy implementation and validation of DSP designs, allowing for measuring performance that is close to the upper limit available on the given hardware.

6 CONCLUSIONS

The solutions proposed in this paper, in spite of its unconventional characteristics, were proven to be practical in software that needs to deal with fine grained, computationally intense problems. High performance, code readability and configurability were achieved with statically defined modular architecture. Disassembly of generated software confirms that usage of metaprogramming methods leads to highly optimised machine code, that indeed looks like it has been hand tweaked by experienced assembly programmer. Moreover, being able to easily switch between SIMD implementations makes it easy to test performance on different architectures, keeping code organised and readable.

Acknowledgments

This work was supported by MNiSW resources for science as statutory activity.

REFERENCES

[1] Abrahams, D.—Gurtovoy, A.: C++ Template Metaprogramming: Concepts, Tools, and Techniques from Boost and Beyond. Pearson Education, 2004.

[2] Ahmed, N.—Rao, K. R.: Walsh-Hadamard Transform. Orthogonal Transforms for Digital Signal Processing. Chapter 6. Springer, 1975, pp. 99–152, doi: 10.1007/978-3-642-45450-9_6.

[3] Alexandrescu, A.: Modern C++ Design: Generic Programming and Design Patterns Applied. Addison-Wesley, 2001.

[4] Burrus, N.—Duret-Lutz, A.—Géraud, T.—Lesage, D.—Poss, R.: A Static C++ Object-Oriented Programming (SCOOP) Paradigm Mixing Benefits of Traditional OOP and Generic Programming. Proceedings of the Workshop on Multiple Paradigm with OO Languages (MPOOL '03), Anaheim, CA, USA, 2003.

[5] Chen, J.—Joo, B.—Watson, W.—Edwards, R.: Automatic Offloading C++ Expression Templates to CUDA Enabled GPUs. 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops and Ph.D. Forum (IPDPSW), 2012, pp. 2359–2368, doi: 10.1109/IPDPSW.2012.293.

[6] Driesen, K.—Hölzle, U.: The Direct Cost of Virtual Function Calls in C++. ACM Sigplan Notices, Vol. 31, 1996, No. 10, pp. 306–323, doi: 10.1145/236337.236369.

[7] Fog, A.: Optimizing Software in C++: An Optimization Guide for Windows, Linux and Mac Platforms, 2004.

[8] Frigo, M.—Johnson, S. G.: FFTW: An Adaptive Software Architecture for the FFT. Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, 1998, Vol. 3, pp. 1381–1384, doi: 10.1109/ICASSP.1998.681704.

[9] Frigo, M.—Johnson, S. G.: The Design and Implementation of FFTW3. Proceedings of the IEEE, Vol. 93, 2005, No. 2, pp. 216–231, doi: 10.1109/JPROC.2004.840301.

[10] Gamma, E.—Helm, R.—Johnson, R.—Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Pearson Education, 1994.

[11] Gawlik, I.—Pędzimąż, T.—Pałka, S.—Ziółko, B.: Efficient Vectorized Architecture for Feedback Delay Network Reverberator with Policy Based Design. Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), Poznan, 2015, pp. 124–127.

[12] Hammarlund, P.—Martinez, A. J.—Bajwa, A. A.—Hill, D. L.—Hallnor, E. et al.: Haswell: The Fourth-Generation Intel Core Processor. IEEE Micro, Vol. 34, 2014, No. 2, pp. 6–20, doi: 10.1109/MM.2014.10.

[13] Jot, J.-M.—Chaigne, A.: Digital Delay Networks for Designing Artificial Reverberators. Audio Engineering Society Convention, AES, Vol. 90, 1991, Art. No. 3030.

[14] Lee, J.—Moon, S.—Sung, W.: H.264 Decoder Optimization Exploiting SIMD Instructions. Proceedings of the 2004 IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS), 2004, Vol. 2, pp. 1149–1152.

[15] Mahapatra, N. R.—Venkatrao, B.: The Processor-Memory Bottleneck: Problems and Solutions. Crossroads – Computer Architecture, Vol. 5, 1999, No. 3es, Art. No. 2.

[16] McCartney, J.: Rethinking the Computer Music Language: SuperCollider. Computer Music Journal, Vol. 26, 2002, No. 4, pp. 61–68, doi: 10.1162/014892602320991383.

[17] Nguyen, H.—John, L. K.: Exploiting SIMD Parallelism in DSP and Multimedia Algorithms Using the AltiVec Technology. Proceedings of the 13th International Conference on Supercomputing (ICS '99), ACM, 1999, pp. 11–20, doi: 10.1145/305138.305150.

[18] Norilo, V.—Laurson, M.: Kronos – A Vectorizing Compiler for Music DSP. Proceedings of the 12th International Conference on Digital Audio Effects (DAFx-09), 2009.

[19] Orlarey, Y.—Fober, D.—Letz, S.: FAUST: An Efficient Functional Approach to DSP Programming. In: Assayag, G., Gerzso, A. (Eds.): New Computational Paradigms for Computer Music. IRCAM, 2009.

[20] Plauger, P. J.—Stepanov, A. A.—Lee, M.—Musser, D.: C++ Standard Template Library. Prentice Hall PTR, 2000.

[21] Powell, D. B.—Lee, E. A.—Newman, W. C.: Direct Synthesis of Optimized DSP Assembly Code from Signal Flow Block Diagrams. 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP-92), Vol. 5, 1992, pp. 553–556, doi: 10.1109/ICASSP.1992.226560.

[22] Puschel, M.—Moura, J. M.—Johnson, J. R.—Padua, D.—Veloso, M. M.—Singer, B. W.—Xiong, J.—Franchetti, F.—Gacic, A.—Voronenko, Y.—Chen, K.—Johnson, R. W.—Rizzolo, N.: SPIRAL: Code Generation for DSP Transforms. Proceedings of the IEEE, Vol. 93, 2005, No. 2, pp. 232–275, doi: 10.1109/JPROC.2004.840306.

[23] Robelly, J. P.—Cichon, G.—Seidel, H.—Fettweis, G.: Implementation of Recursive Digital Filters into Vector SIMD DSP Architectures. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '04), 2004, Vol. 5, pp. 165–168, doi: 10.1109/ICASSP.2004.1327073.

[24] Rocchesso, D.—Smith, J. O.: Circulant and Elliptic Feedback Delay Networks for Artificial Reverberation. IEEE Transactions on Speech and Audio Processing, Vol. 5, 1997, No. 1, pp. 51–63, doi: 10.1109/89.554269.

[25] Shahbahrami, A.—Juurlink, B.—Vassiliadis, S.: Efficient Vectorization of the FIR Filter. Proceedings of the 16th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC), 2005, pp. 432–437.

[26] Sipos, Á.—Porkoláb, Z.—Pataki, N.—Zsók, V.: Meta⟨Fun⟩ – Towards a Functional-Style Interface for C++ Template Metaprograms. Proceedings of 19th International Symposium of Implementation and Application of Functional Languages (IFL 2007), 2007, pp. 489–502.

[27] Stewart, T.: Intel Integrated Performance Primitives: How to Optimize Software Applications Using Intel IPP. Intel Press, 2004.

[28] Stroustrup, B.: The Design and Evolution of C++. Pearson Education India, 1994.

[29] Stroustrup, B.: Foundations of C++. In: Seidl, H. (Ed.): Programming Languages and Systems (ESOP 2012). Springer, Lecture Notes in Computer Science, Vol. 7211, 2012, pp. 1–25, doi: 10.1007/978-3-642-28869-2_1.

[30] Veldhuizen, T.: Expression Templates. C++ Report, Vol. 7, 1995, No. 5, pp. 26–31.

[31] Veldhuizen, T. L.: C++ Templates Are Turing Complete. Available at `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.14.3670&rep=rep1&type=pdf`, 2003.

[32] Wang, G.: The Chuck Audio Programming Language. A Strongly-Timed and On-the-Fly Environ/Mentality. Ph.D. dissertation, Princeton University, 2008.

[33] Wang, G.—Cook, P. R.: ChucK: A Concurrent, On-the-Fly, Audio Programming Language. Proceedings of the 2003 International Computer Music Conference, 2003, pp. 219–226.

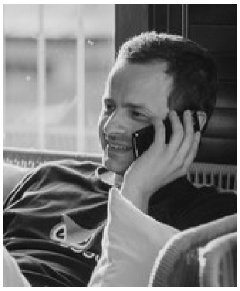
[34] Wess, B.—Kreuzer, W.: Optimized DSP Assembly Code Generation Starting from Homogeneous Atomic Data Flow Graphs. Proceedings of the 38th Midwest Symposium on Circuits and Systems, 1995, Vol. 2, IEEE, pp. 1268–1271, doi: 10.1109/MWSCAS.1995.510327.

[35] Ziółko, B.—Pędzimąż, T.—Pałka, S.—Gawlik, I.—Miga, B.—Bugiel, P.: Real-Time 3D Audio Simulation in Video Games with RAYAV. Making Games, Vol. 1, 2015.

Ireneusz Gawlik received his M.Sc. degree in acoustical engineering and B.Sc. in computer science from the AGH University of Science and Technology in Kraków, Poland. Currently he is working toward his Ph.D. degree in computer science at the AGH UST. His research is focused on creating models of acoustic phenomena and development of audio processing algorithms. He is also working on development of machine learning algorithms, machine learning in BigData setups, recommender systems and learning to rank.

Szymon Pałka received his M.Sc. degree in computer science from the AGH University of Science and Technology in Kraków, Poland. Currently he is working toward his Ph.D. degree in computer science at the AGH UST. His research is focused on optimization of geometric techniques for 3D scene analysis for spatial audio processing. He is a co-author of a patent application and scientific papers regarding simulation of sound propagation and speech processing. He teaches "Computer Graphics" class at the AGH University.

Tomasz Pędzimąż received his M.Sc. degree in computer science from the AGH University of Science and Technology in Kraków, Poland. Currently he is working toward his Ph.D. degree in computer science at the AGH UST. He is an author or co-author of 10 scientific papers and 2 patent applications, with one patent granted. He has participated in several national and European research projects. His research is focused on natural language processing regarding speech recognitions language models. He teaches "Computer Graphics" class at the AGH University.



Bartosz Ziółko studied electronics and telecommunications at the AGH University of Science and Technology in Krakow. Next he did his Ph.D. in computer science at the University of York. He is an author or co-author of over 100 scientific papers and of 3 patent applications, with two patents granted. He is the main author of book "Przetwarzanie mowy" (Eng. Speech Processing). His research interests include automatic speech recognition, natural language modeling, speaker recognition and soundtracing. He is CEO and a co-founder of Techmo – an AGH spin-off company and Assistant Professor at AGH University of Science and Technology. He has participated in several national and European research projects. He was also engaged as external consultant in speech technologies for companies. His R & D activity resulted in a few products licensed to companies, universities and Court. He is also governmental technology expert. He was trained in research commercialization by Stanford and in Science Infrastructure Management by IBM and Fraunhofer Institute.

Computing and Informatics, Vol. 37, 2018, 291–310, doi: 10.4149/cai_2018_2_291

AHP AIDED DECISION-MAKING IN VIRTUAL MACHINE MIGRATION FOR GREEN CLOUD

Liumei Zhang

School of Computer Science and Technology, Xidian University
2 South Taibai Road, Xi'an, Shaanxi, P.R. China 710071
&
School of Computer Science, Xi'an Shiyou University
Xi'an, Shaanxi, P.R. China 710065
e-mail: `rikrun@gmail.com`

Jianfeng Ma

School of Computer Science and Technology, Xidian University
2 South Taibai Road, Xi'an, Shaanxi, P.R. China 710071
e-mail: `jfma@mail.xidian.edu.cn`

Tianshi Liu

School of Computer Science, Xi'an Shiyou University
Xi'an, Shaanxi, P.R. China 710065
e-mail: `liutianshi@xsyu.edu.cn`

Yichuan Wang, Di Lu

School of Computer Science and Technology, Xidian University
2 South Taibai Road, Xi'an, Shaanxi, P.R. China 710071
e-mail: {`ctechsky, nijino2002`}`@gmail.com`

Abstract. In this study, an analytical hierarchy process based model is proposed to perform the decision-making for virtual machine migration towards green cloud computing. The virtual machine migration evaluation index system is established based on the process of constructing hierarchies for evaluation of virtual machine migration, and selection of task usage. A comparative judgment of two hierarchies has been conducted. In the experimental study, five-point rating scale has been adopted to map the raw data to the scaled rating score; this rating method is used to analyze the performance of each virtual machine and its task usage data. The results show a significant improvement in the decision-making process for the virtual machine migration. The deduced results are useful for the system administrators to migrate the exact virtual machine, and then switch on the power of physical machine that the migrated virtual machine resides on. Thus the proposed method contributes to the green cloud computing environment.

Keywords: Green cloud computing, VM migration, AHP, decision support

Mathematics Subject Classification 2010: 68U35, 68Q87, 68Q99

1 INTRODUCTION

The ever developing cloud computing infrastructures always pose problems to environment safety and protection. The data center hosting cloud applications consume large amounts of energy, that results in high operating costs and carbon is released into the atmosphere [1]. The usage of clouds gives rise to the questions whether cloud computing is a cloud of pollution. The term green cloud computing has been defined to depict the study area focused on the alleviation of negative effects that cause the environmental burden. A number of researchers have presented a wide range of methods to support the green cloud.

In green cloud computing a number of factors have been considered such as power consumption, space occupancy and heat dissipation to achieve energy saving and environmental protection. Among all the factors, power consumption and heat dissipation are of the highest importance because these two factors have direct impact on the environment. The green cloud computing has gained appreciative popularity among academic researchers worldwide. The energy saving strategies for cloud computing platform were proposed in [2]. A framework is proposed to automatically manage resources of cloud infrastructures in order to reduce the amount of energy consumption to a minimum level [3]. Energy efficient multithreading local search algorithm is proposed for solving the multiobjective scheduling problem in heterogeneous computing systems [4]. Naturally, the best way to reduce the energy consumption and heat emission is to limit the usage of such infrastructure. More precisely, it is a good idea to switch off the cloud server when it is not in use. In the previous work [5], authors have proposed an attribute clustering based collaborative filtering method to generate the migration recommendation of the virtual machine (VM) for administrator. The presented method can calculate the similarities between defined target VM to migration and other VMs. However, it requires the administrator to manually perform the migration of VM. Therefore, it is unable to help the administrators to identify the exact VM to migrate according to their criteria. The identification of the servers in a cluster to switch off is a challenging task. The cloud servers today are often the host of a bunch of virtual machines (VM); therefore, before shutting it down, a cloud server needs to be analyzed to secure the active running VMs. The active VMs can be migrated onto other running cloud servers before shutdown. Another challenging task is to determine whether a VM is active. Fortunately, the behavior of usage of a VM can be used to determine whether a VM is active. The cloud computing can enable more energy-efficient use of computing power, especially when the computing tasks are of low intensity or infrequent [6]. It is feasible to measure the VMs with task usage information; however, the decision of the VMs to migrate is still an open research problem. It is usually referred to as a decision-making problem.

The Analytical Hierarchy Process (AHP) [7] is a decision-making approach designed to aid the solution of complex multiple criteria problems in a number of application domains. This method has been found to be an effective and practical approach that can consider complex and unstructured decisions [8]. AHP has been widely used as a decision-making technique over various application domains [9, 10, 11, 12, 13, 14, 15]. In the cloud computing environment, AHP has also been applied to task scheduling and resource allocation [16], strategies of green energy saving [2], services ranking [17] and evaluation [18], quality of service [19], etc. In this study, the AHP is applied to aid the decision making in VM migration. The factors that affect the VM migration decisions are elicited to establish hierarchy for evaluation of VM resource usage. The administrators can judge the importance of each criterion in pair-wise comparisons; then, prioritized ranking or weighting of each VM migration alternative can be obtained. The task usage factor is dependent on various data types; however, only the CPU usage factor is considered in this study. Based on such factors, this study has focused on formulating an AHP-based model to select a VM which is suitable for migration onto another stable host. Hence, the cloud server with the rest of VMs can be powered off for energy saving.

The rest of this paper is organized as follows. Section 2 introduces the essential network topology of typical cloud computing. In Section 3 index system establishment and its analysis are presented and the comparative judgement analysis of the index system is provided. Section 4 describes the algorithm. The simulation and the experimental results are discussed in Section 5. Section 6 concludes the paper.

2 NETWORK TOPOLOGY

The virtual machines cannot operate as an independent system. They reside on the virtual machine manager (VMM) of the physical machine which is further connected by core switches in a cloud server cluster. Furthermore, the core switches are connected with migration manager (MM) for decision-making of VM migration strategy. In other words, the MM is responsible for the determination when and which the VM to be migrated. A server cluster often employed with firewalls serves the Internet via routers. This paper presents a strategy to resolve the configuration problem within MM. For energy saving purposes, when the cluster server has light load, some of the VMs are migrated onto other physical running server and the idle physical machines are powered off. Figure 1 illustrates the typical network topology of a cloud server cluster; where the Physical Machine (PS), Migration Manager (MM), Virtual Machine (VM), Switcher (SW), Firewall (FW), and Router (R) are clearly presented.

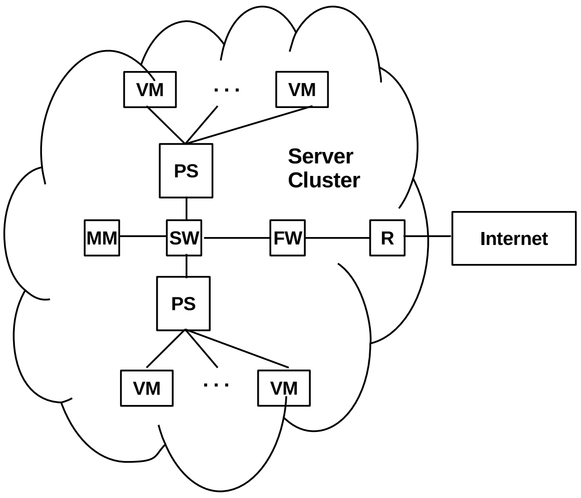


Figure 1. Network topology

The migration process of a VM falls into two categories: the "cold" and "hot" migration techniques. The "cold" migration indicates the VM needs to be powered off before migration. It will be rebooted on the destination physical server after the migration process. On the other hand, "hot" migration indicates the VM needs to be halted before the migration. It will be activated on the destination physical server after the migration process. The "hot" migration VM has very short service interruption time, often counted less than a hundred ms. Therefore, "hot" migration techniques are more common than the "cold" migration techniques. For clarity, the "hot" migration steps are given as follows.

1. MM decides the migration, and the destination physical machine for a VM,
2. halts the VM to migrate,
3. copies the entire memory mapping and CPU register status onto physical server,
4. registers the migration VM on the VMM of the destination physical server,
5. activates the migration VM on the VMM of the destination physical server,
6. switches off the original VM on the source physical server.

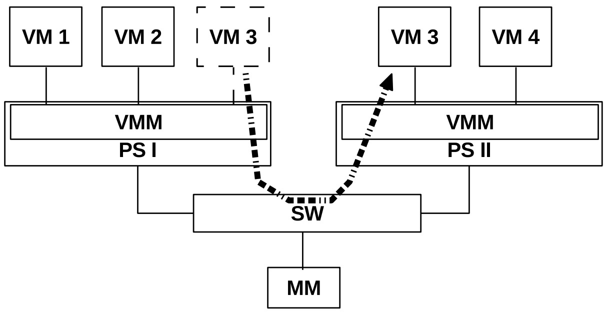


Figure 2. Migration of VM

3 VM MIGRATION EVALUATION MODEL

As the cloud service subscribers are increasing and the costs of the rent of such services are decreasing, VMM may contain the VMs that are infrequently used or idle long-term. Keeping such VMs together with frequently used VMs running on the same VMM is worthless. Particularly, when the number of idle VMs is greater than the number of frequent VMs, much power is wasted to keep the idle VMs alive. In order to alleviate the power consumption of cloud cluster, the PS can be powered off once the frequently used VMs is defined and migrated. On the other hand, the resource usage of receiving PS can also be maximized to improve the power performance of the system. All the VMs shall be evaluated according to certain criterion that reflects the utilization rate before deciding which of the VMs can be migrated. Thus, the frequently used VMs migration is considered as a decision-making problem in this study. The applications of AHP to complex decision situations have numbered in thousands [20], and it has produced extensive results in problems involving planning, resource allocation, priority setting, and selection among alternatives. Other areas include forecasting, total quality management, business process re-engineering, quality function deployment, and the balanced scorecard [21]. Therefore, in this paper, the AHP is adopted for the evaluation of VMs to find out the suitable migration VM according to the specified criteria of cluster administrator. Basically, there are three steps needed to apply the AHP to decision-making problems: constructing hierarchies; comparative judgment; and synthesis of priorities [22].

3.1 Establishment of VM Evaluation Index System

The construction of hierarchies requires eliciting the indicators for building an index system. Such system can be a comprehensive measurement for the scale sets towards the object. It is also a system analysis method that has been well accepted among many application domains such as social, economic and management science. The evaluation system often has a hierarchical structure, which contains two levels to form a multi-level evaluation system: the objectives and the principles.

For constructing hierarchies, the indicators that reflect properties which are helpful for VM migration need to be elicited. The goal of such model is the evaluation of VM migration. The CPU usage and cyclicity are considered as two performance criteria. The CPU usage consists of two sub-criteria including the average CPU rate and maximum CPU rate. The cyclicity consists of two sub-criteria including cycles per instruction and task duration. Figure 3 shows the hierarchy with VMs as the alternatives.

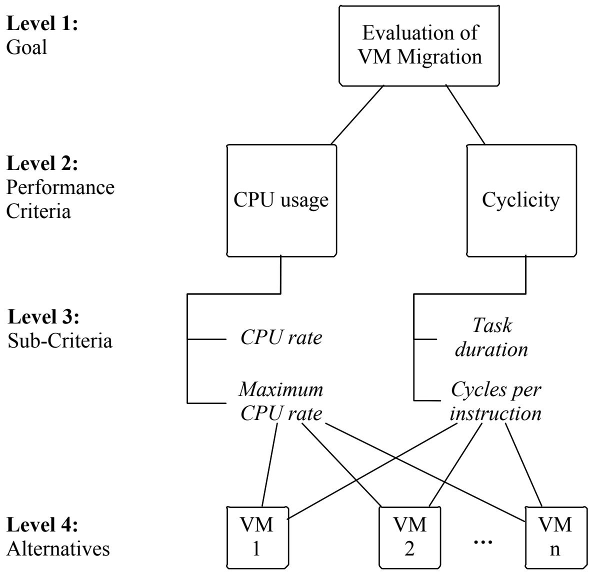


Figure 3. VM migration evaluation hierarchy

From the hierarchy shown in Figure 3 the VMs which are feasible to migrate can be obtained by the ranking result produced in AHP calculation. However, the VMs execute tasks from time to time. Therefore, a single task usage information cannot represent the exact behavior of a VM. A single task usage of VM consists of arbitrary task vectors. For this reason, a second hierarchy is proposed to select the most representative task usage. The configuration of the hierarchy is presented in Figure 4.

It can be seen that the Figure 4 is partially similar to Figure 3, and the goal of such evaluation is to select most representative task usage information. This information is obtained through the performance criteria. Similarly, CPU usage and the cyclicity are main indicators for task measurement. In addition, a new factor called task validity has been introduced to depict the time interval of a task from start to present. Task validity is used to describe that the completion time of a task execution to the time of AHP calculation. That is to say, for a same

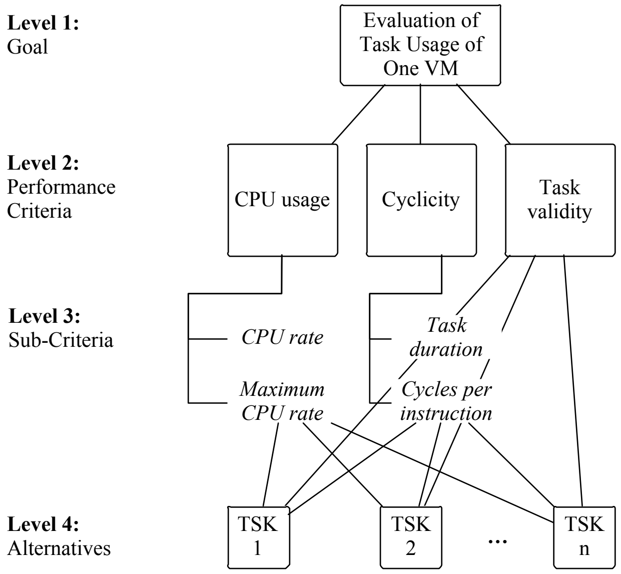


Figure 4. Task selection evaluation hierarchy

VM, if the completion time of a task execution is aged, then this task would be less representative than the task which executes recently.

3.2 Comparative Judgment

Once the hierarchy has been established, the comparative judgment needs to be implemented towards the referenced indicators. A pair-wise comparison is needed to formalize the weight of all indicators in this step.

Before such comparison, all the indicators should be analyzed and their influence upon VM migration must be established.

Definition 1 (Suitable/unsuitable for migration)**.** A virtual machine (VM) is defined as suitable for migration where the VM consumes low CPU resources of a physical machine and the task duration is infrequent. A virtual machine (VM) is defined as unsuitable for migration where the VM consumes high CPU resources of a physical machine and the task duration is frequent.

When the average CPU rate of a virtual machine is low in a cloud cluster, the running cloud service consumes low CPU resources, or perhaps it is idle. This indicates that the migrated virtual machine will not take up much of the CPU resource of the receiving physical machine; such migration is helpful in saving power. Similarly, the virtual machine that has a low value of maximum CPU rate would behave the same way. On the other hand, a short CPU task duration indicates the frequent CPU scheduling tasks on virtual machine. Therefore, if an infrequent is found, it should be migrated. The cycles per instruction describe the instruction type that CPU has executed. When the cycles per instruction are short, it may indicate the instruction is conventional, that includes the operation instruction, a short instruction and a short data operation instruction. Typically, those instructions are only used for calling register. In this situation, the virtual machine can be migrated for a long-term operation. A long cycle per instruction indicates that the co-processor is required for large data operation or an abnormal control transfer is executed. More precisely, the virtual machine will be terminated if the operation duration is not long. Therefore, it is unnecessary to migrate.

The criteria and sub-criteria identified as being important in the VM migration decisions can be summarized from Figures 3 and 4; it is provided in Table 1.

It can be concluded from Table 1 that the goals of proposed model are the evaluation of VM migration and task usage of the VM. For VM evaluation (G_1) task, the CPU usability (C_1) and cyclicity (C_2) are two criteria which further consist of CPU rate (C_{11}), maximum CPU rate (C_{12}), task duration (C_{21}) and cycles per instruction (C_{22}). For task usage evaluation (G_2) task, the CPU usability (B_1), cyclicity (B_2) and task validity (B_3) are the key criteria where CPU usability (B_1) and cyclicity (B_2) consist of CPU rate (B_{11}), maximum CPU rate (B_{12}), task duration (B_{21}) and cycles per instruction (B_{22}).

Goal	Criteria	Sub-Criteria
VM evaluation (G_1)	CPU usability (C_1)	CPU usage (C_{11})
		Maximum CPU usage (C_{12})
	Cyclicity (C_2)	CPU task duration (C_{21})
		Cycles per instruction (C_{22})
Task usage evaluation (G_2)	CPU usability (B_1)	CPU usage (B_{11})
		Maximum CPU usage (B_{12})
	Cyclicity (B_2)	CPU task duration (B_{21})
		Cycles per instruction (B_{22})
	Task validity (B_3)	

Table 1. Criteria and sub-criteria of VM migration

When the cloud cluster administrator determines the migration of VMs for power saving reason, he may refer to the logs of VMs that resides in such VMM. The VMs can be pairwise compared using the comparison matrix technique. A comparison matrix can be built based on the Saaty Rating Scale [23], as shown in Table 2, which is used to determine the relative importance of each VM in terms of each criterion. Furthermore, the weights of all VMs can be derived using the AHP.

The pair-wise comparison matrices are developed to determine the weights of all criteria and sub-criteria. The weights for all the pairwise comparison matrices are then computed.

Intensity of Importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective.
3	Weak importance of one over another	Experience and judgment slightly favor one activity over another.
5	Essential or strong importance	Experience and judgment strongly favour one activity. over another.
7	Demonstrated importance	An activity is strongly favoured and its dominance demonstrated in practice.
9	Absolute importance	The evidence favouring one activity over another is of the highest possible order of affirmation.
2, 4, 6, 8	Intermediate values between the two adjacent judgements	When compromise is needed.
Reciprocals of above nonzero	If activity i has one of the above nonzero numbers assigned to it when compared with activity j, then j has the reciprocal value when compared with i.	

Table 2. Saaty's rating scale

	C_1	C_2	Weight
C_1	1	5	0.833
C_2	1/5	1	0.167
CR = 0			

Table 3. VM evaluation criterion comparison matrix

According to the definition, it is assumed that the cyclicity (C_2) has stronger importance compared with CPU usability (C_1). Thus, following reciprocal matrix is obtained.

$$\mathrm{G}_1 = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \left[\begin{array}{cc} 1 & 5 \\ & 1 \end{array} \right] \end{array}.$$

The reciprocal values of the upper diagonal are used to fill the lower triangular matrix. In other words, if g_{ij} is the element of row i column j of the matrix, then the lower diagonal is filled using $g_{ji} = \frac{1}{g_{ij}}$. Thus a complete comparison matrix is constructed.

$$\mathrm{G}_1 = \begin{array}{c} \\ C_1 \\ C_2 \end{array} \begin{array}{c} \begin{array}{cc} C_1 & C_2 \end{array} \\ \left[\begin{array}{cc} 1 & 5 \\ \frac{1}{5} & 1 \end{array} \right] \end{array}.$$

Notice that all the elements in the comparison matrix are positive ($a_{ij} > 0$).

Then by applying the same method to the rest of criteria, all other comparison matrices for VM evaluation can be obtained as follows:

$$\mathrm{C}_1 = \begin{array}{c} \\ C_{11} \\ C_{12} \end{array} \begin{array}{c} \begin{array}{cc} C_{11} & C_{12} \end{array} \\ \begin{bmatrix} 1 & 7 \\ \frac{1}{7} & 1 \end{bmatrix} \end{array}, \quad \mathrm{C}_2 = \begin{array}{c} \\ C_{21} \\ C_{22} \end{array} \begin{array}{c} \begin{array}{cc} C21 & C_{22} \end{array} \\ \begin{bmatrix} 1 & 5 \\ \frac{1}{5} & 1 \end{bmatrix} \end{array}.$$

Moreover, all the matrices for task usage evaluation can also be obtained.

$$\mathrm{G}_2 = \begin{array}{c|ccc} & B_1 & B_2 & B_3 \\ \hline B_1 & 1 & 7 & \frac{1}{2} \\ B_2 & 7 & 1 & \frac{1}{7} \\ B_3 & 2 & \frac{1}{7} & 1 \end{array}, \quad \mathrm{B1} = \begin{array}{c|cc} & B_{11} & B_{12} \\ \hline B_{11} & 1 & 6 \\ B_{12} & \frac{1}{6} & 1 \end{array}, \quad \mathrm{B}_2 = \begin{array}{c|cc} & B_{21} & B_{22} \\ \hline B_{21} & 1 & 6 \\ B_{22} & \frac{1}{6} & 1 \end{array}.$$

3.3 Priority Vectors

Having all the comparison matrices, next step is to compute the priority vector, which is the normalized eigenvector of the matrix. The method adopted for priority vectors calculation is the approximation of eigenvector (and eigenvalue) of a reciprocal matrix. This approximation has worked well for small matrix. Nevertheless, it is easy to compute because it only requires normalization of each column of the matrix.

Summing each column of the reciprocal matrix of G_1 results in:

$$\mathrm{G}_1 = \begin{array}{c|cc} & C_1 & C_2 \\ \hline C_1 & 1 & 5 \\ C_2 & \frac{1}{5} & 1 \\ sum & \frac{6}{5} & 6 \end{array}.$$

Then each element of the matrix is divided by the sum of its column to produce the normalized relative weight where the sum of each column is 1.

$$\mathrm{G}_1 = \begin{array}{c|cc} & C_1 & C_2 \\ \hline C_1 & \frac{5}{6} & \frac{5}{6} \\ C_2 & \frac{1}{6} & \frac{1}{6} \\ sum & 1 & 1 \end{array}.$$

The normalized principal eigenvector [24] can be obtained by averaging across the rows.

$$\omega = \frac{1}{2} \begin{bmatrix} \frac{5}{6} & + & \frac{5}{6} \\ \frac{1}{6} & + & \frac{1}{6} \end{bmatrix} = \begin{bmatrix} 0.833 \\ 0.167 \end{bmatrix}.$$

The normalized principal eigenvector is also called priority vector. Since it is normalized, the sum of all elements in the priority vector is 1. The priority vector shows relative weights among the evaluation indicators that are compared. In this example, the values of C_1 is 83.33 % and C_2 is 16.67 %.

Then, by calculating all the pair-wised criteria addressed in Table 1, the results are summarized in Tables 3, 4, 5, 6, 7, 8 accordingly, where all the CRs are the consistency ratio of each of the comparison matrix. If the value of CR is smaller or equal to 10 %, the inconsistency is acceptable. If the CR is greater than 10 %, the subjective judgment needs to be revised.

The Maximum CPU usage (C_{12}) has a higher importance compared to the CPU usage (C_1) in Table 4. Similarly in Table 5, cycles per instruction (C_{22}) has a higher significance than the CPU task duration (C_{21}).

C_1	C_{11}	C_{12}	Weight
C_{11}	1	7	0.729
C_{12}	1/7	1	0.104
CR = 0			

Table 4. C_{11} and C_{12} comparison matrix

C_2	C_{21}	C_{22}	Weight
C_{21}	1	5	0.139
C_{22}	1/5	1	0.028
CR = 0			

Table 5. C_{21} and C_{22} comparison matrix

	B_1	B_2	B_3	Weight
B_1	1	7	1/2	0.363
B_2	7	1	1/7	0.066
B_3	2	1/7	1	0.571
CR = 0.0519				

Table 6. Task evaluation criterion comparison matrix

Table 6 illustrates that the task validity (B_3) is of less importance compared to CPU usability (B_1). On the other hand, the task validity (B_3) has demonstrated a higher importance than cyclicity (B_2), while in Table 7, compared with CPU usage (B_{11}), Maximum CPU usage (B_{12}) has a higher importance. Moreover, in Table 8, cycles per instruction (B_{22}) has a higher significance than CPU task duration (B_{21}).

4 ALGORITHM DESCRIPTION

Let us assume for a particular VM, O is the vector of raw task usage data of VM i in n dimension and J is the evaluation vector. $T(O, J)$ is the task scoring vector.

B_1	B_{11}	B_{12}	Weight
B_{11}	1	6	0.311
B_{12}	1/6	1	0.052
CR = 0			

Table 7. B_{11} and B_{12} comparison matrix

B_2	B_{21}	B_{22}	Weight
B_{21}	1	6	0.057
B_{22}	1/6	1	0.010
CR = 0			

Table 8. B_{21} and B_{22} comparision matrix

Each vector (task) has m dimensional attributes. Thus the vector of attributes is denoted as $A = [A_1 \dots A_n]^T$, $A_i = [a_1 \dots a_i \dots a_m]$, then A is a $n \times m$ matrix. The relative weight value is $W^A = [w_1 \dots w_i \dots w_m]$. V is assumed as the AHP value, then $V = A \times W^A$. V is a n dimensional vector. Thus, $\max(V)$ needs to be calculated, in other words obtain V_i^* where task V_i is the most representative task and index i is the task number. For each VM the above operation is applied, then $VM_j = V_i$ is the typical task of the j^{th} VM. Assuming there are N VMs, then vector $VM = [VM_1 \dots VM_N]$ represents all the typical task vectors of all the VMs. Next, the attributes processing is performed. The weight value $W^{VM} = [w_1 \dots w_i \dots w_N]$, $V^{VM} = A \times W^{VM}$ is an N dimensional vector. Thus, by finding $\max\left(V^{VM}\right)$, V_i^{VM*} and related index i can be obtained resulting in VM_i as the target VM to migrate. The "AHP Decision Algorithm for VM Migration" is described in Algorithm 1.

Algorithm 1 AHP Decision Algorithm for VM Migration

```
 1: Setup VMs and attributes matrix;
 2: for Each VM indexed as $j$ in VMs set do
 3:     Compute the AHP value $V = A \times W^A$;
 4:     Find the index $i$ where $V_i = \max\{V\}$;
 5:     Set $VM_j = V_i$;
 6:     $j = j + 1$
 7: end for
 8: Compute $V^{VM} = A \times W^{VM}$;
 9: Find the index $i$ where $VM_i = \max\left\{V^{VM}\right\}$;
10: Return $i$;
```

5 SIMULATION RESULTS

This section demonstrates how the VM migration decisions are made using the proposed model. The simulation and experimental results also provide a feedback to identify the points where the model can be improved to make it more usable and flexible. The model has been applied to Google clusterdata-2011-1 dataset [25], particularly to task usage data part-00000-of-00500.

Score	VL	L	M	H	VH	Relative Weight
Very Low (VL)	1	3	5	7	9	0.513
Low (L)	1/3	1	3	5	7	0.261
Moderate (M)	1/5	1/3	1	3	5	0.129
High (H)	1/7	1/5	1/3	1	3	0.063
Very High (VH)	1/9	1/7	1/5	1/3	1	0.034

Table 9. Rating scale for C_{11}, C_{12}, C_{22}, B_{11}, B_{12} and B_{22}

Score	VL	L	M	H	VH	Relative Weight
Very Short (VS)	1	3	5	7	9	0.513
Short (S)	1/3	1	3	5	7	0.261
Moderate (M)	1/5	1/3	1	3	5	0.129
Long (L)	1/7	1/5	1/3	1	3	0.063
Very Long (VL)	1/9	1/7	1/5	1/3	1	0.034

Table 10. Rating scale for C_{21}, B_{21} and B_3

Before using the proposed model on the Google cluster dataset, Liberatore's [26] five-point rating scale is employed to rate each sub-factor of alternative VMs. It is better to reduce the time and effort in making pair-wise comparisons. Table 9 and Table 10 show the pair-wise comparison matrix of such rating scale. This matrix is then normalized to obtain the relative weight of each rating scale. The five-point rating factors are modified to use them for the measurement of the dataset. To normalize the raw data according to their values for the CPU rate, Maximum CPU rate and cycles per instruction attributes, the following scales are used: very-high, high, moderate, low and very-low. Similarly, for CPU task duration and task validity following scales are used: very-long, long, moderate, short and very-short. Then, the weights of very-high, high, moderate, low and very-low are calculated, which are equal to 0.513, 0.261, 0.129, 0.063 and 0.034, respectively. The weights of very-long, long, moderate, short and very-short are calculated in a similar manner. On the other hand, the attribute task validity is calculated in the same way with task duration.

As the next step, the dataset was preprocessed according to the proposed rating scale. The mean, maximum and minimum value are summarized according to the task usage data part-00000-of-00500. Moreover, five partitions are created for each attribute value range. A set of 5 VMs data is sampled from part-00000-of-00500. For each VM, 10 task usage samples are collected. Such dataset was sampled by using operators in Rapidminer 5.0. Consequently, the five-point ratings are assigned to the attribute with respect to the value of the raw data and corresponding partitions. Then the task usage of each VM is evaluated. Table 11 shows the task evaluation of

Criteria	Sub Criteria	Global Weights	Task 1 Score	GW	Task 2 Score	GW
B_1	B_{11}	0.311	VL = 0.513	0.160	VL = 0.513	0.160
	B_{12}	0.052	L = 0.261	0.014	VL = 0.513	0.027
B_2	B_{11}	0.057	VS = 0.513	0.029	VL = 0.034	0.002
	B_{11}	0.010	VL = 0.513	0.005	VL = 0.513	0.005
B_3		0.571	VL = 0.034	0.019	L = 0.063	0.359

Table 11. Single VM task usage evaluation

only one VM with only two tasks as the example. After all VMs been applied with such method, the most representative task usage information is obtained for each VM from 10 collected task usage samples.

Node ID	Task ID	Value
2994441279	10	0.497
587080532	10	0.474
17504375	10	0.411
4302816019	10	0.409
2568530361	9	0.367

Table 12. Task usage evaluation result

Criteria	Sub Criteria	Global Weight	Node 1 Score	GW	Node 2 Score	GW
C_1	C_{11}	0.729	VL = 0.513	0.160	VL = 0.513	0.160
	C_{12}	0.104	L = 0.261	0.014	VL = 0.513	0.027
C_2	C_{11}	0.139	VS = 0.513	0.029	VL = 0.034	0.002
	C_{11}	0.028	VL = 0.513	0.005	VL = 0.513	0.005

Table 13. VM evaluation

Table 12 shows the task with highest value after application of the proposed model. The task usage data is loaded into VM migration evaluation model. Table 13 shows the VM evaluation of VM with only two VMs as the example.

Obviously, after testing all the VMs against defined criteria, the VM with the highest value should be migrated first. Table 14 shows the related results: where VM with ID 2994441279 has the highest value. If it is assumed that all the tested VMs are on a same physical cloud server machine, VM 2994441279 and VM 587080532 shall be migrated as these two VMs has similar values. The similarity of the values may require other methods to further decide the migration; however, it is beyond the scope of this research topic.

It is observed that the usage of VM in a server obeys a normal distribution. That is, from 6:00 to 21:00 is the frequently used duration of the day, then from 21:00 to 6:00 is the infrequent period. Thus, normal distribution $\mathcal{N}(12, 0.7)$ is adopted as

VM ID	Task ID	Value
2994441279	10	0.478
17504375	10	0.282
4302816019	10	0.282
2568530361	9	0.179
587080532	10	0.426

Table 14. VM evaluation result

the probability distribution of usage scale of the physical machine, with 12 as the mean value and 0.7 as the standard deviation. Assuming there are 10 000 physical machines, with each having the maximal power of 500 W, the physical machine consumes 42 % electric power in infrequent period. The comparison of electric power consumption before and after migration is shown in Figure 5.

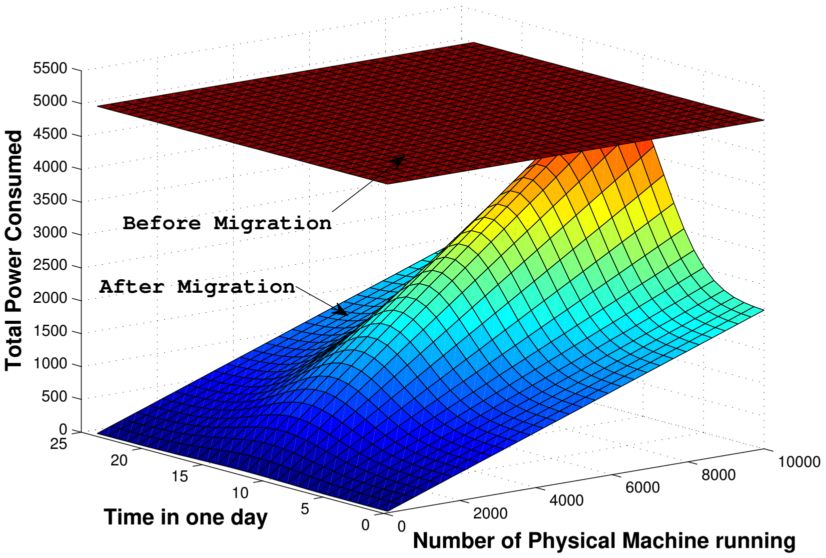


Figure 5. Power saving effectiveness verification

Assume t is the event when AHP algorithm is loaded, N is the total number of VMs, then the decision making consumes $t * N + t$ according to the proposed method. Thus Figure 6 shows the efficiency of the Algorithm 1.

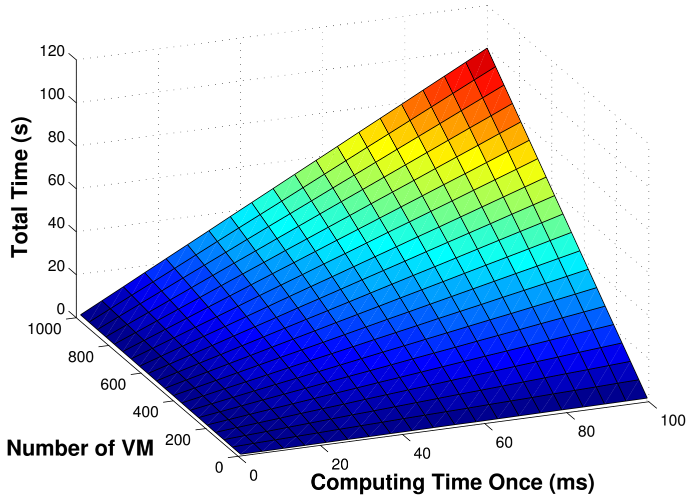


Figure 6. Efficiency comparison

6 CONCLUSIONS

This study has presented an AHP-based decision-making model to assist cloud cluster administrators in evaluating VM migration decisions. The paper investigates the applicability and efficiency of the proposed AHP model by conducting a usability study with 5 VMs usage data and then demonstrating how the model can be applied in a real application. Based on the simulated and experimental results, it can be concluded that the model can facilitate the decision making process and assist administrators to identify all information sources of input data for pair-wise comparisons. The pair-wise comparison procedure is able to capture relative judgments of two elements at once in a trustworthy manner and ensures consistency of these values. The results show that the model has the capability to identify the potential VMs that are stable and need to be migrated. This consolidation of the manner of green cloud computing means that it saves the energy and alleviates the carbon dioxide emissions. Therefore this is a great contribution towards the clean and safe global environment.

Acknowledgements

This work has been supported in part by the Program for Changjiang Scholars and Innovative Research Team in the University No. IRT1078, The Key Program of NSFC – Guangdong Union Foundation No. U1135002, The Fundamental Research Funds for the Central Universities No. JY0900120301.

REFERENCES

[1] Buyya, R.—Beloglazov, A.—Abawajy, J.: Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges. arXiv Preprint arXiv:1006.0308, 2010.

[2] Gong, L.—Xie, J.—Li, X.—Deng, B.: Study on Energy Saving Strategy and Evaluation Method of Green Cloud Computing System. 8th IEEE Conference on Industrial Electronics and Applications (ICIEA), 2013, pp. 483–488.

[3] Guazzone, M.—Anglano, C.—Canonico, M.: Exploiting VM Migration for the Automated Power and Performance Management of Green Cloud Computing Systems. Energy Efficient Data Centers (E2DC 2012). Springer, Lecture Notes in Computer Science, Vol. 7396, 2012, pp. 81–92, doi: 10.1007/978-3-642-33645-4_8.

[4] Iturriaga, S.—Nesmachnow, S.—Dorronsorro, B.—Bouvry, P.: Energy Efficient Scheduling in Heterogeneous Systems with a Parallel Multiobjective Local Search. Computing and Informatics, Vol. 32, 2013, No. 2, pp. 273–294.

[5] Zhang, L.-M.—Ma, J.-F.—Wang, Y.-C.—Lu, D.: Toward Green Cloud Computing: An Attribute Clustering Based Collaborative Filtering Method for Virtual Machine Migration. Information Technology Journal, Vol. 12, 2013, No. 23, pp. 7275–7279.

[6] Baliga, J.—Ayre, R. W. A.—Hinton, K.—Tucker, R. S.: Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport. Proceedings of the IEEE, Vol. 99, 2011, No. 1, pp. 149–167, doi: 10.1109/JPROC.2010.2060451.

[7] Saaty, T. L.: Fundamentals of Decision Making and Priority Theory with the Analytic Hierarchy Process. The Analytic Hierarchy Process Series, Vol. 6, Pittsburgh, 1994.

[8] Partovi, F. Y.: Determining What to Benchmark: An Analytic Hierarchy Process Approach. International Journal of Operations and Production Management, Vol. 14, 1994, No. 6, pp. 25–39, doi: 10.1108/01443579410062068.

[9] Jain, R.—Rao, B.: Application of AHP Tool for Decision Making of Choice of Technology for Extraction of Anti-Cancer Bioactive Compounds of Plant Origin. International Journal of the Analytic Hierarchy Process, Vol. 5, 2013, No. 1, pp. 3–29, doi: 10.13033/ijahp.v5i1.153.

[10] Uçal Sari, I.—Öztayşi, B.—Kahraman, C.: Fuzzy Analytic Hierarchy Process Using Type-2 Fuzzy Sets: An Application to Warehouse Location Selection. Chapter 12. In: Doumpos, M., Grigoroudis, E. (Eds.): Multicriteria Decision Aid and Artificial Intelligence. John Wiley & Sons, 2013, pp. 285–308.

[11] Cay, T.—Uyan, M.: Evaluation of Reallocation Criteria in Land Consolidation Studies Using the Analytic Hierarchy Process (AHP). Land Use Policy, Vol. 30, 2013, No. 1, pp. 541–548.

[12] Chan, W. W.—Yueng, S.—Chan, E. S.—Li, D.: Hotel Heat Pump Hot Water Systems: Impact Assessment and Analytic Hierarchy Process. International Journal of Contemporary Hospitality Management, Vol. 25, 2013, No. 3, pp. 428–446.

[13] Pourghasemi, H. R.—Moradi, H. R.—Fatemi Aghda, S. M.: Landslide Susceptibility Mapping by Binary Logistic Regression, Analytical Hierarchy Process, and

Statistical Index Models and Assessment of Their Performances. Natural Hazards, Vol. 69, 2013, No. 1, pp. 749–779, doi: 10.1007/s11069-013-0728-5.

[14] Roig-Tierno, N.—Baviera-Puig, A.—Buitrago-Vera, J.—Mas-Verdu, F.: The Retail Site Location Decision Process Using GIS and the Analytical Hierarchy Process. Applied Geography, Vol. 40, 2013, pp. 191–198, doi: 10.1016/j.apgeog.2013.03.005.

[15] Borade, A. B.—Kannan, G.—Bansod, S. V.: Analytical Hierarchy Process-Based Framework for VMI Adoption. International Journal of Production Research, Vol. 51, 2013, No. 4, pp. 963–978, doi: 10.1080/00207543.2011.650795.

[16] Ergu, D.—Kou, G.—Peng, Y.—Shi, Y.—Shi, Y.: The Analytic Hierarchy Process: Task Scheduling and Resource Allocation in Cloud Computing Environment. The Journal of Supercomputing, Vol. 64, 2013, No. 3, pp. 835–848.

[17] Garg, S. K.—Versteeg, S.—Buyya, R.: A Framework for Ranking of Cloud Computing Services. Future Generation Computer Systems, Vol. 29, 2013, No. 4, pp. 1012–1023.

[18] Harnisch, S.—Buxmann, P.: Evaluating Cloud Services Using Methods of Supplier Selection. Business Information Systems (BIS 2013). Springer, Lecture Notes in Business Information Processing, Vol. 157, 2013, pp. 1–13, doi: 10.1007/978-3-642-38366-3_1.

[19] Wang, J.—Li, F.—Zhang, L.: QoS PreferenceAwareness Task Scheduling Based on PSO and AHP Methods. International Journal of Control and Automation, Vol. 7, 2014, No. 4, pp. 137–152.

[20] De Steiguer, J. E.—Duberstein, J.—Lopes, V.: The Analytic Hierarchy Process as a Means for Integrated Watershed Management. In: Renard, K. G. (Ed.): First Interagency Conference on Research on the Watersheds, 2003, pp. 736–740.

[21] Forman, E. H.—Gass, S. I.: The Analytic Hierarchy Process – An Exposition. Operations Research, Vol. 49, 2001, No. 4, pp. 469–486, doi: 10.1287/opre.49.4.469.11231.

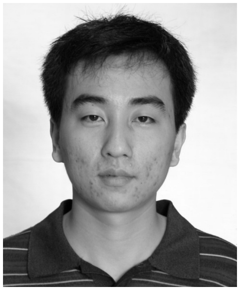
[22] Haq, A. N.—Kannan, G.: Fuzzy Analytical Hierarchy Process for Evaluating and Selecting a Vendor in a Supply Chain Model. The International Journal of Advanced Manufacturing Technology, Vol. 29, 2006, No. 7-8, pp. 826–835.

[23] Saaty, T. L.: How to Make a Decision: The Analytic Hierarchy Process. European Journal of Operational Research, Vol. 48, 1990, No. 1, pp. 9–26.

[24] Osman, S. M.—Gadalla, H. M.—Zeanedean, A. R.—Rabie, M. R.: A Compromise Weighted Solution for Multilevel Linear Programming Problems. International Journal of Engineering Research and Applications, Vol. 3, 2013, pp. 927–936.

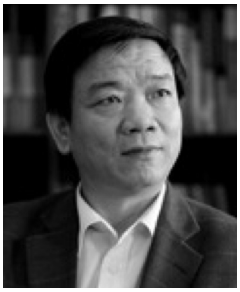
[25] Reiss, C.—Wilkes, J.—Hellerstein, J.: Google Cluster-Usage Traces: Format+Schema. Google Inc., White Paper, 2011.

[26] Liberatore, M. J.—Nydick, R. L.—Sanchez, P. M.: The Evaluation of Research Papers (Or How to Get an Academic Committee to Agree on Something). Interfaces, Vol. 22, 1992, No. 2, pp. 92–100, doi: 10.1287/inte.22.2.92.



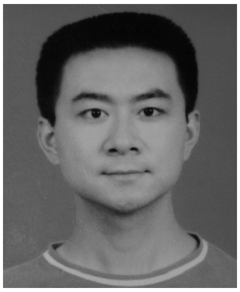
Liumei Zhang received his B.Eng. in computer science from Air-Force Engineering University, Xi'an, Shaanxi, China, in 2003, and his M.I.T. degree in database management from The School of Information Technology, University of Sydney, Sydney, Australia, in 2007. Currently he is a Ph.D. candidate in computer architecture at Xidian University, Xi'an, Shaanxi, China. His research interests include data mining, wireless sensor networks, and cloud computing.

Jianfeng Ma received his B.Sc. degree in mathematics from Shaanxi Normal University, China in 1985, and his M.Eng. and Ph.D. degrees in computer software and communications engineering from Xidian University, China in 1988 and 1995, respectively. From 1999 to 2001, he was with Nanyang Technological University of Singapore as a research fellow. He is an IEEE member and a senior member of Chinese Institute of Electronics (CIE). Currently he is Professor and Ph.D. supervisor in School of Computer Science at Xidian University, Xi'an, China, and the director of the Shaanxi Key Laboratory of Computer Networks and System Security. His current research interests include distributed systems, wireless and mobile computing systems, computer networks, and information and network security. He has published over 150 refereed articles in these areas and coauthored ten books.



Tianshi Liu received his B.Sc. degree from the Northwest University, China in 1982, his M.Eng. degree from Xi'an Jiaotong University, China in 1985, and the Ph.D. degree from Northwestern Polytechnical University, China in 2005. From 1994 to 1995 he was engaged in advanced studies in the Delft University of Technology. From 1985 to 1996, he was with the Institute of Computing Technology of Aviation Industry Corporation. He is a member of China Computer Society. In 2008, he was with School of Computer Science, The University of Birmingham as a research fellow. Currently he is Professor and Master Degree supervisor in School of Computer Science at the Xi'an Shiyou University, Xi'an, China. His current research interests include distributed systems, information system, computer networks, and communication optimization algorithm.

Yichuan Wang received his B.Sc. degree in computer science and technology from the Zhengzhou University of China in 2005, and his M.Sc. degree in computer and its applications from the Chengdu University of Information Technology, China in 2009. Currently he is a Ph.D. candidate with the School of Computer Science and Technology, Xidian University. His research areas include key management and networks security.



Di Lu received his B.Sc. and M.Sc. degrees in computer science and technology from the Xidian University, China in 2006 and 2009, respectively. Currently he is a Ph.D. student in the School of Computer Science and Technology, Xidian University. His research interests include virtualization technology, operating system and storage technology.

Computing and Informatics, Vol. 37, 2018, 311–326, doi: 10.4149/cai_2018_2_311

LOAD BALANCING SCHEDULING ALGORITHM FOR CONCURRENT WORKFLOW

Sunita Singhal

Manipal University Jaipur, School of Computing & Information Technology
Jaipur, Rajasthan, 303007, India
e-mail: `sunita.singhal@jaipur.manipal.edu`

Jemin Patel

Birla Institute of Technology & Science, Pilani
&
Department of Computer Science & Information Systems
Pilani Campus, Pilani, Rajasthan, 333031, India

Abstract. Concurrent workflow scheduling algorithm works in three phases, namely rank computation, tasks selection, and resource selection. In this paper, we introduce a new ranking algorithm that computes the rank of a task, based on its successor rank and its predecessors average communication time, instead of its successors rank. The advantage of this ranking algorithm is that two dependent tasks are assigned to the same machine and as a result the scheduled length is reduced. The task selection phase selects a ready task from each workflow and creates a task pool. The resource selection phase initially assigns tasks using min-min heuristic, after initial assignment, tasks are moved from the highly loaded machines to the lightly loaded machines. Our resource selection algorithm increases the load balance among the resources due to tasks assignment heuristic and reassignment of tasks from the highly loaded machines. The simulation results show that our proposed scheduling algorithm performs better over existing approaches in terms of load balance, makespan and turnaround time.

Keywords: Rank computation, multiple workflow, scheduling algorithm, task allocation, concurrent workflow

1 INTRODUCTION

Heterogeneous computing systems consist of a heterogeneous collection of machines, connected over a high-speed network, communication protocols and programming environments which provide a variety of architectural capabilities that have a diverse execution requirements. Cluster computing, grid computing, and cloud computing [1] are examples of heterogeneous computing systems. One of the key challenges of heterogeneous computing systems is the scheduling problem. In heterogeneous computing system a user submits an application to the system, an application profiling tool [2] is used to extract the properties of an application. The application properties include a number of jobs forming the workflow, the size of each job in terms of the number of instructions and the number of bytes required to be exchanged between the two jobs in case of parent and child relationship between jobs, relationship (parent and child) among the tasks in a workflow. Analogical Benchmarking [3, 4] provides a measure of how well a resource can perform a given type of application. On the basis of available information and quality of service requirement, the scheduler schedules tasks to the available machines in the system.

A workflow scheduling problem is an NP-complete problem and it is solved using static scheduling algorithms, dynamic scheduling algorithms, single workflow and multiple workflow scheduling algorithms. Static scheduling algorithms [5] assume that all the information about tasks, and resources are known and remain constant. Resources performance are varying in dynamic scheduling algorithm [14]. Single workflow scheduling algorithms [13] schedule a single workflow at a time while multiple workflow scheduling algorithms [15] schedule more than one workflow at a time.

The multiple workflow scheduling algorithm is also known as concurrent workflow scheduling algorithm. The concurrent workflow scheduling algorithm works in three phases known as rank computation, task selection and resource selection. We have worked on rank computation and resource selection phases. Our rank computation algorithm computes the rank of a task, based on its successor rank and its predecessors average communication cost. Our resource selection algorithm assigns tasks to resources using Min-min [5] heuristic. After that, tasks are reassigned from the highly loaded machines to the lightly loaded machines.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 presents the workflow model. Section 4 describes our proposed algorithm named Load balancing scheduling algorithm for concurrent workflow. The results are presented and discussed in Section 5. Finally, in Section 6 we conclude the work and suggest future research directions.

2 RELATED WORK

A multiple workflow scheduling algorithm is categorized as offline and online scheduling algorithms [7]. The offline workflow scheduling algorithms assume that all the workflows are available before the scheduling starts. The online workflow scheduling algorithms assume that the workflow arrival time is not known in advance.

The two most popular ranking algorithms are Heterogeneous Earliest Finish Time (HEFT) [6] and Critical Path on a Processor (CPOP) [6] ranking algorithms. The HEFT ranking algorithm computes the rank of each task, on the basis of average communication and computation cost between the current task and its successor. The CPOP ranking algorithm uses the summation of the downward and upward rank of a task. CPOP ranking algorithm does not perform better than HEFT [6] ranking algorithm. The HEFT ranking algorithm does not consider communication time of its immediate predecessor. As a result, two dependent tasks might be assigned to different resources and increase the scheduled length.

Zhao and Sakellariou [7] presented offline multiple workflow scheduling algorithm for composition and fairness of workflow. Their composition scheduling algorithm composes many workflows into a single workflow. The tasks are selected from each workflow in a round robin fashion and assigned to the resource that completes earliest. Their fairness scheduling algorithm computes the task's rank based on the slowness of the workflow. Our work differs from their approaches. We worked on concurrent workflow where the arrival time of workflow was not known.

Various online workflow scheduling algorithms are designed by researchers and are known as the Rank Hybrid [9] scheduling algorithm, the Online Workflow Management (OWM) [15] scheduling algorithm and the Fair Share [8] scheduling algorithm.

The Rank hybrid scheduling algorithm first computes the rank of each task using HEFT ranking algorithm. After ranking, all the ready tasks are selected from the workflow and assigned to the machines in increasing order of rank. Therefore, it gives priority to the smaller workflow. While the OWM scheduling algorithm selects a single task from each workflow, instead of selecting all the ready tasks from each workflow, the OWM scheduling algorithm assigns tasks to the machine that takes minimum time instead of the machine that completes first. It means if the minimum time machine is not free, it keeps the tasks in a waiting queue. Therefore, the lowest priority task may be executed first. The Fair share scheduling algorithm also selects a single task from each workflow, after that the rank of each task is again computed based on the percentage of the remaining task of a workflow. The highest rank task is assigned first. Thus, it does not differentiate between longer and smaller workflow.

All these scheduling algorithms compute the rank using HEFT ranking algorithm and assign tasks to the machine using Minimum Completion Time (MCT) [5] scheduling algorithm, while our ranking algorithm considers its successor rank and its predecessors average communication time. After ranking the tasks of a workflow, a tasks pool is created, then these tasks are initially assigned using Min-min heuristic. After initial assignment of tasks, they are reassigned from the highly loaded machines to the lightly loaded machines. Our reassignment algorithm is also different from the Balance Minimum Completion Time (BMCT) [12] resource selection algorithm. The BMCT resource selection algorithm moves tasks from the highest completion time machine to other machines in the system. Thus, it may transfer tasks to the machine that is not lightly loaded. As a result, the load may not be balanced.

3 WORKFLOW MODEL

A typical scientific application is represented using Directed Acyclic Graph (DAG) to model an application. A sample DAG of 10 tasks is shown in Figure 1 a). A workflow is represented by G $= (v, e, p, w)$, where v and e are the set of tasks and directed edges, respectively. A node in the task graph represents a task that runs non-preemptively on any cluster. Each edge is denoted by e_{ij} corresponding to the data communication between t_i and t_j, where t_i is called the immediate parent task of t_j. A child task cannot be started until all of its parent tasks are completed. A task which does not have a parent task is called an entry task t_{entry}. A task that does not have a child task is called an exit task t_{exit}. w is a $v \times p$ computation matrix, where v is the number of tasks and p is the number of processors in the system. w_{ij} is the estimated execution time of task t_i on processor p_j. t_{pred} and t_{succ} are the sets of immediate predecessors and successors of task t_i, respectively. The average computation time of task t_i is calculated as follows:

$$w_i = \frac{\sum_{j \in P} w_{ij}}{p}. \tag{1}$$

Here w_i is the average computation time of t_i, p is the number of processor on which task can be executed, and P is the set of processors on which task can be executed. w_{ij} is the expected execution time of t_i on processor p_j. The average communication cost between edge i and edge j is defined as follows:

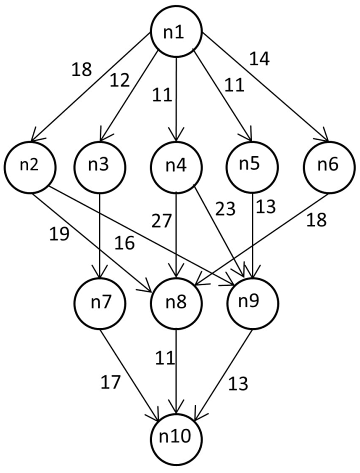
$$c_{ij} = \frac{L + data_{ij}}{B}. \tag{2}$$

Here c_{ij} is the average communication cost between task i and j, L is the average communication start-up cost of all processors and B is the average bandwidth of all links connecting to P. $data_{ij}$ is the amount of data that task t_i transfers to task t_j, when tasks are assigned to different processors.

The resources are heterogeneous computers, connected to each other.

4 LOAD BALANCING SCHEDULING ALGORITHM FOR CONCURRENT WORKFLOW

This algorithm works in three phases as shown in Figure 2. The first phase computes the rank of the tasks. The second phase selects a single ready task from each workflow in order to provide fairness among available workflow. The last phase first assigns the tasks using Min-min heuristic then computes the average load of the system. Based on the average load of the system, tasks are transferred from the



(a) DAG

Processor → / Task ↓	P1	P2	P3
n1	14	16	9
n2	13	19	18
n3	11	13	19
n4	13	8	17
n5	12	13	10
n6	13	16	9
n7	7	15	11
n8	5	11	14
n9	18	12	20
n10	21	7	16

(b) Computation Matrix

Figure 1. DAG and computation matrix

highly loaded machines to the lightly loaded machines in order to reduce the load imbalance.

4.1 Rank Computation Phase

Our rank computation algorithm is called Modified Heterogeneous Earliest Finish Time (MHEFT) ranking algorithm. The MHEFT ranking algorithm computes the rank from downwards. t_{exit} task rank is defined as follows:

$$\text{rank}(t_{exit}) = w_{exit} + \max_{t_k \in t_{pred}\{t_{exit}\}} c(exit, k). \tag{3}$$

Here t_{pred} is the set of predecessors of t_{exit}, $c(exit, k)$ is the average communication cost between task t_{exit} and t_k, and w_{exit} is the average computation cost of task t_{exit}. The rank of a task t_i is recursively defined and computed as follows:

$$\text{rank}(t_i) = w_i + \max_{t_j \in t_{succ}\{t_i\}} (\ c(i,j) + \text{rank}(t_j)) + \max_{t_k \in t_{pred}\{t_i\}} c(i,k). \tag{4}$$

Here t_{succ} is the set of immediate successors of t_i, w_i is the average computation cost of task t_i. $c(i,j)$ is the average communication cost between task t_i and t_j, t_j is the successor of t_i, and $c(i,k)$ is the average communication cost between task t_i and t_k. t_{pred} is the set of predecessors of t_i.

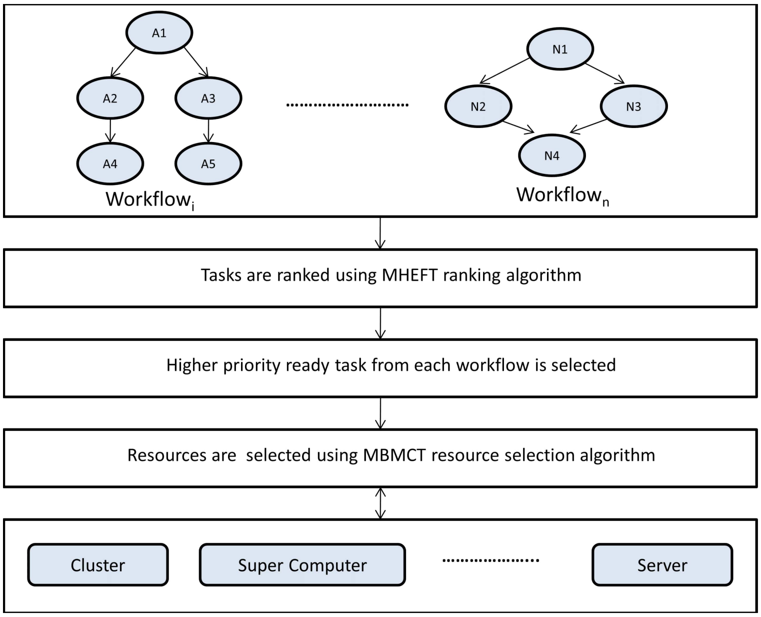

A1
A2
A3
A4
A5
Workflow$_i$
N1
N2
N3
N4
Workflow$_n$
Tasks are ranked using MHEFT ranking algorithm
Higher priority ready task from each workflow is selected
Resources are selected using MBMCT resource selection algorithm
Cluster
Super Computer
Server

Figure 2. Pictorial diagram of load balancing concurrent workflow scheduling algorithm

4.1.1 Illustrative Example

Consider a sample DAG of 10 tasks and the computation matrix are given in Figures 1 a) and 1 b), respectively. An identical matrix is used by Topcuoglu et al. [6] to show the working of HEFT ranking algorithm. It has 10 tasks and 3 processors. Computed rank and Gantt chart of HEFT, CPOP and MHEFT ranking algorithms are shown in Table 1 and Figure 3, respectively.

Task	HEFT	CPOP	MHEFT
n1	108	108	108
n2	77	108	95
n3	79	105	91
n4	80	104	91
n5	69	93	80
n6	63	90	77
n7	42	105	65
n8	35	102	62
n9	44	108	67
n10	14	108	31

Table 1. Rank of scheduling algorithms

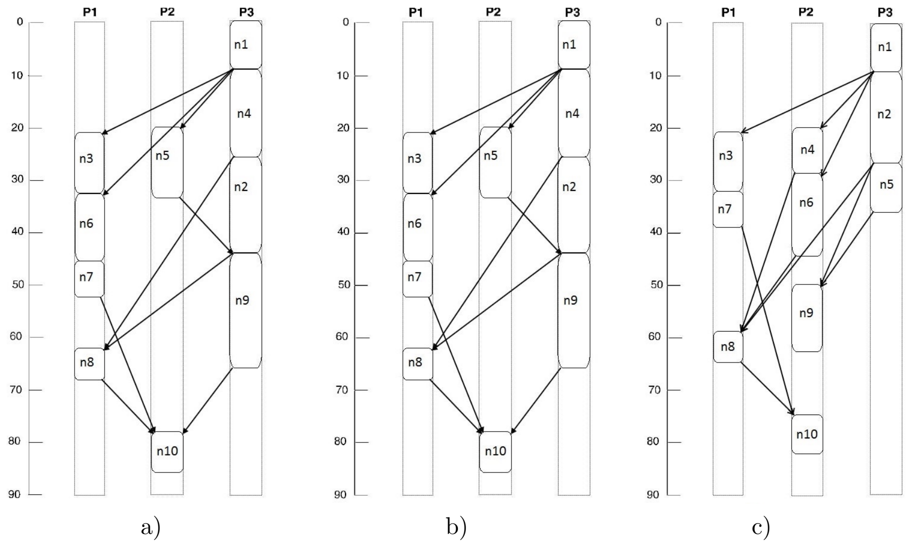


Figure 3. Gantt chart of ranking. a) HEFT ranking algorithm (schedule length = 86) b) CPOP ranking algorithm (schedule length = 86) c) MHEFT ranking algorithm (schedule length = 82).

The scheduled length of MHEFT ranking algorithm is 82 in this example, while CPOP and HEFT ranking algorithms scheduled length is 86 in both cases. The HEFT ranking algorithm computes the rank as follows:

$$\mathrm{rank}(t_i) = w_i + \max_{t_j \in t_{succ}\{t_i\}} (c(i,j) + \mathrm{rank}(t_j)). \tag{5}$$

The Critical Path on a Processor (CPOP) ranking Algorithm computes the rank as follows:

$$\mathrm{rank}(t_i) = \max_{t_j \in t_{pred}\{t_i\}} (w_j + \ c(i,j) + \mathrm{rank}(t_j)). \tag{6}$$

4.2 Task Selection Phase

This phase selects the highest priority ready task from each workflow and a task pool is created.

4.3 Resource Selection Phase

Our resource selection phase algorithm is called the Modified Balance Minimum Completion Time (MBMCT) resource selection algorithm. The MBMCT resource selection algorithm pseudo code is shown in Algorithm 1.

The MBMCT resource selection algorithm first finds the initial assignment of tasks using Min-min heuristic. Min-min heuristic first assigns the task that has the minimum expected execution time then updates the remaining task's expected execution time, this step is repeated till all the tasks are assigned. After completion of the initial assignment, the average makespan is computed (line 5). Based on the average makespan, the machines are categorized into lightly loaded, moderately loaded and highly loaded machines (line 6–7). The moderately loaded machines load is close to ($\pm\delta$) average load of the system. The lightly loaded machines load is less than moderately loaded machines. The highly loaded machines load is greater than the average load of the system. The reassignment procedure is called till makespan is changed (line 8–12).

The reassignment procedure finds a task that can be transferred from the highly loaded machines to the lightly loaded machines. As a result, the load imbalance is decreased among machines. The pseudo code of the reassignment procedure is shown in Algorithm 2.

The reassignment procedure finds a task which can be reassigned from M_i machine (a machine that belongs to the highly loaded machines) to the lightly loaded machine. Then, it computes the completion time of M_i machine and lightly loaded machine (line 5–6). If completion time of M_i machine and lightly loaded machine is less than the makespan, the makespan is updated (line 7–9). It also keeps information about the machine and the task, where and which task will be moved (line 10–11). At the end, it returns the updated makespan (line 17).

Algorithm 1 Pseudo code of MBMCT resource selection algorithm

```
1:  Select ready task from each workflow;
2:  Assign tasks to machine using Min-min;
3:  repeat
4:      $makeSpan \leftarrow$ computeMakespan();
5:      $avrgMakeSpan \leftarrow$ computeAverage();
6:      $machLow \leftarrow findLowMachine(avrgMakeSpan, \delta)$;
7:      $machHigh \leftarrow findHighMachine(avrgMakeSpan, \delta)$;
8:      $newMakeSpan \leftarrow reAssignment(machLow, machHigh, makeSpan)$;
9:      if $newMakeSpan < makeSpan$ then
10:         moveTask();
11:     end if
12: until $newMakeSpan < makeSpan$;
```

The time complexity of MBMCT resource selection algorithm is $O(k * m * n^2)$. Where k is the number of iterations when the makespan value change, m is the number of machines, and n is the number of tasks.

5 SIMULATION AND RESULTS

In this section, we discuss how to generate workflow, what are the performance parameters, and compare our results with existing algorithms. A Java based simulator is designed.

Algorithm 2 Pseudo code of reassignment procedure

```
1: for all $M_i \in machHigh$ do
2:   for all $M_j \in machLow$ do
3:     for $T_k \in M_i$ do
4:       $comTime'' \leftarrow computeFinishTime(M_j)$;
5:       $comTime'' \leftarrow comTime'' +$ expected execution time of $T_k$ on $M_j$;
6:       $comTime' \leftarrow makeSpan -$ expected execution time of $T_k$ on $M_i$;
7:       if $comTime'' < makeSpan$ then
8:         if $comTime' < makeSpan$ then
9:           $makeSpan \leftarrow comTime''$;
10:          $task \leftarrow T_k$;
11:          $machine \leftarrow M_j$;
12:        end if
13:      end if
14:    end for
15:  end for
16: end for
17: return $makeSpan$
```

5.1 Workflow Generation

Random workflows are generated based on a given approach [6]. The parameters to generate a workflow are the number of nodes, the average computation cost, the computation to communication ratio, and the arrival rate is shown in Table 2. The average computation cost denotes an average number of instructions required to execute a task, the computation and communication ratio denotes the average data transfer required between two tasks, the heterogeneity factor denotes the heterogeneity in processor speed, and the shape decides the out degree of nodes in a workflow. The arrival rate is simulated as the corresponding percentage of tasks completed from the recently arrived workflow. For example, 10 % of the inter-arrival time means workflow X will be inserted after completion of 10 % tasks of workflow Y.

Parameter Name	**Range**
Number of nodes	30 to 100
Average computation cost	10 to 200
Computation to communication ratio	0.1 to 10
Heterogeneity factor of resources	0.2 to 0.5
Shape	0.1 to 0.45
Arrival rate	10 to 50
Number of processors	3 to 30

Table 2. Workflow parameters

5.2 Performance Parameters

The performance parameters are Makespan, Schedule Length Ratio (SLR), Load balance, and Turnaround Time Ratio (TTR). Makespan is the difference between execution start time and execution finish time of a workflow. The turnaround time is the super-set of the makespan because it also includes the waiting time of workflow. TTR is the ratio of turnaround time to makespan. The SLR is a normalized makespan based on the critical path. SLR is defined as follows:

$$SLR = \frac{Makespan}{\sum_{t_i \in cp_{min}} \min_{p_j \in P}(w_i, j)}. \tag{7}$$

The denominator in SLR is the minimum computation cost of the critical path tasks (cp_{min}). There is no makespan less than the denominator of the SLR equation. Therefore, the algorithm with lowest SLR is the best algorithm. Average SLR values over several workflows are used in our experiments. Load balance of resource$_i$ is computed as follows:

$$LoadBal_i = \sum_{j \in K} (Makespan_j - ResUti_i). \tag{8}$$

Here K is the set of workflows. Makespan$_j$ is the makespan of workflow$_j$ and $ResUti_i$ is the sum of total time spent by a resource on execution of all assigned tasks of a workflow$_j$. Average load balance is defined as follows:

$$AvgLoadBal = \frac{\sum_{i \in P} LoadBal_i}{p}. \tag{9}$$

5.3 Simulation Results and Analysis

The value of δ is selected $\pm 5\,\%$ of makespan based on experiment. The results presented are averaged out over thirty simulations of each type of workflow. Figures 4 and 5 show the results of different numbers of resources for average makespan and average SLR, respectively. In each case, we have generated 100 workflows of 50 nodes. Workflows are a combination of different communication and computation ratio. At the x-axis, it has a number of resources. In Figure 4, it can be observed that MHEFT ranking algorithm average makespan is less than HEFT ranking algorithm. The difference of average makespan increases as the number of resources are increased. A similar trend is followed in average SLR, as shown in Figure 5. Even when the number of resources is less the MHEFT ranking algorithm's average SLR is better than the HEFT ranking algorithm.

Figures 6 and 7 present the results of different inter-arrival time for the number of iterations and average makespan, respectively. In Figure 7, it can be observed that MBMCT resource selection algorithm's average makespan is less than BMCT resource selection algorithm in each inter-arrival time. The number of iterations of

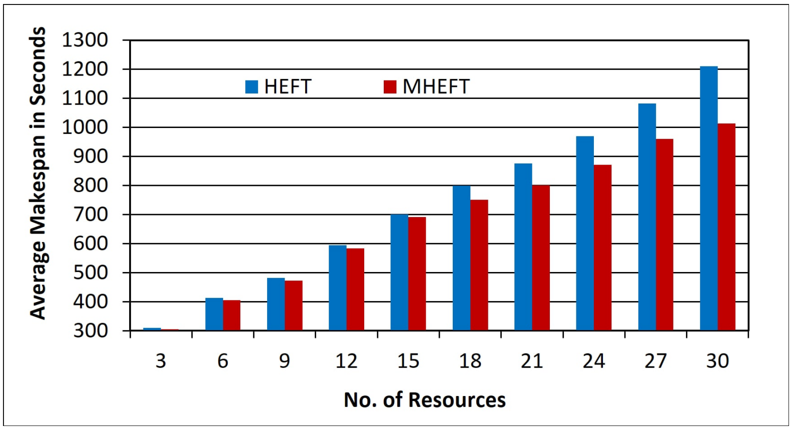


Figure 4. Results of different number of resources for average makespan

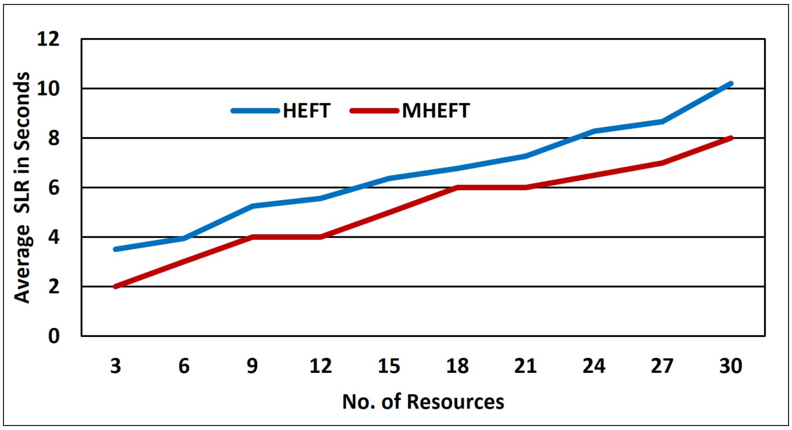


Figure 5. Results of different number of resources for average schedule length ratio

MBMCT resource selection algorithm are slightly greater than the BMCT resource selection algorithm because MBMCT resource selection algorithm moves a task to the lightly loaded machine, while BMCT resource selection algorithm may transfer a task to the moderately loaded machine.

In Figures 8, 9 and 10, LB* scheduling algorithm is the combination of MHEFT ranking algorithm and MBMCT resource selection algorithm. LB scheduling algorithm is the combination of HEFT ranking algorithm and MBMCT resource selection algorithm. To assess the proposed algorithms, we have generated 100 workflows of 70 nodes, that is the combination of different average computation cost and communication ratio. The average computation and communication ratio is randomly chosen from a particular set. In all the performance parameters, LB scheduling al-

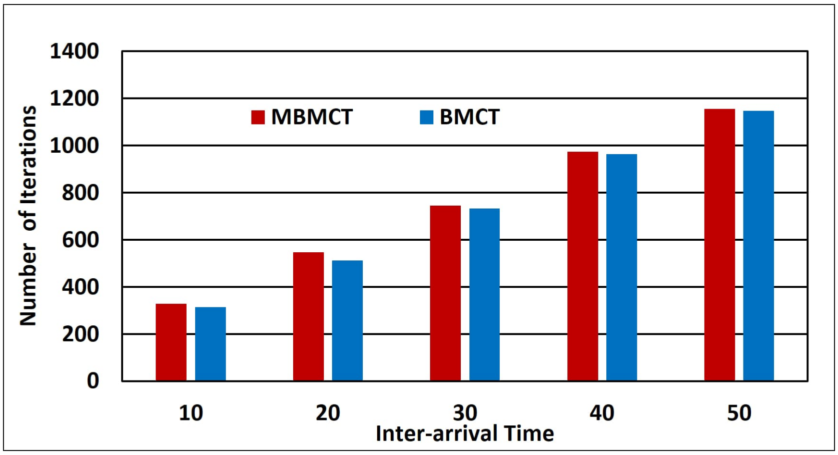


Figure 6. Results of different inter-arrival time for number of iterations

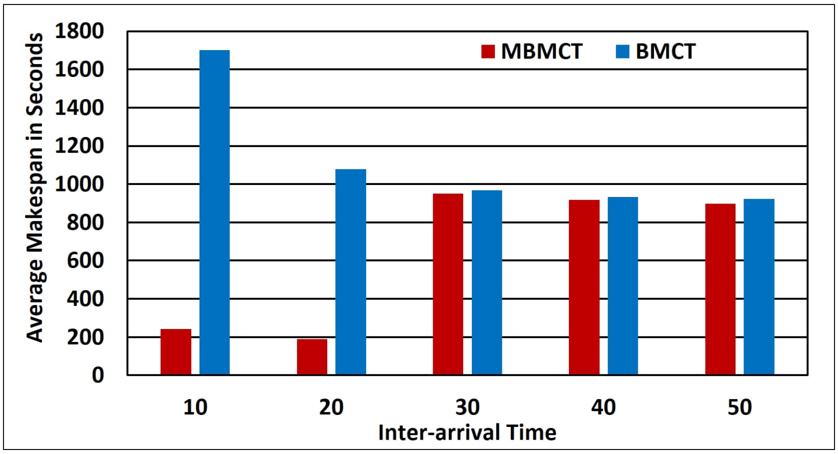


Figure 7. Results of different inter-arrival time for average makespan

gorithm performs better or equally to existing algorithms. It performs better when the inter-arrival time is shorter because in shorter inter-arrival time more number of tasks are scheduled thus, more move is possible. LB* scheduling algorithm performs better than LB scheduling algorithm due to better ranking of tasks in a workflow.

Figure 8 displays the results of different inter-arrival time for average makespan. The LB* scheduling algorithm average makespan is 10 % of the Fair share scheduling algorithm in case of 10 and 20 inter-arrival time. The overall average makespan of LB* scheduling algorithm is 15 % better than the LB scheduling algorithm.

Figure 9 demonstrates the results of different inter-arrival time for average load balance. LB* scheduling algorithm average load balance is 47 %, 37 %, 41 %, and

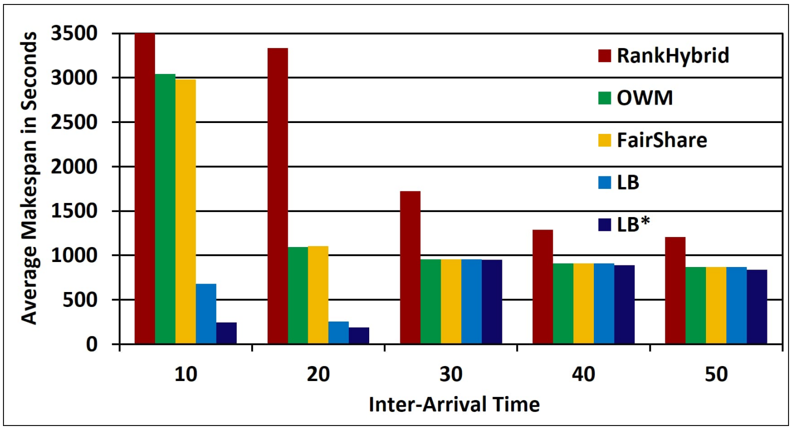


Figure 8. Results of different inter-arrival time for average makespan

26 % less than the LB, the Fair share, the OWM and the Rank hybrid scheduling algorithm, respectively.

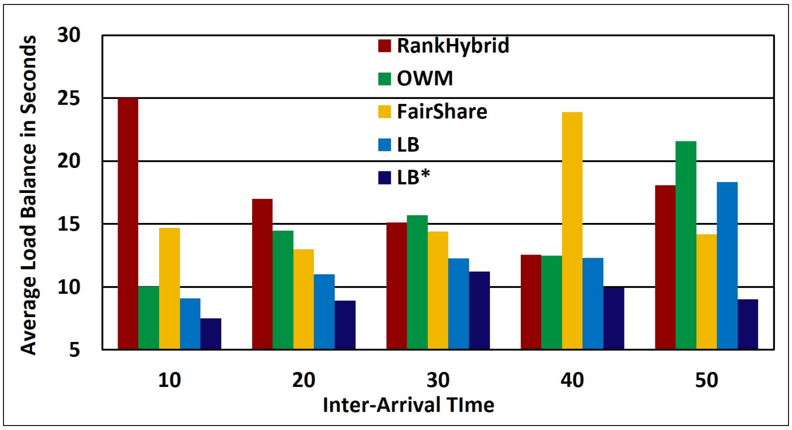


Figure 9. Results of different inter-arrival time for average load balance

Figure 10 shows the results of different inter-arrival time for average TTR. The average TTR of LB* scheduling algorithm is 8 %, 50 %, 50 %, and 73 % less than the LB, the Fair share, the OWM and the Rank hybrid scheduling algorithm respectively.

6 CONCLUSION

We have proposed the load balancing scheduling algorithm for the concurrent workflow that is the combination of modified heterogeneous earliest finish time ranking

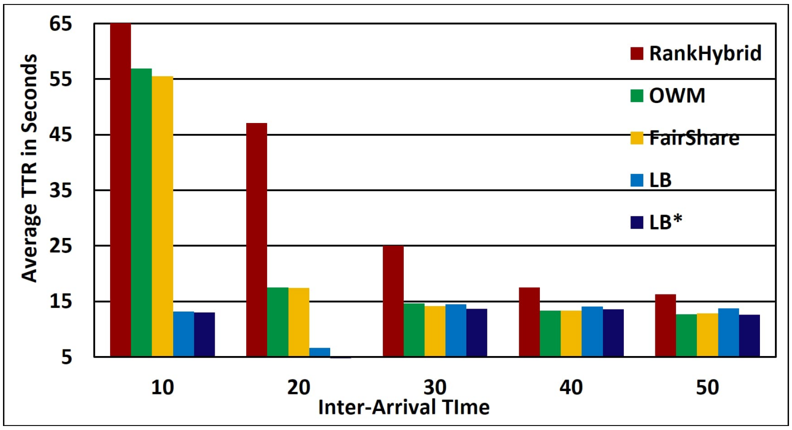


Figure 10. Results of different inter-arrival time for average turnaround time ratio

algorithm and modified balance minimum completion time resource selection algorithm. The modified heterogeneous earliest finish time ranking algorithm considers a communication time of parent task that plays a significant role when tasks are communication intensive in a workflow. Similarly, the modified balance minimum completion time resource selection algorithm transfers the tasks between the highly loaded machines to the lightly loaded machines instead of transfer tasks to all the machines in the system. As a result, faster load balance is achieved in a shorter time. The load balancing scheduling algorithm performs better than other existing scheduling algorithms in a shorter inter-arrival time. In future, we intend to test our algorithm on the real large-scale workflow and heterogeneous environment.

REFERENCES

[1] Foster, I.—Kesselman, C.: The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, USA, 1999.

[2] Hoschek, W.—Jaen-Martinez, J.—Samar, A.—Stockinger, H.—Stockinger, K.: Data Management in an International Data Grid Project. Proceedings of the First International Workshop on Grid Computing (GRID 2000), Bangalore, India, 2000. Lecture Notes in Computer Science, Vol. 1971, 2000, pp. 77–90, doi: 10.1007/3-540-44444-0_8.

[3] Wolski, R.—Spring, N. T.—Hayes, J.: Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing. Future Generation Computer Systems, Vol. 15, 1999, No. 5-6, pp. 757–768, doi: 10.1016/S0167-739X(99)00025-4.

[4] Gong, L. G.—Sun, X. H.—Watson, E. F.: Performance Modeling and Prediction of Nondedicated Network Computing. IEEE Transactions on Computers, Vol. 51, 2002, No. 9, pp. 1041–1055.

[5] Braun, T. D.—Siegel, H. J.—Maciejewski, A. A.—Hong, Y.: Static Resource Allocation for Heterogeneous Computing Environments with Tasks Having Dependencies, Priorities, Deadlines, and Multiple Versions. Journal of Parallel and Distributed Computing, Vol. 68, 2008, No. 11, pp. 1504–1516, doi: 10.1016/j.jpdc.2008.06.006.

[6] Topcuoglu, H.—Hariri, S.—Wu, M.-Y.: Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems, Vol. 13, 2002, No. 3, pp. 260–274, doi: 10.1109/71.993206.

[7] Zhao, Y.—Wilde, M.—Foster, I.—Voeckler, J.—Dobson, J.—Gilbert, E.—Jordan, T.—Quigg, E.: Virtual Data Grid Middleware Services for Data-Intensive Science. Concurrency and Computation: Practice and Experience, Vol. 18, 2006, No. 6, pp. 595–608.

[8] Arabnejad, H.—Barbosa, J.: Fairness Resource Sharing for Dynamic Workflow Scheduling on Heterogeneous Systems. Proceedings of the 10th IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2012, pp. 633–639, doi: 10.1109/ISPA.2012.94.

[9] Yu, Z.—Shi, W.: A Planner-Guided Scheduling Strategy for Multiple Workflow Applications. Proceedings of the IEEE International Conference on Parallel Processing-Workshops (ICPP-W '08), 2008, pp. 1–8.

[10] Zhao, H.—Sakellariou, R.: Scheduling Multiple DAGs onto Heterogeneous Systems. Proceedings of the 20th International Parallel and Distributed Processing Symposium (IPDPS), Rhodes Island, Greece, 2006, pp. 1–14, doi: 10.1109/IPDPS.2006.1639387.

[11] Bansal, S.—Hota, C.: Efficient Refinery Scheduling Heuristic in Heterogeneous Computing Systems. Journal of Advances in Information Technology, Vol. 2, 2011, No. 3, pp. 159–164, doi: 10.4304/jait.2.3.159-164.

[12] Sakellariou, R.—Zhao, H.: A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. Proceedings of the 18th International Parallel and Distributed Processing Symposium (IPDPS), Santa Fe, New Mexico, USA, 2004, pp. 1–13, doi: 10.1109/IPDPS.2004.1303065.

[13] Yu, J.—Buyya, R.—Ramamohanarao, K.: Workflow Scheduling Algorithms for Grid Computing. In: Xhafa, F., Abraham, A. (Eds.): Metaheuristics for Scheduling in Distributed Computing Environments. Springer, Studies in Computational Intelligence, Vol. 146, 2008, pp. 173–214.

[14] Alam, T.—Raza, Z.: A Dynamic Load Balancing Strategy with Adaptive Thresholds DLBAT for Parallel Computing System. International Journal of Distributed Systems and Technologies, Vol. 5, 2014, No. 1, pp. 54–69.

[15] Hsu, C.-C.—Huang, K.-C.—Wang, F.-J.: Online Scheduling of Workflow Applications in Grid Environments. Future Generation Computer Systems, Vol. 27, 2011, No. 6, pp. 860–870.

Sunita Singhal has been working as Associate Professor, School of Computing and Information Technology of Manipal University Jaipur, India since 2016. Before that she worked at the Birla Institute of Technology and Science (BITS), Pilani Campus, Pilani, India from 2005. She earned her Ph.D. degree from the BITS Pilani in computer science engineering. Her current research focuses on cloud computing, high performance computing, internet of things and distributed computing. She has published several papers in international conferences and journals over the past few years. She acted as a member of the organizing committee and program committee of many international conferences and workshops.

Jemin Patel earned his M.Eng. degree in software systems from the Birla Institute of Technology and Science (BITS), Pilani Campus, Pilani, India. He received his B.Tech. degree in computer engineering from the Dharmsinh Desai University, Nadiad, Gujarat, India in 2013. He is currently doing his internship at Amazon. His research interests include distributed computing and cloud computing.

Computing and Informatics, Vol. 37, 2018, 327–348, doi: 10.4149/cai_2018_2_327

FINE-GRAINED ACCESS CONTROL SYSTEMS SUITABLE FOR RESOURCE-CONSTRAINED USERS IN CLOUD COMPUTING

Yinghui Zhang*, Dong Zheng

National Engineering Laboratory for Wireless Security
Xi'an University of Posts and Telecommunications
Xi'an 710121, P.R. China
&
State Key Laboratory of Cryptology
P.O. Box 5159, Beijing 100878, P.R. China
e-mail: yhzhaang@163.com, zhengdong@xupt.edu.cn

Rui Guo, Qinglan Zhao

National Engineering Laboratory for Wireless Security
Xi'an University of Posts and Telecommunications
Xi'an 710121, P.R. China
e-mail: guorui@xupt.edu.cn, zhaoqinglan@foxmail.com

Abstract. For the sake of practicability of cloud computing, fine-grained data access is frequently required in the sense that users with different attributes should be granted different levels of access privileges. However, most of existing access control solutions are not suitable for resource-constrained users because of large computation costs, which linearly increase with the complexity of access policies. In this paper, we present an access control system based on ciphertext-policy attribute-based encryption. The proposed access control system enjoys constant computation cost and is proven secure in the random oracle model under the decision Bilinear Diffie-Hellman Exponent assumption. Our access control system supports AND-gate access policies with multiple values and wildcards, and it can efficiently support direct user revocation. Performance comparisons indicate that the proposed solution is suitable for resource-constrained environment.

* Corresponding author

Keywords: Attribute-based encryption, constant computation, access control, revocation, cloud computing

1 INTRODUCTION

Cloud computing is a promising computing paradigm in which vast and scalable resources are provided as services over the Internet. As the development of cloud computing, users' concerns about data security become main obstacles that impede cloud computing from wide adoptions. However, traditional access control technologies are no longer suitable for cloud computing environment because the service provider is fully trusted by users. In cloud computing, the data service manager is not trusted by users and is assumed to be honest-but-curious, that is, it will honestly execute the tasks assigned by legitimate participants in the system. Meanwhile, it would like to learn secret information as much as possible.

Attribute-based encryption (ABE) is known as an important tool for implementing secure and fine-grained access control over untrusted cloud storage. In an attribute-based system, access control over encrypted data is closely related to attributes, which are used to describe users in the system and define fine-grained access policies. There are two kinds of ABE constructions: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In a KP-ABE scheme, the access policy is necessary for generation of attribute secret keys and ciphertexts are computed based on a set of attributes. In CP-ABE, access policies are used to generate ciphertexts and every secret key is corresponding to an attribute set. A user can successfully decrypt a ciphertext only if the attribute set associated with the user's secret key matches the access policy specified for the ciphertext by data owners.

Although promising in designing fine-grained access control system in cloud computing, efficiency challenges still remain there before ABE can be widely deployed in practical cloud platforms. Specifically, the computation cost of most existing ABE schemes linearly grows with the complexity of the access formula. The drawback appears more serious in resource-constrained scenarios such as wireless sensors and mobile phones. Moreover, an efficient revocation mechanism is necessary in secure and scalable ABE systems.

Aiming at tackling the challenges described above, we present a fine-grained data access control system in cloud computing, where CP-ABE serves as a fundamental building block. The proposed access control system enjoys small and constant computation cost and it can efficiently support direct user revocation. It is proved that the system is secure against adaptively chosen ciphertexts attacks (CCA2) in the random oracle model under the decision m-BDHE assumption, where m is an upper bound of the total number of users in the system. Particularly, our access control system enables AND-gate access policy with multiple attribute values and wildcards. Performance comparisons indicate that our solution is suitable for real-world application scenarios where users are resource-constrained.

2 RELATED WORK

The notion of ABE was introduced by Sahai and Waters [1] as a fuzzy identity-based encryption, and was firstly dealt with by Goyal et al. [2]. There are two complementary notions of ABE: KP-ABE and CP-ABE. The first KP-ABE construction [2] realized the monotonic access structure for key policies, while the first CP-ABE scheme supporting tree-based access structures in generic group models was proposed by Bethencourt et al. [3]. To achieve enhanced security, Cheung and Newport [4] presented a CP-ABE scheme supporting AND-gate policy with positive and negative attribute values and wildcards in the standard model.

However, most existing ABE schemes are inefficient because the computation overhead is linearly proportional to the complexity of access policies. In many real-world application scenarios, users often have constrained computing power [5, 6, 7, 8, 9, 10, 11] and cannot afford the computation cost of many previous ABE solutions. Therefore, computationally efficient ABE schemes [12, 13, 14] have received a lot of attention recently. Emura et al. [13] proposed a CP-ABE scheme, which only needs three exponentiation and two pairing operations in encryption and decryption phases, respectively. However, the scheme only supports AND-gate policies with multiple values without wildcards. If a decryptor's attributes are different from the access policy, he/she cannot decrypt corresponding ciphertexts. Similarly, the CP-ABE scheme [12] suffers the disadvantage [13] in that it supports AND-gate policies with single positive value without wildcards. Chen et al. [14] proposed two CP-ABE constructions, which support AND-gate policies with positive and negative attribute values and wildcards. Zhang et al. [15] proposed a CP-ABE scheme supporting AND-gate policies with multiple attribute values and wildcards. Very recently, Li et al. [16] have proposed a data deduplication technique suitable for cloud storage. As a promising technique, data deduplication has been widely adopted in cloud storage to save storage resources and bandwidth. There are also many other ABE constructions, such as multi-authority ABE [17, 18], outsourced ABE [19, 20, 21], anonymous ABE [22, 23, 24, 25], and traceable ABE [26, 27, 28], etc. In the multi-authority ABE scheme [17], any polynomial number of attribute authorities are allowed to control attributes and issue attribute secret keys. It is worth noting that these authorities are mutually independent. A data owner can encrypt his/her data so as to a user can decrypt the corresponding ciphertext only if he/she has particular attributes controlled by an attribute authority. Certainly, multi-authority ABE schemes must resist the collusion attacks of multiple malicious attribute authorities. In outsourced ABE schemes, intensive computing tasks during encryption and decryption phases are outsourced to cloud servers without revealing any private data or secret keys. Outsourced ABE has wide applications considering the mobile cloud computing environment. It enables resource-constrained users to complete heavy computation tasks with the help of cloud servers. Different from traditional ABE schemes, outsourced ABE has to introduce corresponding outsourcing servers. In the outsourced ABE scheme [20], secure outsourced decryption and key distribution are realized. In anonymous ABE schemes, access policies information is not revealed in ciphertexts. Therefore, any people cannot learn of policy information from ciphertexts, and even legitimate decryptors fail to guess what access policies are adopted by encryptors. Anonymous ABE can realize users' privacy protection and hence it has a wide range of applications. Particularly, in the anonymous ABE scheme [25], a novel technique called match-then-decrypt is proposed, in which a matching phase is additionally introduced before the anonymous decryption phase. The match-then-decrypt technique can significantly improve the efficiency in decryption phase of anonymous ABE. ABE with attribute hierarchies further realize the expressiveness of access policies.

Furthermore, efficient revocation mechanisms are indispensable for ABE schemes in that some secret keys might get compromised at some point. Yu et al. [29] proposed a CP-ABE scheme supporting immediate attribute revocation mechanism with the help of a semi-trusted proxy server. Yang et al. [30] proposed an attribute revocation method to cope with the dynamic changes of users' access privileges. However, all the above ABE schemes only support indirect revocation, that is, the attribute authority indirectly enables revocation by forcing revoked users to be unable to update their secret keys. Direct revocation enjoys a desirable property that revocation can be done without affecting any non-involved users. Attrapadung et al. [31] suggested two directly user-revocable CP-ABE schemes by combining the techniques of ABE and broadcast encryption (BE). Since Fiat et al. [32] first introduced the notion of BE, Boneh et al. [33] proposed a collusion resistant BE scheme, which features short ciphertexts and private keys and is adopted in our construction to realize direct user revocation. There are many other researches on revocation, offline computation and policy update [34, 35, 36]. However, the computation cost of the above schemes linearly increases with the complexity of access structures and the number of revoked users.

3 PRELIMINARIES

3.1 Cryptographic Background

Definition 1 (Bilinear pairing)**.** Let $\mathbb{G}$ be a cyclic multiplicative group of a prime order p, $g \in_R \mathbb{G}$ be a generator, and $\mathbb{G}_T$ be a cyclic multiplicative group of the same order, identity of which we denote as 1. We call $\hat{e}$ a bilinear pairing if $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ is a map with the following properties:

1. Bilinear: $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$ for all $a, b \in \mathbb{Z}_p^*$.
2. Non-degenerate: There exists $g_1, g_2 \in \mathbb{G}$ such that $\hat{e}(g_1, g_2) \neq 1$.
3. Computable: $\hat{e}(g_1, g_2)$ can be efficiently computed for all $g_1, g_2 \in \mathbb{G}$.

Definition 2 (Decision (t, ϵ, ℓ)-BDHE assumption)**.** Security of our construction is based on a complexity assumption called the Bilinear Diffie-Hellman Exponent assumption (BDHE). Let $\mathbb{G}$ be a bilinear group of prime order p, and g, h two independent generators of $\mathbb{G}$. Denote $\overrightarrow{y}_{g,\alpha,\ell} = (g_1, g_2, \cdots, g_\ell, g_{\ell+2}, \cdots, g_{2\ell}) \in \mathbb{G}^{2\ell-1}$, where $g_i = g^{(\alpha^i)}$ for some unknown $\alpha \in \mathbb{Z}_p^*$. An algorithm $\mathcal{B}$ that outputs $\mu \in \{0,1\}$ has advantage ϵ in solving the decision ℓ-BDHE problem if

$$|\Pr[\mathcal{B}(g, h, \overrightarrow{y}_{g,\alpha,\ell}, \hat{e}(g_{\ell+1}, h)) = 1] - \Pr[\mathcal{B}(g, h, \overrightarrow{y}_{g,\alpha,\ell}, Z) = 1]| \geq \epsilon.$$

We say that the decision (t, ϵ, ℓ)-BDHE assumption holds in $\mathbb{G}$ if no t-time algorithm has advantage at least ϵ in solving the decision ℓ-BDHE problem in $\mathbb{G}$.

3.2 Access Policy

Usually, notation $L \models W$ is used to represent the fact that the attribute list L satisfies the access policy W, and the case of L does not satisfy W is denoted by $L \not\models W$. In our construction, we consider AND-gate policy supporting multiple attribute values and wildcards, which is a generalization of the access policy in [4] and is also adopted in [23]. Formally, given an attribute list $L = [L_1, L_2, \cdots, L_n]$ and an access policy $W = [W_1, W_2, \cdots, W_n] = \bigwedge_{i \in \mathcal{I}_W} W_i$, where $\mathcal{I}_W$ is a subscript index set and $\mathcal{I}_W = \{i | 1 \leq i \leq n, W_i \neq *\}$, we say $L \models W$ if $L_i = W_i$ or $W_i = *$ for all $1 \leq i \leq n$ and $L \not\models W$ otherwise. Note that the wildcard $*$ in W plays the role of "do not care" value.

4 SYSTEM ARCHITECTURE AND ADVERSARY MODEL

4.1 System Architecture

As shown in Figure 1, the system architecture of access control in cloud computing consists of four entities AA (Attribute Authority), CSP (Cloud Service Provider), DO (Data Owner), and DU (Data User).

AA is an entity who generates public parameters and master secret keys for the system. It is in charge of issuing attribute secret keys for users. It is fully trusted by all entities joining the system.

CSP is an entity that hosts the encrypted files of DO. It consists of cloud storage servers and a data service manager. Encrypted files from data owners are stored in cloud storage servers. The data service manager is in charge of controlling the accesses from outside users to the encrypted files.

DO is an entity who owns files, and wishes to upload them to the cloud storage servers provided by CSP. It is responsible for defining attribute-based access policy, and enforcing it on its own files by encrypting the files under the access policy before uploading them.

DU is an entity who intends to access the encrypted files hosted in the cloud storage servers. If DU is not revoked and his/her attributes match the underlying access policy in the encrypted files specified by DO, then he/she will succeed in decrypting the encrypted files.

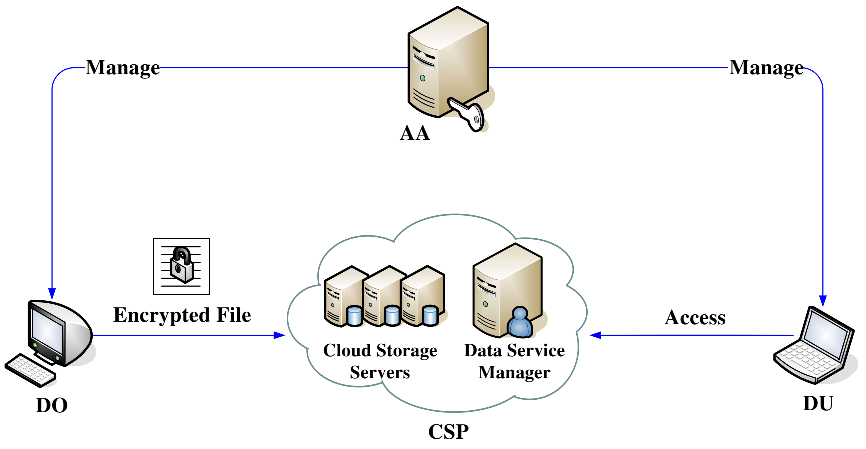


Figure 1. System architecture of access control in cloud computing

We give an overview of access control in cloud computing.

System Setup. AA generates public parameters and master secret keys for the system, and keeps master secret keys secretly.

User Registration. When a user wants to join the system, AA issues attribute secret keys to him/her based on his/her attributes.

New File Creation. When DO wants to share a file with some users, he/she encrypts the file under a specific access policy and uploads the resulted ciphertext to CSP.

File Access. When DU wants to access an outsourced file, he/she downloads the ciphertext from CSP and decrypts it.

4.2 Adversary Model and Security Goals

Similar to the previous systems, CSP is assumed to be honest-but-curious. In our system, the adversary is modeled as users colluding with CSP. The security goal is semantic security of data and it is reflected in the following three security requirements.

Data Confidentiality. Unauthorized DU who does not have enough attributes matching the access policy specified for a ciphertext by DO should be prevented from accessing the plaintext of the files. In addition, unauthorized access from CSP to the plaintext of the encrypted files should also be prevented.

Collusion-Resistance. If multiple DU and CSP collude, they may be able to access the plaintext of an encrypted file by combining attributes even if each of them cannot decrypt encrypted files alone. In practical attribute-based data sharing systems, these colluders should not succeed in decrypting encrypted files.

Revocation. Any user involved in a revocation event fails to access the plaintext of subsequent ciphertexts exchanged after he/she is revoked from the system.

5 BUILDING BLOCK CP-ABE

5.1 Definition of CP-ABE

A CP-ABE scheme consists of the following four algorithms:

Setup$(1^\lambda) \rightarrow (PK, MK)$**:** On input a security parameter λ, it returns the system public key PK which is distributed to DO and DU, and the master key MK which is kept private.

KeyGen$(PK, MK, L) \rightarrow SK_L$**:** On input the system public key PK, the master key MK and an attribute list L, it outputs SK_L as the attribute secret key associated with L.

Encrypt$(PK, M, W, \mathcal{R}) \rightarrow CT_W$**:** On input the system public key PK, a message M, an access policy W specified by DO and a revocation set $\mathcal{R}$ issued by AA, it generates a ciphertext CT_W as the encryption of M with respect to W and $\mathcal{R}$, which is outsourced to CSP. Note that $\mathcal{R}$ specifies the users who are revoked from the system.

Decrypt$(PK, CT_W, SK_L) \rightarrow M$ **or** $\perp$**:** On input the system public key PK, a ciphertext CT_W of a message M under W and $\mathcal{R}$, and a secret key SK_L associated with L, it outputs the message M if the user is not revoked and $L \models W$, and the error symbol $\perp$ otherwise.

5.2 Formalized Security Models for CP-ABE

In the proof of our construction, we adopt a security model called indistinguishability against selective ciphertext-policy and adaptively chosen-ciphertext attacks (IND-sCP-CCA2), which is demonstrated in the following IND-sCP-CCA2 game.

Init: The adversary $\mathcal{A}$ commits to a challenge ciphertext policy W^* and a revocation information set $\mathcal{R}^*$.

Setup: The challenger $\mathcal{S}$ chooses a sufficiently large security parameter λ, and runs the Setup algorithm to get a master key SK and the corresponding system public key PK. It retains SK and gives PK to $\mathcal{A}$.

Phase 1: In addition to hash queries, $\mathcal{A}$ issues a polynomially bounded number of queries to the following oracles:

- **KeyGen Oracle** $\mathcal{O}_{KeyGen}$: $\mathcal{A}$ submits an attribute list L, if $L \not\models W^*$, $\mathcal{S}$ gives $\mathcal{A}$ the secret key SK_L and outputs $\perp$ otherwise.

- **Decrypt Oracle** $\mathcal{O}_{Dec}$: $\mathcal{A}$ submits a ciphertext CT_W of a message M. If CT_W is well-formed, $\mathcal{S}$ returns the message M. Otherwise, $\perp$ is returned.

Challenge: Once $\mathcal{A}$ decides that **Phase 1** is over, it outputs two equal length messages M_0 and M_1 from the message space, on which it wishes to be challenged with respect to W^* and $\mathcal{R}^*$. The challenger $\mathcal{S}$ randomly chooses a bit $b \in \{0,1\}$, computes $CT_{W^*} = \mathsf{Encrypt}(PK, M_b, W^*, \mathcal{R}^*)$ and sends CT_{W^*} to $\mathcal{A}$.

Phase 2: The same as **Phase 1**, except that CT_{W^*} may not be submitted for oracle $\mathcal{O}_{Dec}$.

Guess: $\mathcal{A}$ outputs a guess bit $b' \in \{0,1\}$ and wins the game if $b' = b$. The advantage of $\mathcal{A}$ in the IND-sCP-CCA2 game is defined as follows:

$$\mathrm{Adv}_{\mathrm{CP-ABE}}^{\mathrm{IND-sCP-CCA2}}(\mathcal{A}) = \left|\Pr[b' = b] - \frac{1}{2}\right|.$$

Definition 3. A CP-ABE scheme is said to be IND-sCP-CCA2 secure if no probabilistic polynomial-time adversary can break the IND-sCP-CCA2 game with non-negligible advantage.

6 PROPOSED ACCESS CONTROL SYSTEM

6.1 Main Idea

In the proposed scheme, the decryption cost is constant and it does not linearly increase with the complexity of access policies. The scheme can support AND-gate access policies with multiple values and wildcards and it is IND-sCP-CCA2 secure. In order to realize constant decryption cost, we use the idea of ciphertext aggregation. That is, in the new file creation phase, DO generates ciphertext components by aggregating the system public key components which are specified by the attribute values in access policies. To allow authorized DU to decrypt ciphertexts, in the user registration phase, AA generates attribute secret key components for attribute values appeared in the attribute list of DU. In the file access phase, to successfully decrypt a ciphertext, DU just uses some attribute secret key components in his/her secret key which are specified by values of the access policy. In order to efficiently support AND-gate access policies with multiple values and wildcards, AA only chooses three master secret key components in the system setup phase. Then, for each attribute value, a system public key component is generated by binding the attribute index with a master secret key component based on hash functions. For the sake of CCA2 security, the last ciphertext component is generated from the first three ciphertext components based on a random factor and system public key components. Based on the bilinear pairing, the last ciphertext component helps to answer decryption queries in security proof.

6.2 Our Scheme

In this section, we present an access control system. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two cyclic multiplicative groups of a prime order p. Also, let g be a generator of $\mathbb{G}$ and $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map. Suppose the attribute set of the system is $\mathcal{U} = \{\omega_1, \omega_2, \cdots, \omega_n\}$. Attribute ω_i has n_i values and $S_i = \{v_{i,1}, v_{i,2}, \cdots, v_{i,n_i}\}$ is the multi-value set. Define collision-resistant hash functions $H_0 : \mathbb{Z}_p^* \times \{0,1\}^* \to \mathbb{Z}_p^*$, $H_1 : \mathbb{Z}_p^* \to \mathbb{G}$, and $\hat{H} : \{0,1\}^* \times \mathbb{G}_T \times \mathbb{G}^2 \to \mathbb{Z}_p^*$. The system is described as follows.

System Setup. AA chooses a security parameter λ and runs the following algorithm Setup of CP-ABE to generate a public parameter PK and a master secret key MK for the system. Then AA publishes PK and keeps MK secretly.

- Setup(1^λ): AA chooses $x, y \in_R \mathbb{Z}_p^*$, and computes $X_{i,k_i} = g^{-H_0(x||i||k_i)}$, $Y_{i,k_i} = \hat{e}(g,g)^{H_0(y||i||k_i)}$ for $1 \le i \le n$ and $1 \le k_i \le n_i$. It also chooses $\alpha, \beta \in_R \mathbb{Z}_p^*$ and sets $v = g^\beta$. Suppose the total number of users in the system is bounded above by some natural number m. For notational simplicity, we let $\mathcal{I}_m = \{1, 2, \cdots, m\}$ in the following. For $1 \le i \le 2m$ and $i \ne m+1$, AA computes $g_i = g^{(\alpha^i)}$. In addition, AA chooses $\delta_1, \delta_2, \delta_3 \in \mathbb{G}$. Finally, the system public key is published by AA as

$$PK = \langle g, \{X_{i,k_i}, Y_{i,k_i}\}_{1\le i\le n, 1\le k_i\le n_i}, \{g_i\}_{1\le i\le 2m, i\ne m+1}, v, \delta_1, \delta_2, \delta_3 \rangle,$$

  and the master key is $MK = \langle x, y, \beta \rangle$.

User Registration. When a user with an attribute list L wants to join the system, AA runs the following algorithm KeyGen of CP-ABE to obtain an attribute secret key SK_L and gives it to the user.

- KeyGen(PK, MK, L): AA chooses $sk \in_R \mathbb{Z}_p^*$ for the user. Then for $1 \le i \le n$, suppose $L_i = v_{i,k_i}$, AA computes

$$\overline{\sigma}_i = \sigma_{i,k_i} = g^{H_0(y||i||k_i)} H_1(sk)^{H_0(x||i||k_i)}.$$

  Also, AA computes $d = g_{sn}^\beta$, where $sn \in \{1, 2, \cdots, m\}$ is a serial number and it is used by AA to indicate that the current user is the sn^{th} one to join the system. Finally, the corresponding attribute secret key is $SK_L = \langle sn, sk, \{\overline{\sigma}_i\}_{1\le i\le n}, d \rangle$.

New File Creation. Whenever DO wants to upload a file $\mathcal{F}$ to CSP, he/she chooses a symmetric key K and encrypts $\mathcal{F}$ with K based on a typical symmetric encryption scheme such as AES to obtain a ciphertext CT_0. Then DO defines a ciphertext policy W for $\mathcal{F}$, and runs the following algorithm Encrypt of CP-ABE to encrypt K to get a ciphertext CT_W. Finally, DO sets $CT_{\mathcal{F}} = \{CT_0, CT_W\}$ and uploads $CT_{\mathcal{F}}$ to CSP.

- $\mathsf{Encrypt}(PK, M, W, \mathcal{R})$: Suppose $W_i = v_{i,k_i}$, in order to encrypt a message $M = K$ under a ciphertext policy $W = \bigwedge_{i \in \mathcal{I}_W} W_i$ such that the revoked users specified by $\mathcal{R}$ cannot access it, DO computes

$$\langle X_W, Y_W \rangle = \left\langle \prod_{i \in \mathcal{I}_W} \overline{X}_i, \prod_{i \in \mathcal{I}_W} \overline{Y}_i \right\rangle,$$

where $\langle \overline{X}_i, \overline{Y}_i \rangle = \langle X_{i,k_i}, Y_{i,k_i} \rangle$. Then, DO chooses $s, \hat{s} \in_R \mathbb{Z}_p^*$, computes $K_{\mathcal{R}} = \hat{e}(g_1, g_m)^s$ and $C_{\mathcal{R}} = \left(v \cdot \prod_{i \in \mathcal{I}_m - \mathcal{R}} g_{m+1-i}\right)^s$. It also computes $C_0 = MY_W^s K_{\mathcal{R}}$, $C_1 = g^s$, $C_2 = X_W^s$, $\hat{h} = \hat{H}(W||C_0||C_1||C_2)$, and $C_3 = (\delta_1^{\hat{h}} \delta_2^{\hat{s}} \delta_3)^s$. Finally, DO sets $CT_W = \langle C_0, C_1, C_2, C_3, C_{\mathcal{R}}, \hat{s} \rangle$. Note that the ciphertext policy W and revocation information $\mathcal{R}$ are implicitly included in ciphertexts.

File Access. Whenever DU with an attribute secret key SK_L wants to access and retrieve an outsourced file, he/she firstly downloads the ciphertext $CT_{\mathcal{F}} = \{CT_0, CT_W\}$ from CSP. Then DU computes $K = \mathsf{Decrypt}(PK, CT_W, SK_L)$ by running the following $\mathsf{Decrypt}$ algorithm, and then retrieves the file $\mathcal{F}$ by symmetric decryption based on K. It is worth noting that DU can successfully recover $\mathcal{F}$ if and only if $L \models W$ and he/she is not revoked.

- $\mathsf{Decrypt}(PK, CT_W, SK_L)$: Suppose $CT_W = \langle C_0, C_1, C_2, C_3, C_{\mathcal{R}}, \hat{s} \rangle$ corresponding to W and $\mathcal{R}$, and $W = \bigwedge_{i \in \mathcal{I}_W} W_i$ with $W_i = v_{i,k_i}$. Then CT_W is decrypted by DU with an attribute secret key $SK_L = \langle sn, sk, \{\overline{\sigma}_i\}_{1 \le i \le n}, d \rangle$ as follows. DU first checks whether $L \models W$ and $sn \notin \mathcal{R}$. If not, the decryption algorithm returns $\perp$. Otherwise, DU checks whether $\hat{e}(g, C_3) = \hat{e}(C_1, \delta_1^{\hat{h}} \delta_2^{\hat{s}} \delta_3)$ and $\hat{e}(g, C_2) = \hat{e}(C_1, X_W)$ or not, where $\hat{h} = \hat{H}(W||C_0||C_1||C_2)$ and $X_W = \prod_{i \in \mathcal{I}_W} X_{i,k_i}$. If one of the two equations does not hold, return $\perp$. Otherwise, DU computes $\sigma_W = \prod_{i \in \mathcal{I}_W} \overline{\sigma}_i$ and

$$K_{\mathcal{R}} = \frac{\hat{e}\left(g_{sn}, C_{\mathcal{R}}\right)}{\hat{e}\left(d \cdot \prod_{i \in \mathcal{I}_m - \mathcal{R}}^{i \neq sn} g_{m+1-i+sn}, C_1\right)}.$$

Finally, the message is recovered as $M = K = \frac{C_0}{\hat{e}(\sigma_W, C_1)\hat{e}(H_1(sk), C_2) K_{\mathcal{R}}}$.

The process of outsourcing and access is shown in Figure 2.

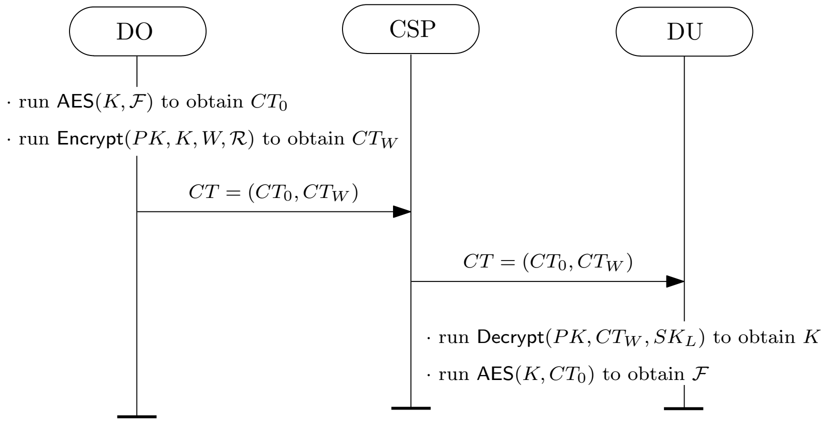

DO
CSP
DU
· run $\mathsf{AES}(K, \mathcal{F})$ to obtain CT_0
· run $\mathsf{Encrypt}(PK, K, W, \mathcal{R})$ to obtain CT_W
$CT = (CT_0, CT_W)$
$CT = (CT_0, CT_W)$
· run $\mathsf{Decrypt}(PK, CT_W, SK_L)$ to obtain K
· run $\mathsf{AES}(K, CT_0)$ to obtain $\mathcal{F}$

Figure 2. The process of outsourcing and access

7 ANALYSIS OF OUR ACCESS CONTROL SYSTEM

7.1 Correctness

If $L \models W$ and the user associated with sn is not revoked, the ciphertext can be successfully decrypted. Notice that $v = g^{\beta}$ and $d = g_{sn}^{\beta}$, we have

$$K_{\mathcal{R}} = \frac{\hat{e}\left(g_{sn}, C_{\mathcal{R}}\right)}{\hat{e}\left(d \cdot \prod_{i \in \mathcal{I}_m - \mathcal{R}_W}^{i \neq sn} g_{m+1-i+sn}, C_1\right)} = \frac{\hat{e}\left(g_{sn}, \left(v \cdot \prod_{i \in \mathcal{I}_m - \mathcal{R}_W} g_{m+1-i}\right)^s\right)}{\hat{e}\left(g_{sn}^{\beta} \cdot \prod_{i \in \mathcal{I}_m - \mathcal{R}_W}^{i \neq sn} g_{m+1-i+sn}, g^s\right)}$$
$$= \frac{\hat{e}\left(g, g\right)^{s\alpha^{sn} \sum_{i \in \mathcal{I}_m - \mathcal{R}_W} \alpha^{m+1-i}}}{\hat{e}\left(g, g\right)^{s \sum_{i \in \mathcal{I}_m - \mathcal{R}_W}^{i \neq sn} \alpha^{m+1-i+sn}}} = \hat{e}(g, g)^{\alpha^{m+1} s} = \hat{e}(g_1, g_m)^s.$$

Suppose the indexes satisfy $L_i = v_{i,k_i}$, then

$$\frac{C_0}{\hat{e}(\sigma_W, C_1)\hat{e}(H_1(sk), C_2)K_{\mathcal{R}}} = \frac{MY_W^s \hat{e}(g_1, g_m)^s}{\hat{e}\left(\sigma_W, g^s\right)\hat{e}\left(H_1(sk), X_W^s\right)K_{\mathcal{R}}}$$
$$= \frac{M(\prod_{i \in \mathcal{I}_W} \overline{Y}_i)^s}{\hat{e}\left(\prod_{i \in \mathcal{I}_W} \overline{\sigma}_i, g^s\right)\hat{e}\left(H_1(sk), \left(\prod_{i \in W} \overline{X}_i\right)^s\right)}$$
$$= \frac{M\left(\prod_{i \in \mathcal{I}_W} \hat{e}(g, g)^{H_0(y||i||k_i)}\right)^s}{\hat{e}\left(\prod_{i \in \mathcal{I}_W} g^{H_0(y||i||k_i)} H_1(sk)^{H_0(x||i||k_i)}, g^s\right)\hat{e}\left(H_1(sk), \left(\prod_{i \in \mathcal{I}_W} g^{-H_0(x||i||k_i)}\right)^s\right)}$$
$$= M.$$

7.2 Security Analysis

We only need to prove the building block CP-ABE scheme is semantically-secure, which is demonstrated in Theorem 1.

Theorem 1. Assume that $\mathcal{A}$ makes at most q_{H_1} queries to the random oracle H_1, at most q_K queries to the key generation oracle, and at most q_D queries to the decryption oracle. If the decision (τ, ϵ, m)-BDHE assumption holds in $\mathbb{G}$, then the proposed scheme is (τ', ϵ', m)-secure, where $\tau' = \tau + \mathcal{O}(q_{H_1} + mq_K + q_D + N)\tau_1 + \mathcal{O}(q_D + N)\tau_2 + \mathcal{O}(q_D)\tau_p$ with $N = \sum_{i=1}^{n} n_i$, and $\epsilon' = \epsilon - \frac{q_D}{p}$. Here, τ_1 and τ_2 denotes the time complexity to compute an exponentiation in $\mathbb{G}$ and $\mathbb{G}_T$, respectively. τ_p represents the time complexity of a pairing operation.

Proof. Suppose that there exists a τ-time adversary $\mathcal{A}$, which breaks the proposed scheme with $\mathrm{Adv}_{\mathrm{CP-ABE}}^{\mathrm{IND-sCP-CCA2}}(\mathcal{A}) \geq \epsilon$. We build a simulator $\mathcal{S}$ that has advantage ϵ in solving the decision m-BDHE problem in $\mathbb{G}$. $\mathcal{S}$ takes as input a random decision m-BDHE challenge $(g, h, \overrightarrow{y}_{g,\alpha,m}, Z)$, where $\overrightarrow{y}_{g,\alpha,m} = (g_1, g_2, \cdots, g_m, g_{m+2}, \cdots, g_{2m})$ and Z is either $\hat{e}(g_{m+1}, h)$ or a random element in $\mathbb{G}_T$. The simulator $\mathcal{S}$ plays the role of the challenger in the IND-sCP-CCA2 game, and interacts with $\mathcal{A}$ as follows.

Init. The simulator $\mathcal{S}$ receives a challenge access structure $W^* = \bigwedge_{i \in \mathcal{I}_{W^*}} W_i$ and a revocation information set $\mathcal{R}^*$ specified by the adversary $\mathcal{A}$, where $\mathcal{I}_{W^*} = \{i_1, i_2, \cdots, i_w\}$ with $\omega \leq n$ represents the attribute index set specified in W^*. During the game, $\mathcal{A}$ will consult $\mathcal{S}$ for answers to the random oracles H_0, H_1 and $\hat{H}$. Roughly speaking, these answers are randomly generated, but to maintain the consistency and to avoid collisions, $\mathcal{S}$ keeps three tables $\mathcal{L}_1$, $\mathcal{L}_2$ and $\hat{\mathcal{L}}$ to store the answers used.

Setup. $\mathcal{S}$ needs to generate a system public key PK. $\mathcal{S}$ firstly chooses $j^* \in_R \{1, 2, \cdots, w\}$ and $x, x', y, y' \in_R \mathbb{Z}_p^*$. Then, it does the following:

1. If $i_j \in \mathcal{I}_{W^*} - \{i_{j^*}\}$, suppose $W_{i_j} = v_{i_j, k_{i_j}}$, then $\mathcal{S}$ computes

$$\left(X_{i_j,k_{i_j}}, Y_{i_j,k_{i_j}}\right) = \left(g^{-H_0(x||i_j||k_{i_j})} g_{m+1-i_j}^{-1}, \hat{e}(g,g)^{H_0(y||i_j||k_{i_j})}\right).$$

   Also, for $k \neq k_{i_j}$, $\mathcal{S}$ computes

$$\left(X_{i_j,k}, Y_{i_j,k}\right) = \left(g^{-H_0(x'||i_j||k)}, \hat{e}(g,g)^{H_0(y'||i_j||k)}\right).$$

2. For i_{j^*}, suppose $W_{i_{j^*}} = v_{i_{j^*}, k_{i_{j^*}}}$, then $\mathcal{S}$ computes

$$\left(X_{i_{j^*},k_{i_{j^*}}}, Y_{i_{j^*},k_{i_{j^*}}}\right)$$
$$= \left(g^{-H_0(x||i_{j^*}||k_{i_{j^*}})} \prod_{t \in \mathcal{I}_{W^*} - \{i_{j^*}\}} g_{m+1-t}, \hat{e}(g,g)^{H_0(y||i_{j^*}||k_{i_{j^*}})} \hat{e}(g,g)^{\alpha^{m+1}}\right).$$

Also, for $k \neq k_{i_{j^*}}$, $\mathcal{S}$ computes

$$\left(X_{i_{j^*},k}, Y_{i_{j^*},k}\right) = \left(g^{-H_0(x'||i_{j^*}||k)}, \hat{e}(g,g)^{H_0(y'||i_{j^*}||k)}\right).$$

3. If $i_j \notin \mathcal{I}_{W^*}$, for $1 \leq k_{i_j} \leq n_{i_j}$, $\mathcal{S}$ computes

$$\left(X_{i_j,k_{i_j}}, Y_{i_j,k_{i_j}}\right) = \left(g^{-H_0(x||i_j||k_{i_j})}, \hat{e}(g,g)^{H_0(y||i_j||k_{i_j})}\right).$$

Furthermore, $\mathcal{S}$ chooses $\varphi_1, \varphi_2, \varphi_3, \phi_2, \phi_3 \in_R \mathbb{Z}_p^*$, and computes $\delta_1 = g_1 g^{\varphi_1}$, $\delta_2 = g_1^{\phi_2} g^{\varphi_2}$, $\delta_3 = g_1^{\phi_3} g^{\varphi_3}$. Note that $\delta_1, \delta_2, \delta_3$ are distributed randomly. Also, $\mathcal{S}$ chooses $\beta \in \mathbb{Z}_p^*$, and sets $v = g^\beta \left(\prod_{j \in U^*} g_{m+1-j}\right)^{-1}$, where $U^* \subseteq \mathcal{R}_{W^*}$ denotes the target set of involved users to be challenged by $\mathcal{A}$ when revocation events occur. Finally, $\mathcal{S}$ sends $PK = \langle g, \{X_{i,k_i}, Y_{i,k_i}\}_{1 \leq i \leq n, 1 \leq k_i \leq n_i}, \{g_i\}_{1 \leq i \leq 2m, i \neq m+1}, v, \delta_1, \delta_2, \delta_3\rangle$ to $\mathcal{A}$.

Phase 1. The adversary $\mathcal{A}$ makes the following queries.

- **Hash Oracle** $\mathcal{O}_{H_0}$ and $\mathcal{O}_{\hat{H}}$ are answered in a trivial way.
- **Hash Oracle** $\mathcal{O}_{H_1}(sk)$: We consider there is not an item containing sk in $\mathcal{L}_1$. If sk corresponds to an attribute list L in the key generation oracle, $\mathcal{S}$ adds the entry $\langle sk, g_{i_j} g^z\rangle$ to $\mathcal{L}_1$ and returns $g_{i_j} g^z$, where $z \in_R \mathbb{Z}_p^*$ and i_j is associated with L and satisfies $L_{i_j} \notin W_{i_j}$. Otherwise, $\mathcal{S}$ randomly chooses $i_j \in_R \{1, 2, \cdots, n\}$, $z \in_R \mathbb{Z}_p^*$, adds the entry $\langle sk, g_{i_j} g^z\rangle$ to $\mathcal{L}_1$ and returns $g_{i_j} g^z$.
- **KeyGen Oracle** $\mathcal{O}_{KeyGen}(L)$: Suppose $\mathcal{A}$ summits an attribute list L in a secret key query. If $L \not\models W^*$, there must exist $i_j \in \mathcal{I}_{W^*}$ such that $L_{i_j} \notin W_{i_j}$. Without loss of generality, assume $L_{i_j} = v_{i_j,\widehat{k}_{i_j}}$ and $W_{i_j} = v_{i_j,k_{i_j}}$. $\mathcal{S}$ chooses $sk \in_R \mathbb{Z}_p^*$. Also, $\mathcal{S}$ computes $\overline{\sigma}_{i_j} = \sigma_{i_j,\widehat{k}_{i_j}} = g^{H_0(y'||i_j||\widehat{k}_{i_j})}(g_{i_j} g^z)^{H_0(x'||i_j||\widehat{k}_{i_j})}$. For $t \neq i_j$, $\mathcal{S}$ chooses $z \in_R \mathbb{Z}_p^*$ and computes $\overline{\sigma}_t$ as follows:

  Case 1. If $t \in \mathcal{I}_{W^*} - \{i_{j^*}\}$, suppose $L_t = v_{t,k_t}$, $\mathcal{S}$ computes

  $$\overline{\sigma}_t = \sigma_{t,k_t} = g^{H_0(y||t||k_t)}(g_{i_j})^{H_0(x||t||k_t)} g_{m+1-t+i_j} (\overline{X}_t)^{-z}.$$

  Case 2. If $t = i_{j^*}$, suppose $L_{i_{j^*}} = v_{i_{j^*},k_{i_{j^*}}}$, $\mathcal{S}$ computes $\overline{\sigma}_{i_{j^*}}$ as

  $$\begin{aligned}\overline{\sigma}_{i_{j^*}} &= \sigma_{i_{j^*},k_{i_{j^*}}} \\ &= g^{H_0(y||i_{j^*}||k_{i_{j^*}})}(g_{i_j})^{H_0(x||i_{j^*}||k_{i_{j^*}})} \left(\prod_{k \in \mathcal{I}_{W^*} - \{i_{j^*}, i_j\}} g^{-1}_{m+1-k+i_j}\right) (\overline{X}_{i_{j^*}})^{-z}.\end{aligned}$$

  Case 3. If $t \notin \mathcal{I}_{W^*}$, suppose $L_t = v_{t,k_t}$, $\mathcal{S}$ computes

  $$\overline{\sigma}_t = \sigma_{t,k_t} = g^{H_0(y||t||k_t)}(g_{i_j} g^z)^{H_0(x||t||k_t)}.$$

Subsequently, $\mathcal{S}$ computes $d = g_{sn}^{\beta} \prod_{j \in U^*} g_{m+1-j+sn}^{-1}$.

- **Decryption Oracle $\mathcal{O}_{Dec}(CT_W)$:** Suppose $\mathcal{A}$ summits $CT_W = \langle C_0, C_1, C_2, C_3, C_{\mathcal{R}}, \hat{s}\rangle$ where $W = \bigwedge_{i \in \mathcal{I}_W} W_i$ with $W_i = v_{i,k_i}$. $\mathcal{S}$ checks whether $\hat{e}(g, C_3) = \hat{e}(C_1, \delta_1^{\hat{h}} \delta_2^{\hat{s}} \delta_3)$ and $\hat{e}(g, C_2) = \hat{e}(C_1, X_W)$ or not, where $\hat{h} = \hat{H}(W|| C_0||C_1||C_2)$, $X_W = \prod_{i \in \mathcal{I}_W} X_{i,k_i}$. If one of the two equations does not hold, return $\perp$. Furthermore, $\mathcal{S}$ checks if $\hat{h} + \hat{s}\phi_2 + \phi_3 = 0$ holds. If so, $\mathcal{S}$ aborts. Otherwise, $\mathcal{S}$ chooses $sn \in_R \{1, 2, \cdots, m\}$, sets $\gamma = \hat{h} + \hat{s}\phi_2 + \phi_3$, $\hat{C} = C_1^{\hat{h}\varphi_1 + \hat{s}\varphi_2 + \varphi_3}$, and returns

$$\frac{C_0}{\hat{e}\left(\left(\frac{C_3}{\hat{C}}\right)^{\gamma^{-1}}, g_m\right) \cdot \hat{e}(C_1, g)^{y_W} \cdot K_{\mathcal{R}}},$$

where $y_W = \sum_{j=1}^{w} H_0(y||i_j||k_{i_j})$, and

$$K_{\mathcal{R}} = \frac{\hat{e}\left(g_{sn}, C_{\mathcal{R}}\right)}{\hat{e}\left(g_{sn}^{\beta} \cdot \prod_{i \in \mathcal{I}_m - \mathcal{R}}^{i \neq sn} g_{m+1-i+sn}, C_1\right)}.$$

Challenge. $\mathcal{S}$ sets $x_{W^*} = \sum_{j=1}^{\omega} H_0(x||i_j||k_{i_j})$ and $y_{W^*} = \sum_{j=1}^{w} H_0(y||i_j||k_{i_j})$, and defines $\langle X_{W^*}, Y_{W^*}\rangle$ as follows:

$$\begin{cases} X_{W^*} & = \overline{X}_{i_{j^*}} \prod_{t \in \mathcal{I}_{W^*} - \{i_{j^*}\}} \overline{X}_t \\ & = \left(g^{-H_0(x||i_{j^*}||k_{i_{j^*}})} \prod_{t \in \mathcal{I}_{W^*} - \{i_{j^*}\}} g_{m+1-t}\right) \cdot \prod_{t \in \mathcal{I}_{W^*} - \{i_{j^*}\}} g^{-H_0(x||t||k_t)} g_{m+1-t}^{-1} \\ & = g^{-x_{W^*}}, \\ Y_{W^*} & = \overline{Y}_{i_{j^*}} \prod_{t \in \mathcal{I}_{W^*} - \{i_{j^*}\}} \overline{Y}_t \\ & = \hat{e}(g,g)^{H_0(y||i_{j^*}||k_{i_{j^*}})} \hat{e}(g,g)^{\alpha^{m+1}} \cdot \prod_{t \in \mathcal{I}_{W^*} - \{i_{j^*}\}} \hat{e}(g,g)^{H_0(y||t||k_t)} \\ & = \hat{e}(g,g)^{\sum_{j=1}^{w} H_0(y||i_j||k_{i_j}) + \alpha^{m+1}}. \end{cases}$$

Suppose $\mathcal{A}$ summits two messages M_0 and M_1 of equal length. $\mathcal{S}$ chooses $b \in_R \{0, 1\}$, and computes $C_0^* = M_b Z^2 \hat{e}(g, h)^{y_{W^*}}$, $C_1^* = h$, and $C_2^* = h^{-x_{W^*}}$. Then $\mathcal{S}$ sets $\hat{h}^* = \hat{H}(W^*||C_0^*||C_1^*||C_2^*)$, $\hat{s}^* = \frac{-(\hat{h}^* + \phi_3)}{\phi_2}$, and computes $C_3^* = h^{\hat{h}^*\varphi_1 + \hat{s}^*\varphi_2 + \varphi_3}$ and

$$\begin{aligned} C_{\mathcal{R}^*} = h^{\beta} &= \left(g^{\beta}\left(\prod_{j \in U^*} g_{m+1-j}\right)^{-1}\left(\prod_{j \in U^*} g_{m+1-j}\right)\right)^{s} \\ &= \left(v \cdot \prod_{j \in U^*} g_{m+1-j}\right)^{s}. \end{aligned}$$

It is noted that $CT_{W^*} = \langle C_0^*, C_1^*, C_2^*, C_3^*, C_{\mathcal{R}^*}, \hat{s}^* \rangle$ is a valid encryption of M_b whenever $Z = \hat{e}(g_{m+1}, h)$. On the other hand, when Z is a random element in $\mathbb{G}_T$, CT_{W^*} is independent of b in the adversary's view.

Phase 2. Similar to **Phase 1** with a restriction that $\mathcal{A}$ cannot query $\mathcal{O}_{Dec}(\cdot)$ on the challenge ciphertext CT_{W^*}.

Guess. $\mathcal{A}$ outputs a guess bit b' of b. If $b' = b$, $\mathcal{S}$ outputs 1 in the decision m-BDHE game to guess that $Z = \hat{e}(g_{m+1}, h)$. Otherwise, it outputs 0 to indicate that Z is a random element in $\mathbb{G}_T$. We note that $\mathcal{S}$ will abort in decryption queries if $\hat{h} + \hat{s}\phi_2 + \phi_3 = 0$ holds. However, since the values ϕ_2 and ϕ_3 are respectively hidden by blinding factors φ_2 and φ_3, $\mathcal{A}$ could not obtain any information on ϕ_2 and ϕ_3 from decryption queries, and hence the probability that $\hat{h} + \hat{s}\phi_2 + \phi_3 = 0$ occurs is at most $\frac{1}{p}$. Therefore, if $Z = \hat{e}(g_{m+1}, h)$, then CT_{W^*} is a valid ciphertext and we have

$$\begin{aligned}\Pr\left[\mathcal{S}(g, h, \overrightarrow{y}_{g,\alpha,m}, \hat{e}(g_{m+1}, h)) = 1\right] &= \frac{1}{2} + \mathrm{Adv}_{\mathrm{CP-ABE}}^{\mathrm{IND-sCP-CCA2}}(\mathcal{A}) - \frac{q_D}{p} \\ &\geq \frac{1}{2} + \epsilon - \frac{q_D}{p}.\end{aligned}$$

If Z is a random element in $\mathbb{G}_T$, the message M_b is completely hidden from $\mathcal{A}$, and we have $\Pr[\mathcal{S}(g, h, \overrightarrow{y}_{g,\alpha,m}, Z) = 1] = \frac{1}{2}$.

Therefore, the simulator $\mathcal{S}$ has at least a non-negligible advantage $\epsilon - \frac{q_D}{p}$ in solving the decision m-BDHE problem in $\mathbb{G}$ within time τ. It easily follows that the time complexity of $\mathcal{S}$ is

$$\tau' = \tau + \mathcal{O}\left(q_{H_1} + mq_K + q_D + \sum_{i=1}^{n} n_i\right)\tau_1 + \mathcal{O}\left(q_D + \sum_{i=1}^{n} n_i\right)\tau_2 + \mathcal{O}(q_D)\tau_p.$$

□

7.3 Performance Analysis

In this section, we analyze and compare the performance of the proposed scheme with the previous CP-ABE schemes from the aspects of security and efficiency. Table 1 shows the performance comparison in terms of the size of ciphertext (CT) and the system public key (PK) size, the computation overheads of encryption and decryption, the expressiveness of access policy, and the revocation mechanism. For simplicity, we use **e** and **p** to represent an exponentiation operation and a pairing operation, respectively. Let n be the total number of attributes in universe, s be the number of attributes the user has to hold in order to match the access policy, t be the number of attributes associated with the user's secret key, s_m and t_m be the maximum size allowed for s and t, m be the maximum number of users in the system, r be the number of revocation events, and N be the total number of attribute values in the system. We denote the bit length of an element in a group $\mathbb{G}$ by $|\mathbb{G}|$. In addition, IAR and DUR respectively represent "Indirect Attribute Revocation" and "Direct User Revocation".

Schemes	Parameter Size		Computation Cost		Policy	Revocation
	CT	PK	Encryption	Decryption		
HSM [12]	$2\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(n+4)\|\mathbb{G}\|$	3 $\mathbf{e}$	2 $\mathbf{p}$	Type 1†	$\times$
EM [13]	$2\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(N+2)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	3 $\mathbf{e}$	2 $\mathbf{p}$	Type 2‡	$\times$
CZF1 [14]	$2\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$2n\|\mathbb{G}\| + 2n\|\mathbb{G}_T\|$	3 $\mathbf{e}$	2 $\mathbf{p}$	Type 3§	$\times$
CZF2 [14]	$3\|\mathbb{G}\| + \|\mathbb{G}_T\| + \|\mathbb{Z}_p^*\|$	$(2n+3)\|\mathbb{G}\| + 2n\|\mathbb{G}_T\|$	6 $\mathbf{e}$	6 $\mathbf{p}$+2 $\mathbf{e}$	Type 3	$\times$
YWR [29]	$(n+1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(3n+1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(s+2)\mathbf{e}$	$(n+1)\mathbf{p}$	Type 3	IAR
AI1 [31]	$(s+2)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(s_m + t_m + 2m + 1)\|\mathbb{G}\|$	$(2s+3)$ $\mathbf{e}$+1 $\mathbf{p}$	$(2s+m+1)$ $\mathbf{p}$	Type 4♯	DUR
AI2 [31]	$(s+2r+1)\|\mathbb{G}\| + \|\mathbb{G}_T\|$	$(s_m + t_m + 7)\|\mathbb{G}_T\|$	$(2s+2r+2)$ $\mathbf{e}$	$(2s+2r+1)$ $\mathbf{p}$	Type 4	DUR
Ours	$4\|\mathbb{G}\| + \|\mathbb{G}_T\| + \|\mathbb{Z}_p^*\|$	$(N+2m+4)\|\mathbb{G}\| + N\|\mathbb{G}_T\|$	8 $\mathbf{e}$	8 $\mathbf{p}$+2 $\mathbf{e}$	Type 5¶	DUR

† AND-gate policy supporting single positive value without wildcards.
‡ AND-gate policy supporting multiple values without wildcards.
§ AND-gate policy supporting positive and negative values with wildcards.
♯ Access structures based on linear secret sharing.
¶ AND-gate policy supporting multiple values with wildcards.

Table 1. Performance comparison of CP-ABE schemes

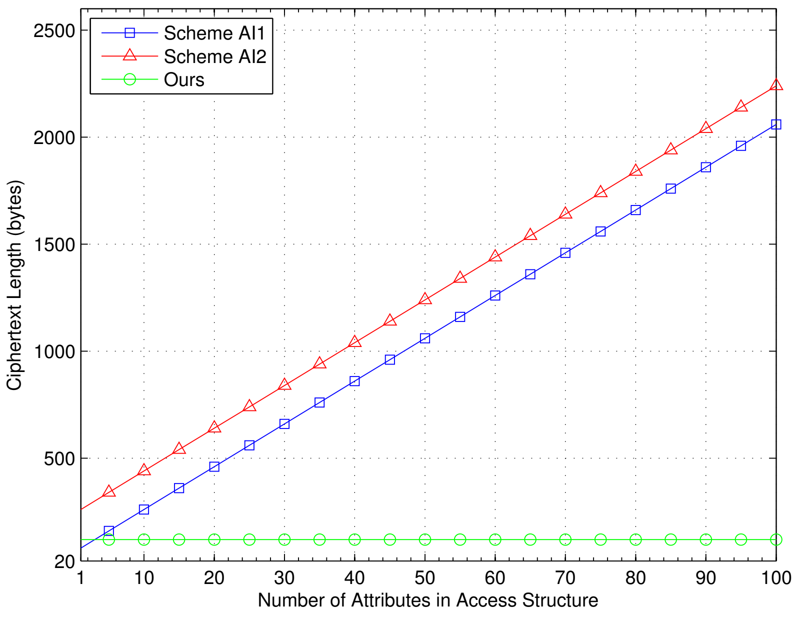


Figure 3. Comparison of ciphertext length

From Table 1, we know that the CP-ABE schemes [12, 13, 14] and the proposed scheme have small and constant computation cost. Although enjoying constant computation cost, the schemes [12, 13, 14] fail to support revocation mechanisms.

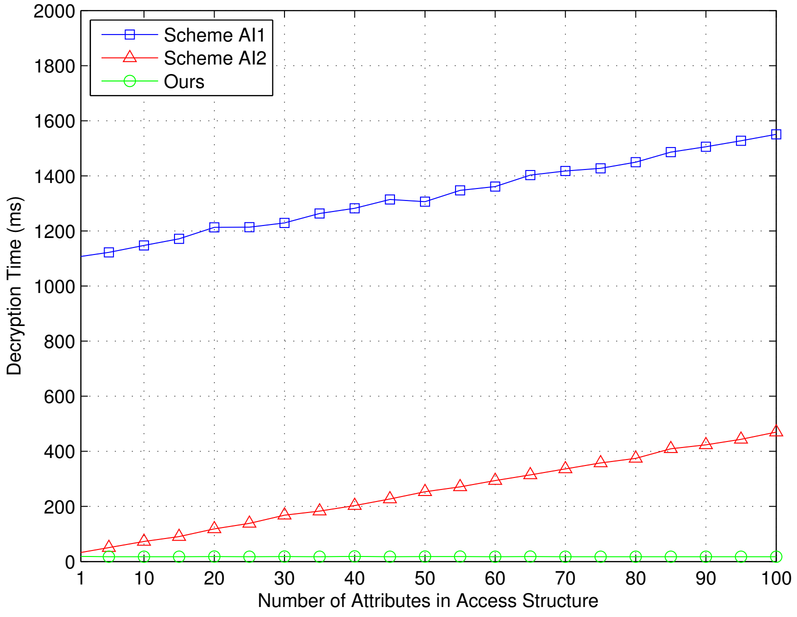


Figure 4. Comparison of decryption cost

Also, the access policies in [12] only support single attribute value. Furthermore, the scheme [29] supports indirect attribute revocation, and only the schemes [31] and the proposed scheme enjoy direct user revocation. However, the schemes [29, 31] suffer an efficiency drawback that the encryption and decryption cost is not constant in terms of the the number of **e** or **p**.

Based on the above analysis, we further compare schemes in [31] denoted as AI1, AI2 and ours with respect to the ciphertext length in Figure 3. As for the ciphertext length comparison, we set $|\mathbb{G}_0| = |\mathbb{G}_T| = 160$ bits and the number of revocation events as $r = 5$. Note that the ciphertext length of the scheme AI2 is linearly proportional to r. Both the ciphertext length of AI1 and AI2 linearly increases with s. On the other hand, we do simulation experiments based on the Stanford Pairing-Based Crypto (PBC) library [37] and a Linux machine with 3.30 GHz $\times$ 8 Intel Xeon(R) E3-1230 CPU and 7.5 GB of RAM. The simulation results are shown in Figure 4. In the simulation, the maximum number of users in the system is set as $m = 500$. In order to precisely evaluate the decryption cost, a total of 100 distinct access policies are generated, where each attribute has a positive occurrence. For each access policy, the experiment is repeated for 30 times and the final result is an average value. It is noted that both the decryption cost of the scheme AI1 and AI2 linearly increases with the number of columns in access policies, and the proposed scheme enjoys small and constant decryption cost. Generally, we argue that the proposed ABE scheme is more suitable for access control in cloud computing.

8 CONCLUSION

In this paper, we propose an efficient data access control system in cloud computing. The main building block is a new CP-ABE scheme, which enjoys constant computation cost and direct user revocation. The proposed access system is proven secure in the random oracle model, and it can efficiently support AND-policy with multiple attribute values and wildcards. Extensive performance comparisons indicate that the proposed solution is extremely suitable for resource-constrained applications.

Acknowledgements

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by National Key R & D Program of China (No. 2017YFB0802000), National Natural Science Foundation of China (No. 61772418, 61402366), Yinghui Zhang is supported by New Star Team of Xi'an University of Posts and Telecommunications.

REFERENCES

[1] Sahai, A.—Waters, B.: Fuzzy Identity-Based Encryption. In: Cramer, R. (Ed.): Advances in Cryptology – EUROCRYPT 2005. Springer, Lecture Notes in Computer Science, Vol. 3494, 2005, pp. 457–473, doi: 10.1007/11426639_27.

[2] Goyal, V.—Pandey, O.—Sahai, A.—Waters, B.: Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS '06), ACM, 2006, pp. 89–98, doi: 10.1145/1180405.1180418.

[3] Bethencourt, J.—Sahai, A.—Waters, B.: Ciphertext-Policy Attribute-Based Encryption. Proceedings of Symposium on Security and Privacy (SP '07), IEEE, 2007, pp. 321–334, doi: 10.1109/SP.2007.11.

[4] Cheung, L.—Newport, C.: Provably Secure Ciphertext Policy ABE. Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), ACM, 2007, pp. 456–465, doi: 10.1145/1315245.1315302.

[5] Zhang, X.—Tan, Y.—Liang, C.—Li, Y.—Li, J.: A Covert Channel over VoLTE via Adjusting Silence Periods. IEEE Access, Vol. 6, 2018, pp. 9292–9302, doi: 10.1109/ACCESS.2018.2802783.

[6] Zhang, Y.—Zheng, D.—Chen, X.—Li, J.—Li, H.: Efficient Attribute-Based Data Sharing in Mobile Clouds. Pervasive and Mobile Computing, Vol. 28, 2016, pp. 135–149, doi: 10.1016/j.pmcj.2015.06.009.

[7] Lin, Q.—Yan, H.—Huang, Z.—Chen, W.—Shen, J.—Tang, Y.: An ID-Based Linearly Homomorphic Signature Scheme and Its Application in Blockchain. IEEE Access, 2018, online, doi: 10.1109/ACCESS.2018.2809426.

[8] Xu, J.—Wei, L.—Zhang, Y.—Wang, A.—Zhou, F.—Gao, C.: Dynamic Fully Homomorphic Encryption-Based Merkle Tree for Lightweight Streaming Authenticated Data Structures. Journal of Network and Computer Applications, Vol. 107, 2018, pp. 113–124, doi: 10.1016/j.jnca.2018.01.014.

[9] Liu, Z.—Huang, Y.—Li, J.—Cheng, X.—Shen, C.: DivORAM: Towards a Practical Oblivious RAM with Variable Block Size. Information Sciences, Vol. 447, 2018, pp. 1–11, doi: 10.1016/j.ins.2018.02.071.

[10] Xie, D.—Lai, X.—Lei, X.—Fan, L.: Cognitive Multiuser Energy Harvesting Decode-and-Forward Relaying System with Direct Links. IEEE Access, Vol. 6, 2018, pp. 5596–5606, doi: 10.1109/ACCESS.2017.2776953.

[11] Lin, Q.—Li, J.—Huang, Z.—Chen, W.—Shen, J.: A Short Linearly Homomorphic Proxy Signature Scheme. IEEE Access, Vol. 6, 2018, pp. 12966–12972, online, doi: 10.1109/ACCESS.2018.2809684.

[12] Han, J.—Susilo, W.—Mu, Y.—Yan, J.: Attribute-Based Oblivious Access Control. The Computer Journal, Vol. 55, 2012, No. 10, pp. 1202–1215, doi: 10.1093/comjnl/bxs061.

[13] Emura, K.—Miyaji, A.—Nomura, A.—Omote, K.—Soshi, M.: A Ciphertext-Policy Attribute-Based Encryption Scheme with Constant Ciphertext Length. Proceedings of International Conference on Information Security Practice and Experience (ISPEC '09). Springer, Lecture Notes in Computer Science, Vol. 5451, 2009, pp. 13–23, doi: 10.1007/978-3-642-00843-6_2.

[14] Chen, C.—Zhang, Z.—Feng, D.: Efficient Ciphertext Policy Attribute-Based Encryption with Constant-Size Ciphertext and Constant Computation-Cost. Proceedings of International Conference on Provable Security (ProvSec 2011). Springer, Lecture Notes in Computer Science, Vol. 6980, 2011, pp. 84–101, doi: 10.1007/978-3-642-24316-5_8.

[15] Zhang, Y.—Zheng, D.—Chen, X.—Li, J.—Li, H.: Computationally Efficient Ciphertext-Policy Attribute-Based Encryption with Constant-Size Ciphertexts. Proceedings of International Conference on Provable Security (ProvSec 2014). Springer, Lecture Notes in Computer Science, Vol. 8782, 2011, pp. 259–273, doi: 10.1007/978-3-319-12475-9_18.

[16] Li, J.—Chen, X.—Li, M.—Li, J.—Lee, P.—Lou, W.: Secure Deduplication with Efficient and Reliable Convergent Key Management. IEEE Transactions on Parallel and Distributed Systems, Vol. 25, 2014, No. 6, pp. 1615-1625, doi: 10.1109/TPDS.2013.284.

[17] Chase, M.: Multi-Authority Attribute Based Encryption. Proceedings of Theory of Cryptography Conference (TCC'07). Springer, Lecture Notes in Computer Science, Vol. 4392, 2007, pp. 515–534, doi: 10.1007/978-3-540-70936-7_28.

[18] Lewko, A.—Waters, B.: Decentralizing Attribute-Based Encryption. Advances in Cryptology – EUROCRYPT 2011. Springer, Lecture Notes in Computer Science, Vol. 6632, 2011, pp. 568–588, doi: 10.1007/978-3-642-20465-4_31.

[19] Green, M.—Hohenberger, S.—Waters, B.: Outsourcing the Decryption of ABE Ciphertexts. Proceedings of the 20th USENIX Conference on Security (SEC '11), USENIX Association, 2011, pp. 1–16, `http://static.usenix.org/events/sec11/tech/full_papers/Green.pdf`.

[20] Li, J.—Huang, X.—Li, J.—Chen, X.—Xiang, Y.: Securely Outsourcing Attribute-Based Encryption with Checkability. IEEE Transactions on Parallel and Distributed Systems, Vol. 25, 2014, No. 8, pp. 2201–2210, doi: 10.1109/TPDS.2013.271.

[21] Li, J.—Chen, X.—Li, J.—Jia, C.—Ma, J.—Lou, W.: Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption. Proceedings of European Symposium on Research in Computer Security (ESORICS '13). Springer, Lecture Notes in Computer Science, Vol. 8134, 2013, pp. 592–609, doi: 10.1007/978-3-642-40203-6_33.

[22] Katz, J.—Sahai, A.—Waters, B.: Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. Advances in Cryptology – EUROCRYPT 2008. Springer, Lecture Notes in Computer Science, Vol. 4965, 2008, pp. 146–162, doi: 10.1007/978-3-540-78967-3_9.

[23] Nishide, T.—Yoneyama, K.—Ohta, K.: Attribute-Based Encryption with Partially Hidden Encryptor-Specified Access Structure. Proceedings of International Conference on Applied Cryptography and Network Security (ACNS 2008). Springer, Lecture Notes in Computer Science, Vol. 5037, 2008, pp. 111–129, doi: 10.1007/978-3-540-68914-0_7.

[24] Zhang, Y.—Chen, X.—Li, J.—Wong, D. S.—Li, H.—You, I.: Ensuring Attribute Privacy Protection and Fast Decryption for Outsourced Data Security in Mobile Cloud Computing. Information Sciences, Vol. 379, 2017, pp. 42–61, doi: 10.1016/j.ins.2016.04.015.

[25] Zhang, Y.—Chen, X.—Li, J.—Wong, D. S.—Li, H.: Anonymous Attribute-Based Encryption Supporting Efficient Decryption Test. Proceedings of 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS '13), ACM, 2013, pp. 511–516, doi: 10.1145/2484313.2484381.

[26] Zhang, Y.—Li, J.—Zheng, D.—Chen, X.—Li, H.: Towards Privacy Protection and Malicious Behavior Traceability in Smart Health. Personal and Ubiquitous Computing, Vol. 21, 2017, No. 5, pp. 815–830, doi: 10.1007/s00779-017-1047-8.

[27] Xhafa, F.—Feng, J.—Zhang, Y.—Chen, X.—Li, J.: Privacy-Aware Attribute-Based PHR Sharing with User Accountability in Cloud Computing. The Journal of Supercomputing, Vol. 71, 2015, No. 5, pp. 1607–1619, doi: 10.1007/s11227-014-1253-3.

[28] Zhang, Y.—Li, J.—Zheng, D.—Chen, X.—Li, H.: Accountable Large-Universe Attribute-Based Encryption Supporting Any Monotone Access Structures. Proceedings of Australasian Conference on Information Security and Privacy (ACISP 2016). Springer, Lecture Notes in Computer Science, Vol. 9722, 2016, pp. 509–524, doi: 10.1007/978-3-319-40253-6_31.

[29] Yu, S.—Wang, C.—Ren, K.—Lou, W.: Attribute Based Data Sharing with Attribute Revocation. Proceedings of the 5th ACM Symposium on Information, Com-

puter and Communications Security (ASIA CCS '10), ACM, 2010, pp. 261–270, doi: 10.1145/1755688.1755720.

[30] Yang, K.—Jia, X.—Ren, K.: Attribute-Based Fine-Grained Access Control with Efficient Revocation in Cloud Storage Systems. Proceedings of 8th ACM SIGSAC Symposium on Information, Computer and Communications Security (ASIA CCS '13), ACM, 2013, pp. 523–528, doi: 10.1145/2484313.2484383.

[31] Attrapadung, N.—Imai, H.: Conjunctive Broadcast and Attribute-Based Encryption. Proceedings of International Conference on Pairing-Based Cryptography (Pairing 2009). Springer, Lecture Notes in Computer Science, Vol. 5671, 2009, pp. 248–265, doi: 10.1007/978-3-642-03298-1_16.

[32] Fiat, A.—Naor, M.: Broadcast Encryption. Advances in Cryptology – CRYPTO 1993. Springer, Lecture Notes in Computer Science, Vol. 773, 1994, pp. 480–491, doi: 10.1007/3-540-48329-2_40.

[33] Boneh, D.—Gentry, C.—Waters, B.: Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. Advances in Cryptology – CRYPTO 2005. Springer, Lecture Notes in Computer Science, Vol. 3621, 2005, pp. 258–275, doi: 10.1007/11535218_16.

[34] Zhang, Y.—Chen, X.—Li, J.—Li, H.—Li, F.: FDR-ABE: Attribute-Based Encryption with Flexible and Direct Revocation. Proceedings of 2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS '13), IEEE, 2013, pp. 38–45, doi: 10.1109/INCoS.2013.16.

[35] Li, J.—Zhang, Y.—Chen, X.—Xiang, Y.: Secure Attribute-Based Data Sharing for Resource-Limited Users in Cloud Computing. Computers and Security, Vol. 72, 2018, pp. 1–12, doi: 10.1016/j.cose.2017.08.007.

[36] Zhang, Y.—Li, J.—Chen, X.—Li, H.: Anonymous Attribute-Based Proxy Re-Encryption for Access Control in Cloud Computing. Security and Communication Networks, Vol. 9, 2016, No. 14, pp. 2397—2411, doi: 10.1002/sec.1509.

[37] Lynn, B.: The Stanford Pairing Based Crypto Library. `https://crypto.stanford.edu/pbc/`.

Yinghui Zhang received his Ph.D. degree in cryptography from the Xidian University, China, in 2013. He is Associate Professor at the National Engineering Laboratory for Wireless Security (NELWS), Xi'an University of Posts and Telecommunications. He has published over 50 research articles including ASIACCS, ACISP, IEEE CSE, computer networks, computers & security. His research interests include cloud security, public key cryptography and wireless network security.

Dong Zheng received his Ph.D. degree in communication engineering from the Xidian University, China, in 1999. He is currently Professor at the National Engineering Laboratory for Wireless Security (NELWS), Xi'an University of Posts and Telecommunications. He has published over 100 research articles including CT-RSA, IEEE Transactions on Industrial Electronics, etc. His research interests include cloud computing and public key cryptography.

Rui Guo received his Ph.D. degree from the Department of State Key Laboratory of Networking and Switch Technology, Beijing University of Posts and Telecommunications, China, in 2014. He is currently Lecturer at the National Engineering Laboratory for Wireless Security (NELWS), Xi'an University of Posts and Telecommunications. His present research interests include attribute-based cryptography, cloud computing and blockchain technology.

Qinglan Zhao received her B.Sc. degree from the Shaanxi Normal University, China, in 1999, and her M.Sc. degree from the Northwestern Polytechnical University, China, in 2006. She received her Ph.D. degree from the Shanghai Jiao Tong University, China, in 2017. Since 2014, she has been Associate Professor at the Xi'an University of Post and Telecommunications, China. Her research interests include cryptographic functions and information security.

Computing and Informatics, Vol. 37, 2018, 349–366, doi: 10.4149/cai_2018_2_349

USING LOCAL REDUCTION FOR THE EXPERIMENTAL EVALUATION OF THE CIPHER SECURITY

Pavol Zajac

Institute of Computer Science and Mathematics
Slovak University of Technology
Ilkovičova 3
812 19 Bratislava, Slovakia
e-mail: `pavol.zajac@stuba.sk`

Abstract. Evaluating the strength of block ciphers against algebraic attacks can be difficult. The attack methods often use different metrics, and experiments do not scale well in practice. We propose a methodology that splits the algebraic attack into a polynomial part (local reduction), and an exponential part (guessing), respectively. The evaluator uses instances with known solutions to estimate the complexity of the attacks, and the response to changing parameters of the problem (e.g. the number of rounds). Although the methodology does not provide a positive answer ("the cipher is secure"), it can be used to construct a negative test (reject weak ciphers), or as a tool of qualitative comparison of cipher designs. Potential applications in other areas of computer science are discussed in the concluding parts of the article.

Keywords: Algebraic cryptanalysis, local reduction, method of syllogisms, SAT solvers

Mathematics Subject Classification 2010: 94A60, 14G50

1 INTRODUCTION

The algebraic cryptanalysis is based on the idea of transforming cryptanalytic problem to a problem of solving a carefully crafted set of equations (with algebraic methods). Algebraic attacks can be executed even if the attacker has only a very small number of P-C pairs. On the other hand, it seems quite difficult to scale the attacks (using SAT solvers, or Gröbner basis solver), or to correctly compare results in assessing the strength of cryptographic primitives.

Nowadays, two main approaches are prevalent in the algebraic cryptanalysis:

1. Transform the cryptanalytic problem to CNF form, and apply fast SAT solvers.
2. Find as many linearly independent MQ equations as possible, and apply relinearization to find solutions [5], or compute solutions by computing the Gröbner basis of the ideal generated by these equations [9, 10].

These methods usually terminate in reasonable time only if some number of key bits is fixed in advance. Moreover, Gröbner basis approach can often fail due to memory constraints.

A different approach to algebraic cryptanalysis was proposed by Raddum and Semaev [15]. An encryption process can be described by a sequence of smaller operations working on some internal state. At the bottom level, the cipher can be described by a logic circuit, with logic gates, and wires. We can assign a variable for each internal value transmitted on the wires during the encryption process. Then each logic gate defines a Boolean equation in its input and output variables. It is easy to enumerate all possible (partial) solutions for Boolean equations with small number of variables (sparse equations). If we descend to a reasonably low level in cipher description, each of these equations is 3-sparse, i.e., it has at most three variables (two inputs, and one output). It might be more practical to work on higher level, e.g. on the full S-box level [13]. The equations taken together with a known P-C pair provide an equation system for the algebraic cryptanalysis. We primarily want to find the value of key bits, but we can consider all intermediate values as unknowns in the system.

We represent equation system as algebraic varieties, resp. as a list of points of these varieties projected to chosen coordinates given by variables the equation depends on. Our goal is to find the intersection of these varieties. If the point P on variety does not belong to the intersection of varieties, we can replace the original variety by a reduced one that does not contain P without influencing the final solution of the system. In some cases, we can determine the condition "does not belong to the intersection" in polynomial time, e.g., if it does not belong to the intersection of 2 varieties, it clearly cannot belong to intersection of all varieties. We call this process the (polynomial time) local reduction of the equation system [24]. If we can simplify each equation to a single solution, the system is trivially solved (it suffices to read values of variables from individual equations). In other cases we can compute the intersection by the (exponentially) difficult Gluing algorithm [14].

Let us suppose that we have a polynomial time local reduction algorithm A, but we are unable to remove any more partial solutions in the system. We suppose that the system is still unsolved, i.e., there are (unfixed) variables that can have both 0 and 1 assigned in each equation they are influencing. Let A be well behaved in a sense that it has a higher chance of reducing system, if individual equations have a lower number of partial solutions. If we take an unfixed variable and guess its value, we can remove at least one partial solution in each equation influenced by this variable. This can restart the local reduction. After guessing a finite number of variable values, we either end with the solved system, or we remove all partial solutions from some equation(s). The latter case means that the system does not have a solution compatible with the guesses (a conflict is detected), and we must backtrack our guessing sequence. The guessing process complexity is exponential in the number of variables that have to be guessed before the system can be reduced to a solution/conflict, and thus it dominates the polynomial complexity of A when we scale the problem. We silently suppose that behavior of A is not negatively influenced by expanding the system. Although the conditions on the behavior of A, and the computation of the influence of the system size on the required number of guesses can be impractical for exact determination of the complexities, it can be useful in practical experimental assessment of the strength of iterated block ciphers, and other cryptographic designs that can be scaled in a similar way.

The structure of the article is as follows: In Section 2 we provide preliminary definitions. In Section 3 we describe generic local reduction scheme, and its use in solving sparse Boolean equation system. In Section 4 we provide concrete algorithms that fit into the local reduction scheme. Finally, in Section 5 we focus on the use of the local reduction in experimental evaluation of the strength of the iterated ciphers against algebraic cryptanalysis. As an illustration, in Section 5.1 we provide experimental results from the analysis of the block cipher DES by the method of syllogisms.

2 PRELIMINARIES

Let $F : GF(2)^n \to GF(2)^m : F(x) = (f_1(x), \ldots, f_m(x))$, be a Boolean function with m component functions f_i. Equation $F(x) = 0$ defines a system of m Boolean equations $f_i(x) = 0$, for $i = 1, \ldots, m$ in n variables $x_1, x_2, \ldots, x_n$.

Let S be the set of solutions of a system of Boolean equations $F(x) = 0$, i.e. $S = \{x \in GF(2)^n; F(x) = 0\}$. We say that the system is inconsistent, iff $S = \emptyset$. We usually suppose, that F is defined in such a way, that there is exactly one solution to the system, i.e. $|S| = 1$ (but this is not a necessary condition).

Let S_i be the set of solutions of i^{th} equation of the system, i.e. $S_i = \{x \in GF(2)^n; f_i(x) = 0\}$. If $x \in S$, then it must also be in each S_i (the converse is not true in general), thus $S \subset S_i$, and $S = \bigcap_{i=1}^{m} S_i$.

Definition 1. Let $f : GF(2)^n \to GF(2)$ be a Boolean function. Let $e^{(i)} \in GF(2)^n$,

$$\mathrm{proj}_j(e^{(i)}) = \begin{cases} 1, & j = i, \\ 0, & j \neq i. \end{cases}$$

We say that f depends on variable x_i iff there exists x such that $f(x) \oplus f(x \oplus e^{(i)}) = 1$.

Let X_i be the set of coordinates on which f_i depends. We say, that Boolean equation system $F(x) = 0$ is l-sparse, iff $|X_i| \leq l$, for each $i = 1, \ldots, m$.

If f_i depends only on a small set of variables, we can represent S_i more effectively by storing only the values of variables on which f_i depends – vectors of length $|X_i|$, indexed by variables in X_i in the chosen order. We call these vectors partial solutions. We will denote a set of such vectors V_i. Other variables can take all possible values for each $v \in V_i$.[1] We will call (X_i, V_i) a symbol representation of the equation $f_i(x) = 0$, and the set $\mathcal{V} = \{(X_i, V_i); i = 1, \ldots, m\}$ a symbol representation of the system.

Let $F(x) = 0$ define an l-sparse $m \times n$ Boolean equation system (as defined above). We can produce a symbol representation of $f_i(x) = 0$ in at most 2^l evaluations of the Boolean function f_i. Thus it is possible to compute $\mathcal{V}$ in at most $m2^l$ evaluations of simple Boolean functions. In some applications it is possible to compute $\mathcal{V}$ even faster. E.g. in algebraic cryptanalysis, we can describe individual S-boxes with equations $y = S(x)$ by symbols (X_i, V_i), where $X_i = \{x_1, \ldots, x_k, y_1, \ldots, y_j\}$, and V_i contains 2^k vectors in the form $(x, S(x))$.

The problem of solving Boolean equation system in symbol representation is as follows: Given $\mathcal{V}$, compute S. We can alternatively have two simpler goals: compute at least one element of S, or show that $S = \emptyset$.

3 SOLVING SPARSE BOOLEAN SYSTEMS BY LOCAL REDUCTION

Let $\mathcal{V} = \{(X_i, V_i)\}$ denote a system of Boolean equations in symbol representation, and let S denote a set of all solutions of the equation system. Let $s \in S$ be a solution, and let s_i be its projection to X_i. Clearly $s_i \in V_i$ for $i = 1, 2, \ldots, m$. We call such vectors true (partial) solutions.

Let us suppose that there exists $v \in V_i$, which is not a projection of any of $s \in S$ – a false (partial) solution. A false solution is in a conflict with the rest of the equation system. It is possible to replace the symbol (X_i, V_i) by the symbol $(X_i, V_i \setminus \{v\})$. The new symbol represents a different Boolean equation $f_i'(x) = 0$ (we factor out the part of f_i corresponding to the root v), but the new system of equations has the same solution set S. We say that we have locally reduced the system. We may remove some partial solutions in such a way that the new equation does not depend on some $x \in X_i$. In this case, we should also remove x from X_i (and corresponding coordinates from V_i).

The primary goal of a local reduction method is to remove all false partial solutions from sets V_i. The reduced system allows us to easily identify a conflict, or a solution (of the original system). Let us suppose that $|S| > 0$, and $\mathcal{V}$ contains a symbol with $|V_i| = 1$. Then exact values of variables from X_i are known to us without the need to further examine the system. Similarly, if we get some $V_i = \emptyset$, then clearly $S = \emptyset$. We say that the equation is solved, if its symbol representation

[1] V_i is a projection of variety $V(f_i)$ into coordinates given by X_i.

does not contain any false partial solutions. The system is solved, if every equation in the system is solved.

Let $\mathcal{P}$ denote a finite set of predicates (in practice, we can use any suitable form of representation). Let us define three basic operations:

1. Apply:

Input: $\mathcal{P}$, (X_i, V_i)
Output: (X_i', V_i')
Description: Let $V_i' = \emptyset$. For each vector $v \in V_i$, check whether it is in conflict with $\mathcal{P}$. If not, add it to V_i'. Finally, construct symbol (X_i', V_i') by removing variables on which the equation no longer depends (if any).

2. Collect:

Input: (X_i, V_i)
Output: Set of predicates P
Description: Computes a set of predicates P that represent (local) information about the symbol (X_i, V_i).

3. Join:

Input: Sets of predicates P, $\mathcal{P}$
Output: Updated $\mathcal{P}$
Description: Merges (local) information P into (global) $\mathcal{P}$.

Generic local reduction scheme works as follows (with input $\mathcal{V}$, $\mathcal{P}$):

1. Let $i = 1$, $j = 0$.
2. Let $(X_i', V_i') = \mathit{Apply}(\mathcal{P}, (X_i, V_i))$. If $V_i' = \emptyset$, STOP, return CONFLICT.
3. Let $P = \mathit{Collect}(X_i', V_i')$.
4. $\mathcal{P}' = \mathit{Join}(P, \mathcal{P})$. If $\mathcal{P}$ is inconsistent, STOP, return CONFLICT.
5. If $(X_i, V_i) = (X_i', V_i')$ and $\mathcal{P}' = \mathcal{P}$, increment j. Else $j = 0$.
6. If $j = m$, STOP, return REDUCED.
7. $\mathcal{V} = (\mathcal{V} \setminus (X_i, V_i)) \cup (X_i', V_i')$. Cyclically increment i, and GOTO step 2.

This description is general, and not necessarily optimal. If no information about the system is known a priori, initial $\mathcal{P} = \emptyset$. Scheme returns the state (CONFLICT/REDUCED), and the final $\mathcal{V}, \mathcal{P}$, respectively.

It is easy to show that the algorithm stops. In each step either we learn something new about the system, remove a false partial solution, or increment the counter j. We suppose that possible $\mathcal{P}$ is finite. Then there exists a saturation point, i.e., we cannot learn anything more about the system. There is a limited number of partial solutions, thus we must stop removing solutions at some point.

Counter j reaches m, if we were unable to add new information, or remove any partial solution from each of the equations.

Let $m = O(n)$, and $\mathcal{V}$ be l sparse. Let $|\mathcal{P}|$, as well as running times of operations *Apply*, *Collect*, and *Join* be polynomially bounded in n. Then the running time of the *Local Reduction* algorithm is also polynomially bounded in n. This is easy to see: Each repetition of the algorithm uses 1 application of *Apply*, *Collect*, and *Join*, so we get $O(n^{k_1})$ complexity of each repetition, where k_1 is determined by the most difficult of the three operations. The number of repetitions is lower than $|\mathcal{P}| + m2^l + m = O(n^{k_2})$. The total running time is thus bounded by $O(n^{k_1+k_2})$. The similar holds for the memory requirements. In the remainder of this paper we will only consider polynomial *Local Reduction* algorithms.

If the algorithm stops with conflict, we know that the system is inconsistent, and thus it has no solution. Otherwise, the algorithm stops when nothing more can be learned about the system by the selected local reduction method. The algorithm then outputs the reduced system (and optionally also the information about the system $\mathcal{P}$). In some cases, the system is solved, and thus we can find at least one solution $s \in S$ in polynomial time (the trivially detected case is when each V_i contains a single solution). Otherwise we can find the solution using guessing and backtracking:

1. Let $G = \{p_1, p_2, \ldots, p_k\}$ be a set of (mutually exclusive) statements about the solution of the system $\mathcal{V}$ not already contained in $\mathcal{P}$. Furthermore, let G have a property, that if each of p_i is false, then $\mathcal{V}$ has no solution.
2. If $G = \emptyset$, return CONFLICT.
3. Guess: Choose $p \in G$.
4. Reduce: Let $(State, \mathcal{V}', \mathcal{P}') = LocalReduction(\mathcal{V}, \mathcal{P} \cup \{p\})$.
5. If *State* is CONFLICT, remove p from G, GOTO Step 2.
6. Unless solution is found, recursively apply this procedure on $(\mathcal{V}', \mathcal{P}')$. If CONFLICT is returned, remove p from G, GOTO Step 2.

If there was a polynomial instantiation of *Local Reduction* scheme which produces directly (without guessing) a solution for each $\mathcal{V}$, it would mean P = NP. However, we are skeptical that this is the case. We are more interested in determining the bounds for polynomially solvable cases, and the resulting global complexity of the guessing for systems, which are not polynomially solvable.

Due to the recursive nature of the guessing algorithm, its expected complexity is exponential (it depends on the depth and branching of the search tree). We call the sequence of sets G in the recursion the guessing strategy. The guessing strategy significantly influences the final complexity of the attack [21]. We believe that for each polynomial local reduction method (with fixed sparsity), there can be some optimal generic strategy, which leads to the lowest possible complexity. However, we cannot prove this hypothesis.

A typical set of statements G is a choice of a value of a single variable, e.g. $G = \{x_1 = 0; x_1 = 1\}$. If we use this method for algebraic cryptanalysis, the number of values we need to guess (depth of recursion) corresponds to a bit complexity of the attack.

Another type of G can be constructed from a single symbol as a choice of a partial solution, e.g. $G = \{v_1 \text{ is true solution}; v_2 \text{ is true solution}, \ldots\}$. This is especially useful, if we know a priori that there is exactly one solution of the system. If the symbol with $|X_i| = l$ variables contains $|V_i| = k_i$ partial solutions, we can examine all possible values of l variables with only k_i choices. The bit complexity corresponding to a single symbol is given by $\log_2 k_i$, and the bit complexity of the whole attack is given by $\sum \log_2 k_i$, with the sum taken over symbols we use in the attack.

If we want to assess the strength of the cipher against the attack based on local reduction, we can generate the problem instance with a known solution. In each step, when we have to guess some value, we can provide the correct answer to the algorithm, so no backtracking is necessary. We know that finally the process will converge to a (known) solution. Meanwhile, we keep track of the expected bit complexity of the (uninformed) attack. To estimate the complexity of the attack on a random unknown instance, we can randomize the process by trying different random instances (e.g., for the block cipher testing we can use different plaintexts, and keys, respectively), and we can also randomize the guessing strategy. Under the condition that the local reduction method is polynomial, we can expect the evaluation of the complexity can be relatively fast even for large systems.

4 LOCAL REDUCTION ALGORITHMS

We present a collection of polynomial local reduction algorithms that can be used in conjunction with guessing to solve equation systems in symbol forms. The list is not exhaustive, moreover the individual methods can be combined to create a more efficient version of the algorithm. The method "Spreading of constants" is the common part of each of the presented methods, otherwise these methods are distinct (i.e., there are examples of systems that can be solved by one of the methods but not the others). In our algebraic cryptanalysis experiments we use the method of syllogisms, for which we have implemented the experimental software solver [20].

4.1 Local Reduction with Linearization

One of the basic local reduction methods is based on the linearization of the symbols. We try to find linear equations that hold for each partial solution in each symbol individually, and remove incorrect solutions using global linear algebra.

Let $\mathcal{P}$ be represented as a set of linearly independent equations in all n variables. Maximum cardinality of $\mathcal{P}$ is n. We can instantiate the Local reduction procedures as follows:

1. **Apply:** For each partial solution $v \in V_i$ substitute values of variables from X_i within equations in $\mathcal{P}$, and check whether the resulting system of linear equations has at least one solution. If not, v can be removed.

2. Collect: Find the largest possible set P of independent linear equations in variables from X_i, such that each $v \in V_i$ is in a solution space of P.

3. Join: Add equations from P to $\mathcal{P}$, and remove all linearly dependent equations (e.g. by triangularization).

Theorem 1. For each symbol (X_i, V_i), with $0 < |V_i| \leq |X_i|$ there exists at least one affine Boolean function a, such that each $v \in V_i$ is a solution of $a(v) = 0$.

Proof. Let (X_i, V_i) be a symbol with $|X_i| = l$ variables, and $|V_i| = k$ partial solutions, respectively. Let A be $l \times k$ matrix with columns corresponding to the partial solutions. Let I be $l \times l$ identity matrix, with columns corresponding to variables from X_i. Let $M = (B|C)$ be a reduced row echelon form of matrix $(A|I)$. Each row with all zeros in the first k columns, represents a linear equation that holds for each solution (if the row is all zero, it is the trivial equation $0 = 0$, otherwise the coefficients are given by the part of the row in C). If $B = I$, we can construct affine equation by summing all rows of C (the right-hand side becomes one in each solution). $\square$

Sometimes it is possible to find linear equations also when $|V_i| > |X_i|$, but these cases are rare. However, if we guess value of some variable in the symbol, we expect that $|V_i|$ is halved (in average). Thus with each guess we are getting higher chance of finding linear equations, until the system can completely be linearized.

4.2 Spreading of Constants

The spreading of constants can be considered a special case of linearization, when we limit $\mathcal{P}$ to contain only equations in the form $x_i = a_i$. Finding the equations in the individual symbols is very easy. We remark that spreading of constants is also a special case of the method of syllogisms, and Agreeing, respectively.

4.3 Method of Syllogisms

The method of syllogisms was proposed by [24], and we have further been able to adapt it for algebraic cryptanalytic purposes [19, 21]. We provide a simple scheme as follows.

Let $\mathcal{P}$ contain logic formulas (implications) in the form $(x_j = a) \Rightarrow (x_k = b)$.[2] We have $|\mathcal{P}| = (2n)^2$.

1. Apply: For each partial solution $v \in V_i$ check whether all implications in $\mathcal{P}$ are true. If not, remove v.

2. Collect: For each pair of variables $x_j, x_k \in X_i$ make projection of V_i onto $\{x_j, x_k\}$. Each missing tuple (from the set $\{00, 01, 10, 11\}$) defines two new implications of the required form.

[2] In practice we store these formulas in the form of the implication graph.

3. Join: Add implications from P to $\mathcal{P}$, and compute the transitive closure of the corresponding implication graph.

The complexity of operations Collect and Apply is dominated by the number of partial solutions and variables in the symbol. If we consider system to be l-sparse with fixed l, this complexity factors becomes "constant" as n grows. The transitive closure of the implication graph can be computed in $O(n^3)$ (e.g. by Warshall's algorithm). Thus the whole Local reduction based on the method of syllogisms is polynomial as required.

Let $\mathcal{V}$ be an l-sparse equation system with randomly chosen X_i's. Let each l-tuple become a partial solutions in V_i with probability p. In [18] we show that if p is low enough, then the system can be solved by the method of syllogisms without any guessing with very high probability. The actual threshold depends on l (for smaller l's it is higher) but does not seem to depend on n. This means that if the average number of solutions per symbol in the system falls below some threshold, the system can be solved with no (more) guessing. If the system cannot be reduced by the method of syllogisms, with guessing we can remove some partial solutions (incompatible with the guess), thus lowering the average number of partial solutions per symbol. We can thus expect that the guessing process will terminate with the collapse of the system by the method of the syllogisms.

4.4 Agreeing

We can also map the method Agreeing from [14] into the scope of Local reduction methods. The set $\mathcal{P}$ is represented directly by $\mathcal{V}$ (can be augmented by more effective representations).

1. Apply: For each partial solution $v \in V_i$ check whether it is possible to find a matching projection on $X_i \cap X_j$ in V_j, with $X_i \cap X_j \neq \emptyset$. If the projection is missing in some of the connected equations, remove v.

2. Collect: Nothing to be done.

3. Join: Nothing to be done.

In [18] we show that Agreeing has similar behavior as the method of syllogisms when working with random equation systems, however it tends to be more effective when the sparsity l is larger (the threshold is at $l = 7$). We should note that there are now more efficient realizations of the Agreeing algorithm [17]. However, this new algorithm already combines basic Agreeing with guessing, and learning new information during the guessing process, so it is not possible anymore to directly incorporate it into our local reduction scheme.

5 EXPERIMENTAL EVALUATION OF THE SECURITY OF ITERATED CIPHER DESIGNS

The local reduction framework splits the question of the complexity of the algebraic cryptanalysis into two parts: polynomial local reduction algorithm, and exponential guessing. The practical realization of the local reduction algorithm can be quite complex, especially in comparison with the speed of the encryption routine. However, this complexity factor can be omitted when we are scaling the problem (exponential guessing part dominates the polynomial one). The strength of the cipher is then given by the minimum number of intermediate bits the attacker needs to guess before the local reduction method can solve the problem.

Most block ciphers are designed in such a way that it is relatively straightforward to construct an l-sparse Boolean equation system in symbol representation describing the encryption process. In the basic form, all input, output and key bits can also be represented by corresponding unknowns in the system. When analyzing the cipher we fix the input and output bits according to a known P-C pair.

The goal is to solve the equation system in the remaining unknowns with the minimum complexity. When we want to evaluate the complexity of the attack, we can work with instances for which we know the whole solution. We can thus provide the correct solution for each guess incrementally until the system collapses into a solved state. We estimate the complexity of the attack by summing expected bit complexity in each step of the guessing. We measure the final complexity in bits: If we are guessing directly variables, we count the number of guessed variables. If we are guessing whole vectors $v \in V_i$ (which fixes $|X_i|$ variables at once), we add $\log_2 |V_i|$ for each guess (instead of $|X_i| > \log_2 |V_i|$).

We remark, that if we "guess" input and key bits, and use the spreading of constants, the system will collapse into a single solution (spreading of constants emulates the working of the corresponding encryption logic circuit). Thus the upper limit for the minimal complexity of the local reduction methods is the key size (in the single P-C pair scenario).

Let us suppose that we use iterated cipher, and increase the number of rounds of the evaluated cipher. The size of the system increases with each round (due to new intermediate variables). The complexity of the local reduction part scales polynomially (so the experiments are not significantly slower). The bit complexity of the attack depends on the guessing strategy. If we guess values of randomly chosen variables (or partial solutions of randomly chosen symbols), we can expect that the bit complexity will grow with the size of the system. In this case, we simulate completely uninformed attacker, and the result can be considered the upper bound on the bit-complexity of the attack. If the attack with random guessing has lower bit-complexity than the number of key-bits, then the cipher is fundamentally insecure. It does not make sense to use (or even attack) such a cipher, as even the uninformed attacker can use the selected local reduction method to break it.

The guessing strategy is suitable for evaluation if it converges to some fixed upper value as we increase the number of rounds of the evaluated cipher. The bound for the secure cipher should be the key size n_K. In our experiments the strategy having this effect is the "maximum impact" strategy, where we guess the value of variable that occurs in the highest number of equations (or when we guess the partial solution of the equation, that has the highest total occurrence of variables in other equations).

Attack on a cipher with r rounds with expected lower average complexity than $n_K - 1$ can be considered as a shortcut attack. If the complexity of the attack is higher than $n_K - 1$ for each round above t, we can consider $r - t$ as a security margin (or $(r - t)/r$ as a relative security margin). The main problem is that the margin can depend on the chosen local reduction method, and the guessing strategy, respectively. It does not inform us whether the cipher is secure, only that it is at most as secure as given by the worst margin from all known attacks.

The evaluation framework can be used to either compare different attacks (defined by the local reduction method and the guessing strategy) using the bit complexity measure. Furthermore, it can provide us with the expected dependency of the complexity of algebraic attacks on the number of rounds of the selected cipher. Moreover, two ciphers can be compared by the means of their security margin (computed with the strongest known attack on each of the ciphers).

5.1 Experimental Results

As an illustration of the method, we provide experimental evaluation of the block cipher DES, and its security against the algebraic attacks based on the method of syllogisms. The system of equations was constructed from the blocks describing 2 rounds of DES. In each 2-rounds we use 64-bit input and 64-bit output bits of the round as variables as well as 48-bit input and 32-bit output bits of the S-boxes for each round. Thus each additional 2-rounds add $2(48+32)+64$ new unknowns (inputs to next 2-rounds are outputs from the previous one). The nonlinear equations for the S-boxes are 10-sparse (6+4 bits, 2^6 partial solutions each), the rest of the system consists of linear equations for the XOR-s of individual bits (Feistel scheme, and key addition, respectively).

We use the following guessing strategies [23]:

1. RANDOM: Choose random symbol, guess partial solution.
2. MAXINFO: Local strategy, choose symbol with the best ratio k/l (minimal number of guesses to find the value of the maximum number of variables).
3. IMPACTs: Choose symbol with variables that influence the highest number of equations (in total).
4. IMPACTv: Choose variable that influences the highest number of equations.

The results are summarized in Figure 1. Random strategy can be used for up-to 6-round DES, for larger number of rounds the bit-complexity is higher than key-

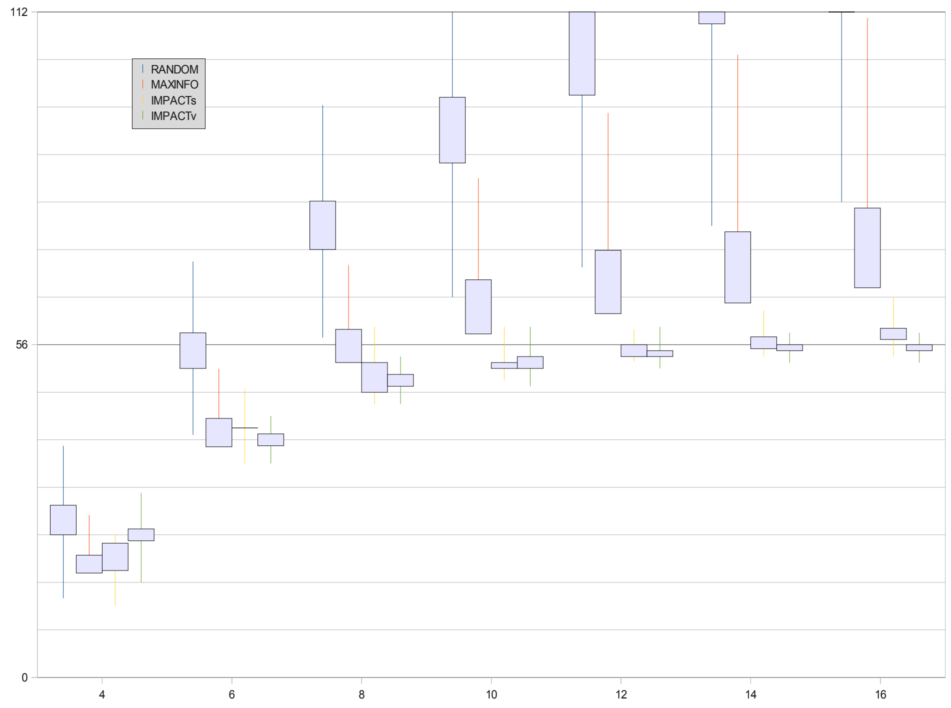


Figure 1. The bit-security of r-rounds DES against the attack with the method of syllogisms using different guessing strategies. Results are based on 1000 experiments, vertical lines connect minimum and maximum values, the thicker bars are bounded by the 1st, and 3rd quartile, respectively.

size. The MAXINFO strategy[3] that was proposed in [21] is better than uninformed guessing, but it does not converge to the key-size bound (but grows with the number of rounds). Both maximum impact strategies converge (statistically) to a key-size bound. In 8-round DES the attacker only needs to guess in average 50-bits of information (minimum was 46 bits). Although the brute-force attack might still be faster in practice, it depends on how fast can we implement the method of syllogisms in comparison with the DES encryption. Still the 8-round DES should not be considered secure. The expected complexity for the full DES is higher than 55 bits. We can see, that both "impact" strategies have lower minimum complexity (53 bits). It might be worth for the attacker to explore the experiments with lower than 55-bit complexity to construct a more effective attack on DES than brute-force, but it is usually cheaper to buy 4-times faster hardware.

[3] Originally this strategy was named GUESS, which is unfortunately not very descriptive.

We stress that all our guessing strategies are generic, i.e., they can be used for any sparse Boolean equation system. There might exist a strategy exploiting some specific properties of DES that can break more rounds, or provide better attack complexity for the given number of rounds. Moreover, some attacks on reduced round ciphers can be extended due to some specific properties of ciphers, as was demonstrated very recently by Courtois [6] in the case of the cipher GOST.

5.2 A Comparison with SAT Solvers

Although the approach provided in this paper is mostly experimental, it might be possible base for better understanding of the complexity of the algebraic crypto-analysis. We believe, that the results obtained by experiments with local reduction apply (in a qualitative way) to other methods used in algebraic cryptanalysis as well. To support this hypothesis, we have performed a series of experiments with SAT solvers. Namely, we compare our results on DES from Section 5.1 with estimated complexity of solving the (randomly) selected (round-reduced) DES instances with MiniSat 2.2.0 [7].

We have not run the whole experiments (which are quite costly in computational power). Instead we have estimated the required complexity in a way similar to [5]. Curtois and Bard provide a time (68 s) required to solve the instance of 6-round DES, when 20 bits of the key are fixed (and known). Given number of fixed bits g, and the solution time t_g respectively, we can estimate the time to solve the whole instance as $t_g 2^g$. We remark, that this estimate can be misleading. The SAT-solver in its basic setting will report the solution as soon as it is found, thus if we are "lucky", we get a solution "too fast", and the estimate is significantly skewed. The expected distribution of running times t_g is usually lognormal [4], so it is better to compute average estimate in logarithmic terms (instead of absolute times).

A different estimate can be obtained by fixing g *incorrect* bits of the key (randomly chosen), and measure the time to reject the value T_g. A local reduction method is more likely to reject the incorrect guess sooner (due to collisions) than confirm a correct one. on the other hand SAT solver is more likely to find the correct solution sooner than to reject the incorrect one. If we have to check all 2^g guesses of the fixed bits, the expected running time also depend on the architecture of the experiment. If the guesses are verified in series, we should base the estimate on the average time of the rejection. If guesses can be verified in parallel, it is possible to stop the verification of incorrect guesses as soon as the correct solution is found (so the minimal time applies).

To evaluate the complexity using SAT solvers, we run the local reduction experiment, and after each guess and reduction we store the corresponding (partially) reduced system. After the solution is found (or rejected), we convert the stored systems to CNF, and try to solve them with MiniSat. We start from the system, which was missing only one bit to be immediately solved by the method of syllogisms, then continue with the system missing 2 bits, etc. As the parameter g decreases, the running times t_g, T_g of the SAT solver increase. We stop the "unguessing" after the running times t_g, T_g are too long (MiniSat takes more than a day to complete).

There is a large variance in the number of decisions and the correspoding guessing times, see Table 1. Moreover, the estimated brute-force time (last column) is not monotone. This behaviour (along with more SAT solver based results that are out of scope of this paper) is explored in more details in [11].

g	Decisions $[10^3]$	t_g [s]	$t_g 2^g$ $[10^6$ Years]
55	70.0 ± 11.0	0.2 ± 0.1	173.49
54	94.0 ± 12.3	0.2 ± 0.1	121.91
53	154.5 ± 16.0	0.5 ± 0.1	138.24
52	199.3 ± 24.3	0.8 ± 0.3	117.63
51	243.7 ± 23.4	1.4 ± 0.5	101.36
50	294.6 ± 37.6	2.5 ± 1.0	87.69
49	415.5 ± 81.1	6.9 ± 4.0	123.34
48	577.8 ± 165.6	13.5 ± 8.3	120.55
47	800.2 ± 275.5	24.8 ± 14.5	110.67
46	2 511.4 ± 1 444.2	113.9 ± 87.7	253.90
45	6 228.8 ± 4 960.0	371.9 ± 369.4	414.64

Table 1. Number of decisions made by MiniSat 2.2.0, along with running times t_g (on Intel i7-3820, 3.60 GHz) reported when solving CNF's constructed from 14-round DES system reduced by the method of syllogisms after g-bits (suggested by IMPACTv strategy) were guessed. Results are averaged from 100 runs, reported along with the experimental standard deviation.

The results of our MiniSat experiments are summarized in Figure 2. As expected, the minimal times to verify the correct guess by MiniSat is significantly lower than the minimal time to reject incorrect guess. Moreover there is a higher difference between average and minimal estimates in these cases. Most notably, the minimal expected complexity to solve the system when using the correct guess is faster than brute force.[4] However, the average expected complexity, as well as the minimal and average complexity of rejecting incorrect guess is lower than brute-force complexity.

Figure 3 compares our estimates obtained by using the method of syllogisms (as implemented in the tool called sylog), and the results obtained by using MiniSat, respectively. Unlike Figure 1, we took into account also the (polynomial) running time required for a reduction (a dashed line shows the estimate without considering the growing cost of reduction). The running time of the sylog tool is inferior to highly optimized MiniSat. However, the dependence of the expected running times on the number of rounds for DES is very similar.

[4] Two different P-C pairs/keys gave different minimums: 2.5-times faster than brute force, and 11-times faster than brute-force, respectively. In both of these cases 16 bits of the key are correctly guessed, guessing 20 bits leads to a higher estimate.

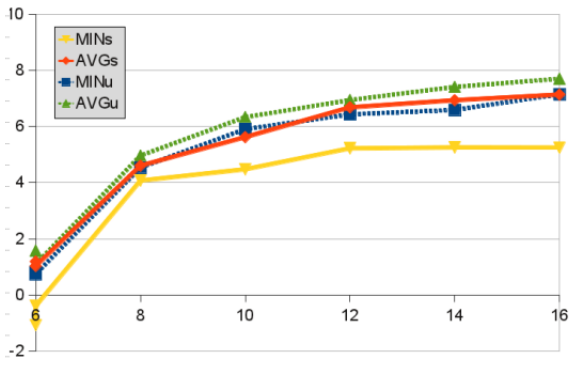


Figure 2. Expected minimal and average times required (on Intel E8200, 2.66 GHz) to solve round-reduced DES with MiniSat, in case of correct partial guess (solid lines), and incorrect partial guess (dotted lines). Results are in logarithmic scale with base 0 corresponding to the estimated brute-force effort on the same computer.

6 DISCUSSION AND CONCLUSIONS

In this paper we have examined a methodology that can be used for a fast evaluation of the complexity of generic algebraic attacks against ciphers. We use symbol representation of the cipher structure, and a polynomial time local reduction algo-

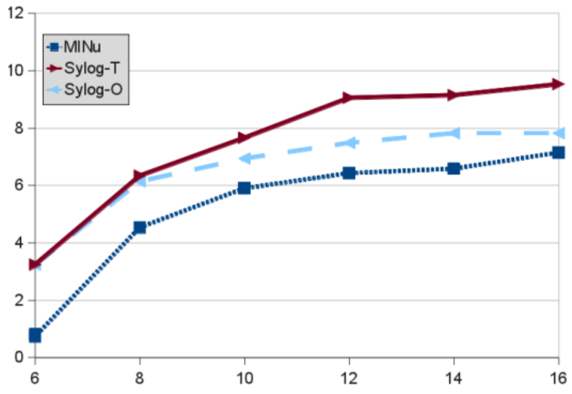


Figure 3. Expected minimal time required (on Intel E8200, 2.66 GHz) to reject a partial guess for round-reduced DES with MiniSat (dotted line) compared with the expected running time of the sylog tool (solid line). Dashed line corresponds to the estimated sylog complexity effort, if we ignore the polynomial factor in the complexity growth. Results are in logarithmic scale with base 0 corresponding to the estimated brute-force effort on the same computer.

rithm combined with informed or uninformed guessing to estimate the dimensions of the search tree. This allows us to predict the exponential part of the complexity of the whole algebraic attack, not only when using symbol based algorithms (such as gluing), but also when using SAT solver based attacks. As such, our method can be adapted to other applications where SAT solvers are used.

Although the expected complexity of attacks is exponential, there are specific instances where algebraic cryptanalysis can perform better then brute-force attacks on ciphers. This is intensified in cases of various experimental lightweight cipher proposals [8, 2, 3], which try to sacrifice some security margins to the speed or hardware resource consumption.

Symbol representation can be extended to MRHS form [13], which is especially suitable to model ciphers with low multiplicative complexity such as recently proposed LowMC [1]. We have proposed a new algorithm that can solve [22] the systems in MRHS form. The exponent in the complexity of this algorithm depends on the total number of right-hand sides (RHS) in the system. Some of the local reduction methods (such as agreeing) can be combined with MRHS representation to reduce the number of RHS in polynomial time, and thus reduce the complexity exponent for the whole system. It is an open question, whether any method that can be applied to symbol representation can be generalised also to MRHS equations. Furthermore, one can ask for each system what is the lowest possible number of RHS one can get with any local reduction method.

Finally, we would like to advise of some practical applications of local reduction methods. Algebraic attacks on ciphers can be mitigated by increasing the key space or other cipher parameters (such as number of rounds). On the other hand, there are various side channel attacks such as DPA [16], or fault attacks [12], that can break ciphers by measuring physical leakage from the cipher implementation. These side-channel attacks can be improved by combining them with algebraic attacks. Here symbol representation provides an advantage because we can capture the equation system as a collection of most probable hypotheses. When the attacker measures the physical leakage, he can try the attack based on local reduction, or just estimate its complexity with our methodology. If the attack complexity is too high, he can try further measurements, otherwise he can find the secret parameters by the algebraic attack.

Acknowledgement

This material is based upon work supported under the grant VEGA 1/0173/13, and VEGA 1/0159/17.

REFERENCES

[1] Albrecht, M. R.—Rechberger, C.—Schneider, T.—Tiessen, T.—Zohner, M.: Ciphers for MPC and FHE. In: Oswald, E., Fischlin, M. (Eds.): Advances in Cryptology – EUROCRYPT 2015. Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 9056, 2015, pp. 430–454.

[2] Antal, E.—Hromada, V.: A New Stream Cipher Based on Fialka M-125. Tatra Mountains Mathematical Publications, Vol. 57, 2013, No. 1, pp. 101–118, doi: 10.2478/tmmp-2013-0038.

[3] Antal, E.—Hromada, V.: A Micro-Controller Implementation of a Fialka M-125 Based Stream Cipher. Tatra Mountains Mathematical Publications, Vol. 60, 2014, No. 1, pp. 101–116, doi: 10.2478/tmmp-2014-0027.

[4] Bard, G.: Algebraic Cryptanalysis. Springer, 2009, doi: 10.1007/978-0-387-88757-9.

[5] Courtois, N.—Bard, G.: Algebraic Cryptanalysis of the Data Encryption Standard. In: Galbraith, S. (Ed.): Cryptography and Coding (Cryptography and Coding 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4887, 2007, pp. 152–169, doi: 10.1007/978-3-540-77272-9_10, doi: 10.1007/978-3-540-77272-9_10.

[6] Courtois, N. T.: Security Evaluation of GOST 28147-89 in View of International Standardisation. Cryptology ePrint Archive, Report 2011/211, 2011, `http://eprint.iacr.org/`.

[7] Een, N.—Sörensson, N.: The MiniSat Page. `http://minisat.se`.

[8] Eisenbarth, T.—Kumar, S.—Paar, C.—Poschmann, A.—Uhsadel, L.: A Survey of Lightweight-Cryptography Implementations. IEEE Design & Test of Computers, Vol. 6, 2007, pp. 522–533, doi: 10.1109/MDT.2007.178.

[9] Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases (F4). Journal of Pure and Applied Algebra, Vol. 139, 1999, No. 1-3, pp. 61–88, doi: 10.1016/S0022-4049(99)00005-5.

[10] Faugère, J.-C.: A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero (F5). Workshop on Applications of Commutative Algebra, Catania, Italy, April 3–6, 2002, ACM Press.

[11] Hromada, V.—Öllös, L.—Zajac, P.: Using SAT Solvers in Large Scale Distributed Algebraic Attacks Against Low Entropy Keys. Tatra Mountains Mathematical Publications, Vol. 64, 2015, No. 1, pp. 187–203, doi: 10.1515/tmmp-2015-0048.

[12] Hromada, V.—Varga, J.: Phase-Shift Fault Analysis of Trivium. Studia Scientiarum Mathematicarum Hungarica, Vol. 52, 2015, No. 2, pp. 205–220, doi: 10.1556/012.2015.52.2.1308.

[13] Raddum, H.: MRHS Equation Systems. In: Adams, C., Miri, A., Wiener, M. (Eds.): Selected Areas in Cryptography (SAC 2007). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 4876, 2007, pp. 232–245, doi: 10.1007/978-3-540-77360-3_15.

[14] Raddum, H.—Semaev, I.: New Technique for Solving Sparse Equation Systems. Cryptology ePrint Archive: Report 475/2006, `http://eprint.iacr.org/2006/475`, 2006.

[15] Raddum, H.—Semaev, I.: Solving Multiple Right Hand Sides Linear Equations. Designes, Codes and Cryptography, Vol. 49, 2008, No. 1-3, pp. 147–160, doi: 10.1007/s10623-008-9180-z.

[16] Repka, M.—Varchola, M.—Drutarovský, M.: Improving CPA Attack Against DSA and ECDSA. Journal of Electrical Engineering, Vol. 66, 2015, No. 3, pp. 159–163.

[17] Schilling, T. E.—Raddum, H.: Solving Equation Systems by Agreeing and Learning. In: Hasan, M. A., Helleseth, T. (Eds.): Arithmetic of Finite Fields (WAIFI 2010). Springer, Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 6087, 2010, pp. 151–165, doi: 10.1007/978-3-642-13797-6_11.

[18] Schilling, T.—Zajac, P.: Phase Transition in a System of Random Sparse Boolean Equations. Tatra Mountains Mathematical Publications, Vol. 45, 2010, pp. 93–105, doi: 10.2478/v10127-010-0008-7.

[19] Zajac, P.: On the Use of the Method of Syllogisms in Algebraic Cryptanalysis. Proceedings of the 1st Plenary Conference of the NIL-I-004, 2009, University of Bergen, pp. 21–30.

[20] Zajac, P.: Implementation of the Method of Syllogisms. Preprint, 2010.

[21] Zajac, P.: Solving Trivium-Based Boolean Equations Using the Method of Syllogisms. Fundamenta Informaticae, Vol. 114, 2012, No. 3–4, pp. 359–373.

[22] Zajac, P.: A New Method to Solve MRHS Equation Systems and Its Connection to Group Factorization. Journal of Mathematical Cryptology, Vol. 7, 2013, No. 4, pp. 367–381.

[23] Zajac, P.—Čagala, R.: Local Reduction and the Algebraic Cryptanalysis of the Block Cipher GOST. Periodica Mathematica Hungarica, Vol. 65, 2012, No. 2, pp. 239–255.

[24] Zakrevskij, A.—Vasilkova, I.: Reducing Large Systems of Boolean Equations. 4th International Workshop on Boolean Problems, Freiberg University, 2000, pp. 21–22.

Pavol Zajac is Associate Professor at the Institute of Computer Science and Mathematics, FEI STU in Bratislava, where he also obtained his Ph.D. in applied mathematics in 2008. His main research is cryptography. He currently works on the design and cryptanalysis of lightweight ciphers which can be used to protect security and privacy on mobile devices. Furthermore, he works on practical algorithms for post-quantum cryptography, which are also resistant to side-channel attacks.

Computing and Informatics, Vol. 37, 2018, 367–384, doi: 10.4149/cai_2018_2_367

XOR-BASED COMPACT TRIANGULATIONS

Abdelkrim Mebarki

Département d'Informatique
Faculté des Mathématiques et Informatique
Université des Sciences et de la Technologie d'Oran – Mohamed Boudiaf
USTO-MB, BP 1505 Oran El M'naouer, 31000, Oran Algérie
e-mail: `abdelkrim.mebarki@univ-usto.dz`

Abstract. Media, image processing, and geometric-based systems and applications need data structures to model and represent different geometric entities and objects. These data structures have to be time efficient and compact in term of space. Many structures in use are proposed to satisfy those constraints. This paper introduces a novel compact data structure inspired by the XOR-linked lists. The subject of this paper concerns the triangular data structures. Nevertheless, the underlying idea could be used for any other geometrical subdivision. The ability of the bitwise XOR operator to reduce the number of references is used to model triangle and vertex references. The use of the XOR combined references needs to define a context from which the triangle is accessed. The direct access to any triangle is not possible using only the XOR-linked scheme. To allow the direct access, additional information are added to the structure. This additional information permits a constant time access to any element of the triangulation using a local resolution scheme. This information represents an additional cost to the triangulation, but the gain is still maintained. This cost is reduced by including this additional information to a local sub-triangulation and not to each triangle. Sub-triangulations are calculated implicitly according to the catalog-based structure. This approach could be easily extended to other representation models, such as vertex-based structures or edge-based structures. The obtained results are very interesting since the theoretical gain is estimated to 38 % and the practical gain obtained from sample benches is about 34 %.

Keywords: XOR operator, XOR-linked list, XOR-based representation, triangular data structure, catalog-based structure

1 INTRODUCTION

Triangulations are widely used in almost all of the computer graphic applications. Several data structures are proposed to represent such subdivisions [1, 2]: vertex-based structures, edge-based structures, and triangle-based structures. All of these structures use an explicit indirection scheme to access to their elements. When dealing with huge triangulations, we need a large time to rearrange the structure in order to reduce the memory cost.

The trivial way to do this is either to compress the structure, or use an Out-of-Core structure. However, we need sometimes to still work In-Core in constant time, with not very huge triangulations, that is why we need compact data structures [3, 4] to delay as far as possible the swap between the main memory and the hard disk.

In this paper, a compact data structure is proposed that uses the bitwise XOR Operator to reduce the amount of references in the memory space. This idea is either applicable on vertex-based structures, edge-based structures, or triangle-based structures, and can be extended to 3D triangulations or any other subdivision.

1.1 Triangle-Based Representation

To illustrate the XOR-combined references idea, we use the explicit triangle-based representation (see Figure 1).

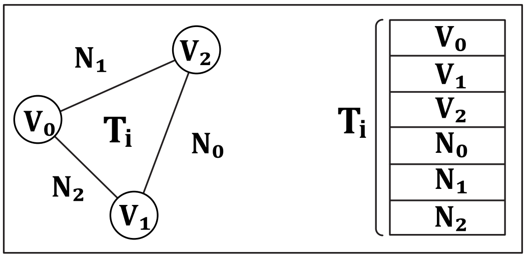


Figure 1. The explicit triangle-based representation. Each triangle maintains three references to its vertices and three references to its neighbors.

This representation [5] considers the triangle as the elementary object to represent the whole structure. Each triangle maintains references to its three vertices, and to its three neighbors. The whole triangulation is then represented using a table of triangles. Each entry in this table contains six references, three of them for the three triangle vertices, and the other three references indicate the three triangle neighbors. Since the number of triangles for a triangulation of n vertices is limited by $2n$, the global storage cost of such a triangulation is about $6n$ references.

The vertices in each triangle are indexed with 0, 1, and 2 in counterclockwise order. The neighbors are indexed in such a way that the neighbor indexed by i is opposite to the vertex with the same index.

For further details in this paper, two functions have to be defined to handle the item indices: $cw(i)$ and $ccw(i)$ which, given the index of a vertex in a triangle, compute the index of the next vertex of the same face in clockwise or counterclockwise order. Thus, for example the neighbor $cw(i)$ is the neighbor of the triangle which is next to neighbor i turning clockwise around the triangle. The triangle neighbor $cw(i)$ is also the first triangle encountered after the triangle when turning clockwise around vertex i of the triangle. In other words, the *ccw* and *cw* functions allow to enumerate the indices in cyclic way from 0 to 2 for the first function, and from 2 to 0 for the second function.

The remaining of this paper is organized as follows: Section 3 explains the major contribution of this paper. The Section 4 details the basic XOR linked date structure. In Section 5, the resolution scheme used to directly access triangles is presented. Section 6 presents some ideas to extend the XOR linked model to other representations. In Section 7, some sample benched results are presented to demonstrate the practical gain of this structure.

2 STATE OF THE ART

Data structures, and especially geometric data structures are used in many fields ranging from computer-aided design to finite elements [6].

The basic work has targeted the design of data structures [7] that are robust, fast and simple, while respecting the requirements of the different types and aims of the application [1]. This compromise is not always easy to manage, because applications are not always categorized. The efficiency in terms of execution time is required for real-time applications where the need to reduce the space used in memory usually happens in the background, while the economy in terms of memory footprint arises for handling large models, and induced in most cases a decrease of efficiency and loss of simplicity.

We will see in this part of the paper a description of the various solutions that have been proposed for the realization of geometric data structures to meet the various requirements imposed by the variety of applications ranging from simple and explicit data structures designed for small data sets, to compact or succinct data structures. We will also see the outlines of compression algorithms and data structures proposed for applications using auxiliary memories.

The most used structures in the state of the art are the primitive-based representations. Also known as index models or array-based models [8]. Since the triangulation can be seen as a set of vertices, edges, or faces, we can set these structures in three categories: edge-based structures, triangle-based structures, and vertex-based structures.

The first one deals with modes of representation that have been proposed for polygonal models in general, not just the triangulations. The basic object can be the edge or the half-edge. These structures have been developed around the modeling of solids and surfaces [9]. In such structures, the topological information is mainly contained in the edges of the object. A very large literature is available for this set [10, 11, 12, 13, 14, 15, 16, 17, 18, 19].

In this type of representations, the basic concept is the triangle. The basic structure uses two tables: one for the vertices where the geometric coordinates are stored, and the second one is for the triangles, where are stored the references of three vertices that define each triangle. This structure requires at least $6n$ references for a triangulation of n points.

This type of representation is widely used in practice [20, 21, 22], as it allows a minimum storage compared to other strategies, while maintaining a constant time response to incident requests.

In the vertex-based structures [23], the triangulation is represented as a graph of incidence between the nodes (vertices) of the triangulation. The structure is a list of vertices, where each vertex maintains its *degree* (the number of neighboring vertices), the list of these neighbors, and a mark indicating if the vertex is a vertex on the boundary or not. Given that the average degree of a vertex in a triangulation of n points is equal to 6, the cost of such a structure is $7n$ references [24, 25].

Compressing meshes (or triangulations) comprises encoding triangulation eliminating maximum redundancy by minimizing the entropy of the structure. The data structure after compression is unusable, and to be able to manipulate the structure or access items, you must decompress the entire structure and rebuild the explicit structure. For a detailed and comprehensive study of mesh compression techniques, a wide range of publications exists [26, 27, 28, 29, 30, 31, 32, 33].

2.1 Dealing with Huge Triangulations

The treatment of huge meshes of the order of billions of triangles is very limited by the above cited representations. Indeed, the indirection required to access vertices from a given face can be very costly when the index is very wide, even impossible when it exceeds the addressable range in the memory of the machine. Hence the importance of the *Out of Core* algorithms.

An intuitive solution is to not list the vertices, and to include directly the coordinates of the vertices in the faces. This solution that is called *Triangle Soup* would enable the faces of a mesh independently and avoid the step of indirection. But vertex update and neighborhood span is not as easy as in the indexed format. This technique was introduced for several types of applications [34, 35, 36]. The explicit representation of the parts of this mesh in working memory is usually indexed.

Another type of Out-of-Core approaches is the multi-resolution structures [37, 38, 39, 40, 41]. These structures allow access to the grid by induction on spatial subdivisions. These structures are also used to adjust the level of detail of the explicit structures. The involved common basic models are the trees (especially B-trees and its derivatives [42]).

For a data structure suitable for processing mesh, where this is done sequentially, Isenburg et al. [43] have proposed an ordered structure that they called *Streaming Mesh*. The goal is to have a format that allows switch the flow of mesh in the working memory to process it without the need to load the resident parts in the auxiliary memory. To do this, the vertices and the faces of the mesh are inserted into the same structure. Gradually, as the faces scrolled in the structure, the vertices are introduced in the flow, or finalized when they are no longer referenced by any face. This format was used to compress large meshes [44, 45, 46], for the simplification of meshes [47], for the construction of the Delaunay triangulation [48]. In [49], this format is modified to allow direct access to neighboring information.

2.2 Compact Data Structures

Between explicit data structures, and the coding of triangulations, there is a third set of structures called *Compact Data Structures.* This type of structure has two objectives:

- Conceive a structure that is locally accessible: That means that we can access to its components in constant time.
- At the same time, we have to minimize the space occupied by the structure, so that it holds in working memory, and consequently to delay as long as possible the use of the transfer with the auxiliary memory.

The first compact data structure was proposed by Kallmann and Thalmann, they called it *Star Vertices* [24]. It is a vertex-based representation: each vertex handles a list of all of its adjacent vertices (the vertex stores the size of this list), resulting in $6n$ references plus n integers (sizes of lists) to represent the whole triangulation. However, the internal structure has no longer an explicit representation of faces, and queries cost time is proportional to the degree of the involved vertex.

Blandford et al. [50, 4, 51] proposed a compact data structure for representing simplicial meshes, requiring in practice 40 bpt [1]. The representation does admit an edge-based or vertex-based representation, providing basic update operations and standard local navigation between triangles (performing these operations takes $O(1)$ time for the case of meshes with bounded vertex degree). To gain in memory, difference vertex labels are used instead of real pointers, and a preprocessing step consisting of relabelling vertices, for reducing the differences, is needed. This approach takes advantage of properties of graphs with small separators and require some assumptions on the input data.

Devillers et al. have proposed an optimal way of representing a triangulation of n points using $3.24n$ bits [52, 53], with an additional storage cost which is asymptotically negligible (in the case of a triangulation of a topological sphere; for the triangulation bounded by a polygon of arbitrary size the cost is 2.17 bpt. The idea is to gather triangles in tiny patches of size between $\frac{\log n}{12}$ and $\frac{\log n}{4}$, and to introduce a graph of patches to describe adjacency relations between them. Each patch is then represented by a reference to a catalog, consisting of all different tiny patches

[1] bits per triangle

of size less than $\frac{\log n}{4}$. The whole size of all references to the catalog gives the dominant term of 3.24 bpv [2], while the representation of the graph of patches requires a negligible amount of space.

In [54], the authors proposed some catalogs and evaluate in detail the amount of storage needed for representing triangulations using this approach. The implementation showed that the expected improvements are indeed obtained in practice.

In [55], Aleardi et al. proposed a new way of designing compact data structures which can be dynamically maintained. They described a new class of data structures, called *Editable SQuad (ESQ)*, offering the same navigational and storage performance as previous works, while supporting local editing in amortized constant time.

The proposed data structures are simple to implement and provided with an analysis of worst case storage bounds. The simplest solution uses 6 rpv [3], while supporting updates operations in $O(1)$ amortized time: this is obtained with a reordering of input data which allows to encode the map from triangles to vertices. The most compact data structure makes use of a grouping strategy between adjacent triangles, and uses only 4.8 rpv, while still supporting efficient navigation and update operations.

In [56], Gurung et al. proposed a data structure for representing the connectivity of manifold triangle meshes that they called *LR (Laced Ring).* This structure provides the option to store on average either 1.08 rpt [4] or 26.2 bpt. Its construction, from an input mesh that supports constant-time adjacency queries, has linear space and time complexity, and involves ordering most vertices along a nearly-Hamiltonian cycle.

3 CONTRIBUTION

In this paper, we present a novel compact data structure for 2D triangulations based on the bitwise XOR operator [57].

The underlying idea concerns the indexed structures. The XOR-scheme is used to reduce the number of explicit references by combining them two by two.

A bitwise operation operates on binary numerals at the level of their individual bits. It is a fast, primitive action directly supported by the processor, and is used to manipulate values for comparisons and calculations.

A bitwise XOR takes two bit patterns of equal length and performs the logical *exclusiveOR* operation on each pair of corresponding bits. The result in each position is 1 if only the first bit is 1 or only the second bit is 1, but will be 0 if both are 0 or both are 1. Otherwise, we perform the comparison of two bits, being 1 if the two bits are different, and 0 if they are the same.

[2] bits per vertex

[3] references per vertex

[4] references per triangle

The basic idea of the proposed structure is to combine references in the same way as the XOR-linked lists.

The XOR-linked list merges the next and the previous references into one single field using the bitwise XOR operator (see Figure 2). The access to a given element in the linked list is only sequential, that means that we access each element either from its predecessor neighbor, or from its successor neighbor. The obtained gain by replacing the explicit next and previous references with their XOR-combined reference is 50 %, since these two references are replaced by only one reference.

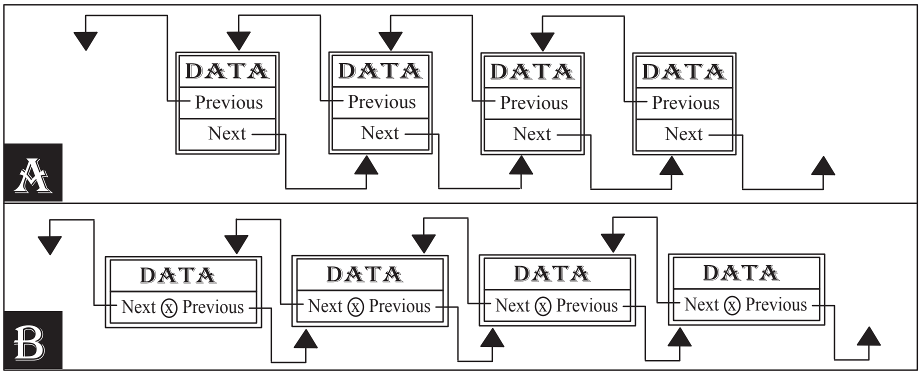


Figure 2. A) The explicit representation of liked list. B) The XOR-linked list: The implicit representation of a linked list using the XOR operator to join the next and previous pointers.

The resolution of the references (that means retrieving the real previous and next references) is done on the fly each time we access to an element. Since the access to the element is sequential, each element is accessed either from its predecessor or successor in the list. So, in the first case, we retrieve the successor reference from the predecessor one and the XOR combined references, and in the second time, we retrieve the predecessor reference from the successor one and the XOR combined references.

4 THE BASIC XOR-BASED TRIANGULAR STRUCTURE

When walking in a triangulation, we access each triangle from its neighborhood. This context defines for each accessed triangle at least one of its neighbor references, and at least two of its three vertex references.

Let us consider the triangle T_i. Where: V_0, V_1, V_2 are its vertices; N_0, N_1, N_2 are its neighbors.

The XOR-based representation of the triangle T_0 is defined by:

$$T_0 \begin{cases} V_0 \oplus V_1 \oplus V_2, \\ N_0 \oplus N_1, \\ N_0 \oplus N_2. \end{cases}$$

As we can remark, the number of references used to represent each triangle in this XOR-based representation is only three, instead of six references in the explicit triangle-based representation. The gain obtained is then equal to 50 %.

When accessing this triangle, coming from a neighbor N_i sharing the two vertices $V_{cw(i)}$ and $V_{ccw(i)}$, the above XOR-based representation allows us to retrieve all of the three vertex references and all of the three neighbor references as follows:

$$\left.\begin{array}{ll} V_0 \oplus V_1 \oplus V_2, & V_{cw(i)}, V_{ccw(i)} \\ N_0 \oplus N_1, & N_i \\ N_0 \oplus N_2, & \end{array}\right\} \Longrightarrow \begin{array}{l} V_i, V_{cw(i)}, V_{ccw(i)}, \\ N_i, N_{cw(i)}, N_{ccw(i)}. \end{array}$$

In order to affect the indices to the vertices and neighbors, a conventional order can be adopted: The vertices are sorted, and the indices are assigned from 0 to 2 in the same order for all the triangles of the subdivision.

The main disadvantage that we have here is the lack of direct access to triangles. The basic scheme presented here needs to have a context each time we access to a triangle. This context is defined as follows:

- One of its neighbors.
- Two of its three vertices.

If we want to enumerate all of the triangles (or read all of them from the triangle container for example), we lack this context. The only way to enumerate all of the triangles is to walk in the triangulation spanning all of its simplices.

5 THE RESOLUTION SCHEME

To be able to directly access to any triangle, we define a Local Resolution Scheme. The Local Resolution Scheme is an algorithm that can extract all of the triangle references from a restricted local neighboring sub-triangulation. That means that we need not to browse the whole triangulation to extract a given triangle references, just a local restricted neighborhood is sufficient to retrieve these references.

To establish such a scheme, we consider a local neighborhood according to a subdivision of the triangulation into packages likewise the catalog-based data structure [54] using only two packages: quadrangles and pentagons. Although we can go further by defining largest catalogs, the quadrangles and pentagons are widely sufficient.

As demonstrated in [54], any triangulation can be represented as a decomposition of sub-triangulations using only these catalog packages.

The global structure remains triangle-based, and the mentioned subdivision is only virtual. We need only to associate triangles belonging to packages between them, to be able to retrieve all of the lacking references.

5.1 Rearranging the Triangulation

In order to minimize additional costs, the triangles of the original triangulation have to be rearranged in such way that each package has its triangles stored in a contiguous area in the memory (that means that its triangles have sequential indices in the triangle container).

If we take the advantage of the memory size word that are at least four aligned bytes (for a 32 bits machine), we can remark that the two least significant bits are useless in the reference. We can squat these bits to store the indices of the triangles into their belonging packages: The triangles of a given quadrangle are indexed 0 and 1, and the triangles of a pentagon are indexed 0, 1, and 2.

Now, all of the triangles are stored in the memory, and all of the package triangles are stored consequently.

Thus, when we access to each triangle directly in the container, we consult its index, and by consulting its direct contiguous neighborhood indices, we can easily determine to which kind of package it belongs, and who are its neighbors in this package.

5.2 Resolution Scheme into a Quadrangle

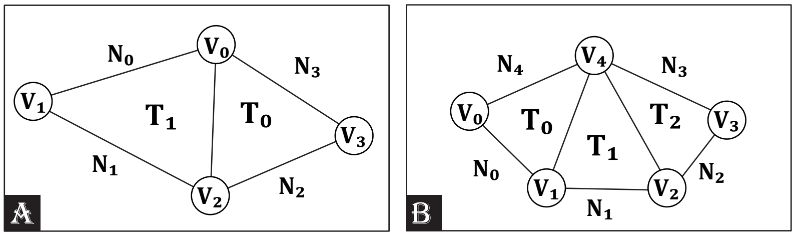


Figure 3. a) Two triangles grouped into one quadrangle patch, b) three triangles grouped into one pentagon patch

Let us suppose that we need to access to a specified triangle in a given quadrangle, the associated resolution scheme of the two triangles has to allow us to calculate all of the references. Supposing we have the two adjacent triangles T_0 and T_1 those belong to the same quadrangle (see Figure 3). The triangle T_0 is represented as follows:

- *vertices*: V_0, V_2, V_3,
- *neighbors*: N_2, N_3, T_1.

The triangle T_1 is represented as follows:

- *vertices*: V_0, V_1, V_2,
- *neighbors*: N_1, T_0, N_0.

The XOR-schemes of the two triangles are:

$$T_0 \begin{cases} V_0 \oplus V_2 \oplus V_3, \\ N_2 \oplus N_3, \\ N_2 \oplus T_1, \end{cases}$$

$$T_1 \begin{cases} V_0 \oplus V_1 \oplus V_2, \\ N_1 \oplus T_0, \\ N_1 \oplus N_0. \end{cases}$$

Since we have the two triangle references T_0 and T_1, we can resolve all of the other two triangle references using this scheme system and two additional fields: the V_0 and V_2 references. The resolution can be done as follows:

$$\left.\begin{array}{l} V_0 \oplus V_2 \oplus V_3, \\ V_0 \oplus V_1 \oplus V_2, \\ V_0, V_2, \end{array}\right\} \Longrightarrow V_0, V_1, V_2, V_3,$$

$$\left.\begin{array}{l} T_0, T_1, \\ N_2 \oplus N_3, N_2 \oplus T_1, \\ N_1 \oplus T_0, N_1 \oplus N_0, \end{array}\right\} \Longrightarrow N_0, N_1, N_2, N_3.$$

The indices of the vertices and neighbors in each triangle can be established by sorting the vertices, and assigning the indices according to the vertex orders as described in the convention cited in Section 4.

5.3 Resolution Scheme into a Pentagon

Let us suppose now that we need to access to a specified triangle in a given pentagon, the associated resolution scheme of the three triangles has to allow us to calculate all of the references. Supposing we have three adjacent triangles T_0, T_1, and T_2 belonging to the same pentagon (see Figure 3), the triangle T_0 is represented as follows:

- *vertices*: V_0, V_1, V_4,
- *neighbors*: T_1, N_4, N_0.

The triangle T_1 is represented as follows:

- *vertices*: V_1, V_2, V_4,
- *neighbors*: T_2, T_0, N_1.

The triangle T_2 is represented as follows:

- *vertices*: V_2, V_3, V_4,
- *neighbors*: N_3, T_1, N_2.

The XOR-schemes of the three triangles are the following:

$$T_0 \begin{cases} V_0 \oplus V_1 \oplus V_4, \\ T_1 \oplus N_4, \\ T_1 \oplus N_0, \end{cases}$$

$$T_1 \begin{cases} V_1 \oplus V_2 \oplus V_4, \\ T_2 \oplus T_0, \\ T_2 \oplus N_1, \end{cases}$$

$$T_1 \begin{cases} V_2 \oplus V_3 \oplus V_4, \\ N_3 \oplus T_1, \\ N_3 \oplus N_2. \end{cases}$$

Using the same approach, we can retrieve all of the triangle references into a pentagon, using their XOR schemes and two additional words: the V_1 and V_4 references. These words that are the middle vertices of the pentagon are considered as the key gate of the package, that represent the local context allowing us to retrieve all of the package references. The resolution is as follows:

$$\left.\begin{array}{l} V_0 \oplus V_1 \oplus V_4, \\ V_1 \oplus V_2 \oplus V_4, \\ V_2 \oplus V_3 \oplus V_4, \\ V_1, V_4, \end{array}\right\} \Longrightarrow V_0, V_1, V_2, V_3, V_4,$$

$$\left.\begin{array}{l} T_0, T_1, T_2, \\ T_1 \oplus N_4, T_1 \oplus N_0, \\ T_2 \oplus T_0, T_2 \oplus N_1, \\ N_3 \oplus T_1, N_3 \oplus N_2, \end{array}\right\} \Longrightarrow N_0, N_1, N_2, N_3, N_4.$$

Using the same convention as in Section 4, the indices of the vertices and faces in each triangle can be established by sorting the vertices, and assigning the indices according to the vertex orders.

5.4 The Estimated Gain

The global gain in the case of the basic XOR-based structure is equal to 50 % since we represent each triangle using three references instead of six. However, in the proposed XOR-based structure allowing direct access to triangles, the gain is reduced.

For each two triangles grouped into a quadrangle, we add two additional references and for each three triangles grouped into a pentagon, we add two additional references.

Theoretically, the maximum gain we can obtain corresponds to the case where all the triangles are grouped into pentagons. In this case, and for a triangulation of n vertices, we can obtain $2n$ triangles, and thus $\frac{2n}{3}$ pentagons.

In this configuration, the global cost is equal to $\frac{22n}{3}$ references instead of $12n$ in the explicit triangle-based original structure. The gain is then equal to 38 %.

The worst gain we can obtain is when all of the triangles are grouped into quadrangles. In this case, the global cost is equal to $8n$. The gain is then equal to 33 %.

6 EXTENSION TO VEREX-BASED AND EDGE-BASED STRUCTURES

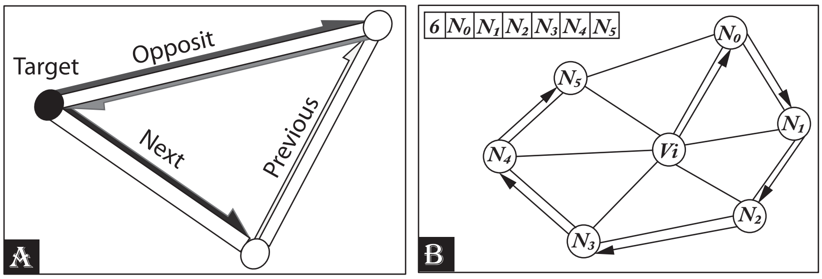


Figure 4. a) Half-edge-based representation, b) vertex-based representation

The XOR-linked representation is not restricted to triangle-based structures. It is also applicable to vertex-based and edge-based structures (see Figure 4).

6.1 Vertex-Based Representation

In the vertex-based structure, the triangulation is represented as a set of vertices. Each vertex is associated to a list of its incident (neighbors) vertices. The whole triangulation is then a set of lists: Each list corresponds to a vertex, including its degree (number of its neighbors), and the list of these neighbor references. The global cost for this representation is equal to $6n$ (where n is the number of vertices). Using directly the basic XOR-linked list to implement the vertex lists, the global gain cost is about 50 %.

6.2 Edge-Based Representation

Let us consider the basic half-edge scheme, where each half-edge stores four references:

- the reference of the target or the departure vertex;
- the reference of the opposite half-edge;
- the reference of the previous half-edge in the same face;
- the reference of the next half-edge in the same face.

This representation is similar to the XOR-linked list, and could be converted to a XOR-based scheme easily. The *next* and *previous* references would be replaced by the result of their XOR combination ($previous \oplus next$).

7 EXPERIMENTAL RESULTS

In order to evaluate the practical impact of the proposed structure, the XOR-based scheme is used to represent a sample Delaunay triangulation in static mode. That means that the Delaunay triangulation is constructed, and then converted from the explicit triangle-based representation to the XOR-based representation. The obtained results are shown in Table 1:

Number of points	100 000 points	1 000 000 points
Number of triangles	199 298	1 997 498
Number of quadrangles	98 764	997 270
Number of pentagons	590	986
Gain in Memory	33.38 %	33.34 %
Browsing time in Explicit Structure	1.661 s	16.476 s
Browsing time in XOR-Based Structure	1.966 s	18.152 s
Loss in Time	15.51 %	9.23 %

Table 1. Number of packages, gain in memory and browsing time for two XOR-based triangulations

As we can remark, the practical results are close to the theoretical expectations. We gain a third of the memory space compared to the explicit triangle-based structure. The loss in execution time is not very significant, since we lose less than 15 % of time. This execution time is calculated by browsing the whole triangulation and outputting all of the triangulation elements into an external file.

8 CONCLUSION

This paper has presented a novel approach for geometrical data structures inspired by the XOR-linked lists. The basic idea is to combine different references using the XOR operator to reduce by two the number of references. Unfortunately, this ratio is not reached if we need to maintain a direct access to each element of the structure. To guarantee the direct access, a local resolution scheme is defined using additional information. With this additional information, the structure remains beneficial and useful. The estimated and the obtained practical gain could be improved, if this method is combined with other compact data structures.

Acknowledgements

I would like to thank Olivier Devillers and Sylvain Pion from INRIA, France, for the very interesting, helpful and fruitful discussions which have oriented me in the elaboration of this paper.

REFERENCES

[1] Goodrich, M. T.—Ramaiyer, K.: Geometric Data Structures. In: Sack, J.-R., Urrutia, J. (Eds.): Handbook of Computational Geometry. Chapter 10. Elsevier Science, 2000, pp. 463–489.

[2] Mebarki, A.: Les Structures de Données Triangulaires Compactes: Théorie et Implémentation. Editions Universitaires Européennes, 2013 (in French).

[3] Mebarki, A.: Implantation de Structures de Données Compactes pour les Triangulations. Ph.D. thesis, Université de Nice-Sophia Antipolis, France, April 2008 (in French).

[4] Blandford, D. K.—Blelloch, G. E.—Cardoze, D. E.—Kadow, C.: Compact Representations of Simplicial Meshes in Two and Three Dimensions. International Journal of Computatinal Geometry and Applications, Vol. 15, 2005, No. 1, pp. 3–24, doi: 10.1142/S0218195905001580.

[5] Boissonnat, J.-D.—Devillers, O.—Pion, S.—Teillaud, M.—Yvinec, M.: Triangulations in CGAL. Computational Geometry, Vol. 22, 2002, No. 1-3, pp. 5–19, doi: 10.1016/S0925-7721(01)00054-2.

[6] Zachmann, G.—Langetepe, E.: Geometric Data Structures for Computer Graphics. Proceedings of ACM SIGGRAPH, July 27–31, 2003.

[7] Mehta, D. P.—Sahni, S.: Handbook of Data Structures and Applications. Chapman & Hall/CRC Computer and Information Science Series. Chapman & Hall/CRC, 2004, doi: 10.1201/9781420035179.

[8] Castelli Aleardi, L.—Devillers, O.: Explicit Array-Based Compact Data Structures for Planar and Surface Meshes. XIV Spanish Meeting on Computational Geometry, Alcala de Henares, Spain, 2011, Special edition for Ferran Hurtado Birthday.

[9] Mäntylä, M.: An Introduction to Solid Modeling. Computer Science Press, Inc., New York, NY, USA, 1987.

[10] Baumgart, B. G.: Winged Edge Polyhedron Representation. Technical report, Stanford University, Stanford, CA, USA, 1972, doi: 10.21236/AD0755141.

[11] Baumgart, B. G.: Winged-Edge Polyhedron Representation for Computer Vision. National Computer Conference, May 1975.

[12] Baumgart, B. G.: Geometric Modeling for Computer Vision. Ph.D. thesis, Stanford University, USA, August 1974.

[13] Preparata, F. P.—Shamos, M. I.: Computational Geometry: An Introduction. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[14] Guibas, L. J.—Stolfi, J.: Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagrams. Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing (STOC '83), ACM Press, New York, NY, USA, 1983, pp. 221–234, doi: 10.1145/800061.808751.

[15] Weiler, K.: Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments. IEEE Computer Graphics and Applications, Vol. 5, 1985, No. 1, pp. 21–40, doi: 10.1109/MCG.1985.276271.

[16] Lienhardt, P.: Subdivisions of n-Dimensional Spaces and n-Dimensional Generalized Maps. Proceedings of the Fifth Annual Symposium on Computational Geometry (SCG '89), ACM Press, New York, NY, USA, 1989, pp. 228–236, doi: 10.1145/73833.73859.

[17] Halbwachs, Y.—Hjelle, Ø.: Generalized Maps in Geological Modeling: Object-Oriented Design of Topological Kernels. In: Langtangen, H. P., Bruaset, A. M., Quak, E. (Eds.): Advances in Software Tools for Scientific Computing. Springer-Verlag, Lecture Notes in Computational Science and Engineering, Vol. 10, 1999, pp. 339–356.

[18] Hjelle, Ø.—Dæhlen, M.: Triangulations and Applications. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[19] Campagna, S.—Kobbelt, L.—Seidel, H.-P.: Directed Edges – A Scalable Representation for Triangle Meshes. Journal of Graphic Tools, Vol. 3, 1998, No. 4, pp. 1–11, doi: 10.1080/10867651.1998.10487494.

[20] Shewchuk, J. R.: Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In: Lin, M. C., Manocha, D. (Eds.): Applied Computational Geometry Towards Geometric Engineering. Springer-Verlag, Lecture Notes in Computer Science, Vol. 1148, 1996, pp. 203–222, doi: 10.1007/BFb0014497.

[21] TRIANGLE: Mesh Generator and Delaunay Triangulator. 2014.

[22] CGAL: Computational Geometry Algorithms Library. `www.cgal.org`.

[23] Cline, A. K.—Renka, R. J.: A Storage-Efficient Method for Construction of a Thiessen Triangulation. Rocky Mountain Journal of Mathematics, Vol. 14, 1984, No. 1, pp. 119–140.

[24] Kallmann, M.—Thalmann, D.: Star Vertices: A Compact Representation for Planar Meshes with Adjacency Information. Journal of Graphics Tools, Vol. 6, 2001, No. 1, pp. 7–18, doi: 10.1080/10867651.2001.10487533.

[25] Kallmann, M.: Object Interaction in Real-Time Virtual Environments. Ph.D. thesis, Swiss Federal Institute of Technology (EPFL), January 2001, Thesis number 2347.

[26] Alliez, P.—Gotsman, C.: Recent Advances in Compression of 3D Meshes. In: Dodgson, N. A., Floater, M. S., Sabin, M. A. (Eds.): Advances in Multiresolution for Geometric Modelling, Mathematics and Visualization. Springer, Berlin, Heidelberg, 2005, pp. 3–26.

[27] Gumhold, S.: Mesh Compression. Ph.D. thesis, University of Tübingen, July 2000.

[28] Gotsman, C.—Gumhold, S.—Kobbelt, L.: Simplification and Compression of 3D Meshes. Proceedings of the European Summer School on Principles of Multiresolution in Geometric Modelling (PRIMUS), August 2001, pp. 319–361.

[29] Isenburg, M.: Compression and Streaming of Polygon Meshes. Ph.D. thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2005.

[30] Lewiner, T.: Mesh Compression from Geometry. Ph.D. thesis, Université de Pierre et Marie Curie (Paris VI), Paris, December 2005.

[31] Peng, J.—Kim, C.-S.—Jay Kuo, C. C.: Technologies for 3D Mesh Compression: A Survey. Journal of Visual Communication and Image Representation, Vol. 16, 2005, No. 6, pp. 688–733.

[32] Rossignac, J.: 3D Mesh Compression. Technical Report GIT-GVU-03-21, Georgia Institute of Technology, Atlanta, GA, USA, 2003.

[33] Rossignac, J.: 3D Mesh Compression. In: Hansen, C., Johnson, C. R. (Eds.): Visualization Handbook. Academic Press, Inc., Orlando, FL, USA, 2004, pp. 339–356.

[34] Lindstrom, P.: Out-of-Core Simplification of Large Polygonal Models. Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00), ACM Press/Addison-Wesley Publishing Co, New York, NY, USA, 2000, pp. 259–262, doi: 10.1145/344779.344912.

[35] Lindstrom, P.—Silva, C. T.: A Memory Insensitive Technique for Large Model Simplification. Proceedings of the Conference on Visualization (VIS '01), IEEE Computer Society, Washington, DC, USA, 2001, pp. 121–126, doi: 10.1109/VISUAL.2001.964502.

[36] Wu, J.—Kobbelt, L.: A Stream Algorithm for the Decimation of Massive Meshes. Proceedings of Graphics Interface 2003, CIPS, Canadian Human-Computer Communication Society and A. K. Peters Ltd., June 2003, pp. 185–192. ISBN 1-56881-207-8, ISSN 0713-5424.

[37] De Floriani, L.—Kobbelt, L.—Puppo, E.: A Survey on Data Structures for Level-of-Detail Models. In: Dodgson, N. A., Floater, M. S., Sabin, M. A. (Eds.): Advances in Multiresolution for Geometric Modelling. Mathematics and Visualization Series, Springer, Berlin, Heidelberg, 2005, pp. 49–74.

[38] Cignoni, P.—Montani, C.—Rocchini, C.—Scopigno, R.: External Memory Management and Simplification of Huge Meshes. IEEE Transactions on Visualization and Computer Graphics, Vol. 9, 2003, No. 4, pp. 525–537, doi: 10.1109/TVCG.2003.1260746.

[39] Samet, H.: The Design and Analysis of Spatial Data Structures. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[40] Samet, H.: Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[41] Shaffer, E.—Garland, M.: A Multiresolution Representation for Massive Meshes. IEEE Transactions on Visualization and Computer Graphics, Vol. 11, 2005, No. 2, pp. 139–148, doi: 10.1109/TVCG.2005.18.

[42] Silva, C. T.—Chiang, Y.-J.—El-Sana, J.—Lindstrom, P.: Out-of-Core Algorithms for Scientific Visualization and Computer Graphics. IEEE Visualization Conference, 2002, Boston, Massachusets, Course Notes.

[43] Isenburg, M.—Lindstrom, P.: Streaming Meshes. Proceedings of the 16th IEEE Visualization Conference (VIS 2005), October 23–28, 2005, Minneapolis, MN, USA, IEEE Computer Society, 2005.

[44] Isenburg, M.—Lindstrom, P.—Snoeyink, J.: Streaming Compression of Triangle Meshes. Proceedings of the Third Eurographics Symposium on Geometry Processing (SGP '05), Eurographics Association, Aire-la-Ville, Switzerland, 2005, Art. No. 111, doi: 10.1145/1187112.1187276.

[45] Isenburg, M.—Gumhold, S.: Out-of-Core Compression for Gigantic Polygon Meshes. ACM Transactions on Graphics (TOG) – Proceedings of ACM SIGGRAPH 2003, Vol. 22, 2003, No. 3, pp. 935–942, doi: 10.1145/1201775.882366.

[46] Isenburg, M.—Lindstrom, P.—Gumhold, S.—Shewchuk, J.: Streaming Compression of Tetrahedral Volume Meshes. Proceedings of Graphics Interface 2006 (GI '06), Canadian Information Processing Society, Toronto, Ontario, Canada, 2006, pp. 115–121.

[47] Isenburg, M.—Lindstrom, P.—Gumhold, S.—Snoeyink, J.: Large Mesh Simplification Using Processing Sequences. Proceedings of the 14th IEEE Visualization 2003 (VIS '03), IEEE Computer Society, Washington, DC, USA, 2003, pp. 465–472, doi: 10.1109/VISUAL.2003.1250408.

[48] Isenburg, M.—Liu, Y.—Shewchuk, J.—Snoeyink, J.: Streaming Computation of Delaunay Triangulations. ACM SIGGRAPH 2006 Papers (SIGGRAPH '06), ACM Press, New York, NY, USA, 2006, pp. 1049–1056, doi: 10.1145/1179352.1141992.

[49] Abid, K.—Mebarki, A.—Hidouci, W. K.: Modified Streaming Format for Direct Access Triangular Data Structures. International Journal of Image, Graphics, and Signal Processing (IJIGSP), Vol. 6, 2014, No. 2, pp. 14–22.

[50] Blandford, D. K.—Blelloch, G. E.—Cardoze, D. E.—Kadow, C.: Compact Representations of Simplicial Meshes in Two and Three Dimensions. The 12th International Meshing Roundtable, 2003, pp. 135–146.

[51] Blandford, D. K.: Compact Data Structures with Fast Queries. Ph.D. thesis, Carnegie Mellon University, USA, February 2006.

[52] Castelli Aleardi, L.—Devillers, O.—Schaeffer, G.: Succinct Representation of Triangulations with a Boundary. Proceedings of Workshop on Algorithms and Data Structures (WADS 2005). Springer, Lecture Notes in Computer Science, Vol. 3608, 2005, pp. 134–145.

[53] Castelli Aleardi, L.—Devillers, O.—Schaeffer, G.: Succinct Representations of Planar Maps. Theoretical Computer Science, Vol. 408, 2008, No. 2-3, pp. 174–187, doi: 10.1016/j.tcs.2008.08.016.

[54] Castelli Aleardi, L.—Devillers, O.—Mebarki, A.: Catalog-Based Representation of 2D Triangulations. International Journal of Computational Geometry and Applications, Vol. 21, 2011, No. 4, pp. 393–402.

[55] Castelli Aleardi, L.—Devillers, O.—Rossignac, J.: ESQ: Editable SQuad Representation for Triangle Meshes. 25th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI 2012), Ouro Preto, Brazil, August 22–25, 2012, pp. 110–117.

[56] Gurung, T.—Luffel, M.—Lindstrom, P.—Rossignac, J.: LR: Compact Connectivity Representation for Triangle Meshes. ACM Transactions on Graphics, Vol. 30, 2011, No. 4, pp. 67:1–67:8.

[57] ISO/IEC 14882. Programming Languages C++. International Organisation for Standardisation, Geneva, Switzerland, 2003.

Abdelkrim Mebarki is currently Assistant Professor at the Science and Technology University of Oran (Mohamed Boudiaf). He received his Ph.D. degree from the University of Nice Sophia Antipolis (France) in 2008. He has his Master degree from the same university and the degree of state engineer from the Science and Technology University of Oran (Mohamed Boudiaf). He is now Assistant Researcher Professor in the same university. His research works include scientific visualization and triangular data structures.

Computing and Informatics, Vol. 37, 2018, 385–404, doi: 10.4149/cai_2018_2_385

ANALYSIS OF ITERATED GREEDY HEURISTIC FOR VERTEX CLIQUE COVERING

David Chalupa

Operations Research Group, Department of Materials and Production
Aalborg University, Fibigerstræ de 16
Aalborg 9220, Denmark
&
Computer Science, School of Engineering and Computer Science
University of Hull, Cottingham Road, Hull HU6 7RX, United Kingdom
e-mail: `dc@m-tech.aau.dk`

Jiří Pospíchal

Faculty of Natural Sciences, University of Ss. Cyril and Methodius
Námestie J. Herdu 2, 917 01 Trnava, Slovakia
e-mail: `jiri.pospichal@ucm.sk`

Abstract. The aim of the vertex clique covering problem (CCP) is to cover the vertices of a graph with as few cliques as possible. We analyse the iterated greedy (IG) algorithm for CCP, which was previously shown to provide strong empirical results for real-world networks. It is demonstrated how the techniques of analysis for randomised search heuristics can be applied to IG, and several practically relevant results are obtained. We show that for triangle-free graphs, IG solves CCP optimally in expected polynomial time. Secondly, we show that IG finds the optimum for CCP in a specific case of sparse random graphs in expected polynomial time with high probability. For Barabási-Albert model of scale-free networks, which is a canonical model explaining the growth of social, biological or computer networks, we obtain that IG obtains an asymptotically optimal approximation in polynomial time in expectation. Last but not least, we propose a slightly modified variant of IG, which guarantees expected polynomial-time convergence to the optimum for graphs with non-overlapping triangles.

Keywords: Vertex clique covering problem, iterated greedy, randomised search heuristics, complex networks, random graphs

Mathematics Subject Classification 2010: 68R10, 05C69, 05C70

1 INTRODUCTION

This paper is dedicated to the analytical study of an *iterated greedy* (IG) heuristic for the vertex *clique covering problem* (CCP) in several practically relevant classes of graphs, including triangle-free graphs, sparse random graphs and models of *complex networks*. These networks include social networks [21, 35], biological networks [12], research citation and collaboration networks [21, 34], language networks [29] or the Internet [4]. Both methods and software tools for exploration of complex networks are developed [14].

The problem we study in this work is closely related to the popular areas of community detection [28], graph clustering [34] and graph mining [9]. The aim of CCP is to partition the vertices into as few pairwise disjoint subsets as possible such that each subset induces a clique. In the context of social networks, CCP is a problem of "strict" community detection, in which the vertices are partitioned into the minimum number of groups so that everybody knows each other within each group.

Definition 1 (Definition of CCP)**.** Let $G = [V, E]$ be an undirected graph on n vertices and m edges. Let $d(G) = \frac{2m}{n(n-1)}$ be its density, with $d(G) = 1$ if $0 \leq n \leq 1$. The objective of CCP is to minimise $k \leq n$ such that there are classes $V_1, V_2, \ldots, V_k$, satisfying the following constraints:

1. each vertex is in exactly one class, i.e.

$$\forall\ i, j = 1..k, i \neq j : V_i \cap V_j = \emptyset, \quad \bigcup_{i=1}^{k} V_i = V,$$

2. each class induces a clique, i.e.

$$\forall i = 1..k : d(G(V_i)) = 1,$$

where $G(V_i) = [V_i, E(V_i)]$ is a *subgraph induced by* V_i, containing only edges between vertices of class V_i. The minimum value of k for which there is a clique covering will be referred to as the *clique covering number* and denoted by $\vartheta(G)$ [10].

CCP is one of the classical NP-hard problems [24]. It corresponds to graph colouring of the complementary graph, which perhaps explains why the current literature mostly overlooks this problem and focuses more on graph colouring. The relationship between CCP and graph colouring influences the approximation results on CCP. To the best of our knowledge, the best general approximation algorithm for CCP is the one for graph colouring, which achieves approximation ratio $\mathcal{O}(n(\log\log n)^2/(\log n)^3)$ [23]. However, better approximation ratio may be obtained or the problem may be solved in polynomial time for restricted graph classes [8].

We note that the similar edge clique covering problem (ECCP) is also studied and is NP-hard, too. For ECCP, more studies seem to be currently published, especially for specific classes of graphs [5, 22, 25].

Iterated greedy (IG) algorithm was previously demonstrated to provide encouraging empirical results for CCP in real-world networks [11]. IG is a heuristic algorithm, which utilises the block-based properties of CCP to find high-quality solutions efficiently. In this context, it is closely related to evolutionary algorithms, as well as randomised search heuristics [3].

Even though IG does not guarantee that the best solution is always found, it usually performs well in practice. It is able to find optimal or near-optimal solutions for social and research collaboration networks [11], as well as protein-protein interaction networks [12]. In addition, IG does not use any prior knowledge of a specific graph class to make the optimisation more efficient. Therefore, even though more suitable algorithms can be found for specific families of graphs, the aim of this paper is to explore the capabilities of a more general approach. Similarly to the research on other randomised search heuristics [32], we obtain that IG mimics the behaviour of classical algorithms to some extent, provably finding optimal or asymptotically optimal solutions in polynomial time for several practically relevant graph classes.

1.1 Contributions

It was previously shown that IG finds the optimal solution for paths in polynomial time [10]. We extend this result by first showing that the behaviour of IG for triangle-free graphs can be modelled using random walks and we prove that the optimal solution is found in expected $\mathcal{O}(n^5m^2)$ time. This bound is based on rather pessimistic assumptions. IG seems to be much faster in practice.

Next, we show that these arguments can be generalised to sparse random graphs generated according to the Erdős-Rényi model [16] $\mathbb{G}(n, c/n)$, i.e. graphs on n vertices with randomly generated edge with probability c/n for each pair of vertices. We show that for graphs generated with $c < 1$, IG will find the optimal clique covering in expected $\mathcal{O}(n^3(\log n)^5)$ time with probability $1 - o(1)$.

As a next step, we study the behaviour of IG for the Barabási-Albert (BA) model of scale-free networks, which is a canonical model explaining the growth of social and other complex networks [4]. We obtain that IG achieves approximation ratio $1+\mathcal{O}\left(\frac{(\log n)^3}{n}\right)$ for graphs generated by BA model in expected polynomial time. This approximation ratio is asymptotically optimal.

Last but not least, we show that even though IG can fail to provide the optimum for graphs with non-overlapping triangles with probability $1 - o(1)$ [10], this drawback can be overcome by putting the triangles as blocks in the initial solution. Such modification leads to an algorithm, which finds the optimum in expected polynomial time.

Even though most of these results are not particularly surprising, our analysis introduces several insights into the behaviour of heuristics, which combine a classical greedy approach with randomised search. It confirms that IG can be viewed as a randomised local search algorithm, with its behaviour modelled using methods of analysis of evolutionary algorithms. This includes the fitness levels method [27, 32, 36], as well as methods modelling the optimisation process as a random walk [2].

The rest of the paper is structured as follows. In Section 2, we briefly review the background of CCP, IG algorithm and related work. In Section 3, we show that IG finds the optimal solution for triangle-free graphs in expected polynomial time. In Section 4, we show that IG finds the optimal solution for the specific case of sparse random graphs in expected polynomial time with high probability. In Section 5, we consider the impact of triangles upon our problem. In Section 6, we show that IG achieves asymptotically optimal approximation ratio for graphs generated by BA model in expected polynomial time. In Section 7, we show how to extend IG so that it guarantees that the optimum is found for graphs with non-overlapping triangles. In Section 8, we give conclusions and summarise the current open problems.

2 ITERATED GREEDY CLIQUE COVERING

IG is a randomised search heuristic, i.e., it does not guarantee that optimal solution is found but it might provide very good results for certain types of problem instances. IG was previously successfully used to solve graph colouring [13], train scheduling [42] or flowshop scheduling problem [33].

Over the last years, analysis of randomised search heuristics in combinatorial optimisation problems has become a very active research area [3, 32]. Problems, for which results have been published, include polynomial-time solvable problems such as the maximum matching problem [20], Eulerian cycle problem [30] or minimum spanning tree problem [31]. However, NP-hard problems are also often considered, including the vertex cover problem [18, 26, 41], Euclidean travelling salesperson problem [38] or the graph colouring problem [37].

At this point, we move on to the description of our IG algorithm for CCP. The roots of this algorithm date back to the work by Culberson and Luo [13], who used a similar approach to solve the graph colouring problem. Inspired by this work, we have relatively recently developed an IG algorithm for CCP. Interestingly, our previous results indicated that IG has all the features of typical *local search.* For paths, IG converges to the optimum in polynomial time, for complements of bipartite graphs, it can get stuck in local optima and there are also specific graph classes, where IG will get stuck in local optima almost certainly [10].

However, the current theoretical results for IG are still relatively distant from its main application in social and other complex networks. In our previous empirical study, IG was able to find optimal solutions for real-world graphs in many cases, while in the rest of the cases, the obtained solutions were very close to the optimum [11]. Therefore, further analytical results for IG are of a high interest.

2.1 Description of IG

Our IG algorithm uses *greedy clique covering* (GCC) [10]. GCC begins with an empty clique covering. Technically, the cliques are marked with labels, similarly to the graph colouring problem. GCC takes the vertices in an order determined by input permutation P. In each iteration, it puts a vertex into the first clique (i.e. with the lowest index of its label) such that the clique property is not violated. If this is not possible, a new label is used, leading to a new clique being created. This way, a solution is iteratively constructed. We will refer to the choice of the first clique as the *First Fit rule* [39]. Efficient implementation techniques are available for GCC to run in $\mathcal{O}(m)$ time, where m is the number of edges in the graph. This makes the algorithm particularly suitable for large but sparse networks. For more detailed information on GCC, the reader may refer to the previous work [10].

```
1   begin with an uniformly random permutation $P$
2   repeat until convergence
3       construct solution $[V_1, V_2, \ldots, V_k]$ with greedy clique covering for $P$
4       let $P = [V_1, V_2, \ldots, V_k]$ so that $V_1$, $V_2$, $\ldots$, $V_k$ form blocks in $P$
5       perform *block_jump* for a uniformly randomly chosen block
        from $V_1, V_2, \ldots, V_k$ to create new $P$
```

Algorithm 1. Iterated greedy (IG) clique covering

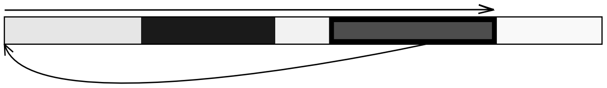


Figure 1. Illustration of the *block_jump* operator, which was introduced as a canonical block-based operator for IG [10]. Operator *block_jump* takes a chosen block representing a clique and puts it to the first position in the permutation. The other blocks are then shifted to the right.

The pseudocode of IG is given in Algorithm 1. First, GCC is used with a uniformly random initial permutation of vertices to construct the initial clique covering. Then, IG groups vertices of the identified cliques into blocks, as shown in Figure 1. One of these blocks is then taken uniformly at random and is put to the first position in the permutation. The other blocks are then shifted to the right. This operation will be further referred to as *block_jump*. GCC is used once again with the resulting permutation to construct clique covering for the next iteration. This new clique covering will never consist of more cliques than the previous one, because of the greedy nature of GCC and the fact that cliques of the previous solution form blocks. The process is repeated until a stopping criterion is met. In this paper, we will investigate the time until IG finds the optimal solution for specific graph classes.

3 RESULT FOR TRIANGLE-FREE GRAPHS

In this section, we move on to our analysis. Although IG is not a typical evolutionary algorithm, it is a closely related method. Therefore, we will use the methods of runtime analysis for evolutionary algorithms, which have been demonstrated as suitable for runtime analysis of IG.

We build our results on a relatively widely used method of *fitness levels* [27, 32, 36]. We divide the search space into levels such that each level contains all solutions with the same number of cliques. Then, Lemma 1 can be used to find an upper bound for the expected running time of our algorithm.

Lemma 1 ([32]). The expected optimization time I of a stochastic search algorithm that works at each time step with a population of size 1 and produces at each time step a new solution from the current solution is upper bounded by:

$$I \leq \sum_{i=1}^{m-1} \frac{1}{p_i}. \tag{1}$$

In Lemma 1, m represents the number of fitness levels and p_i is the minimum probability that in time step i, the stochastic change will cause an improvement. In Lemma 2, we recall the previous result on the quality of initial solution for IG for paths. This result will be used in our next discussions, since paths are a special case of triangle-free graphs.

Lemma 2 ([10]). For paths, the initial solution for IG can contain at most $\lceil 2/3n \rceil$ cliques and there are at most $\lceil n/3 \rceil$ 1-cliques in the result.

We now show that in expectation, IG finds the optimal vertex clique covering in polynomial time for *triangle-free graphs*. Even though CCP can be solved in polynomial time for triangle-free graphs using maximum matching [8], and the simple (1+1) evolutionary algorithm has previously been shown to be a polynomial-time randomised approximation scheme for maximum matching [20], it is interesting to investigate the behaviour of a more general randomised search heuristic for CCP. We will see that IG is able to guarantee polynomial-time convergence to the optimum in expectation. Additionally, analysis for triangle-free graphs represents a step towards analysis for random graphs, as well as complex networks.

It is worth noting that our bound is very pessimistic, due to assumptions used to make the proof simpler. IG seems to be much faster in most practical scenarios.

As a consequence of this result, we also have that IG solves CCP optimally in polynomial time for *trees* and *bipartite graphs* in general. This will be an extension of our previous result of IG for paths, for which IG behaves similarly [10]. However, in contrast to paths, general triangle-free graphs do not have a bounded maximum degree. This makes the random walks, which arise in the analysis of IG, to be slightly more complex than simply "left versus right". Hence, several new ideas will be introduced in the following analysis.

Theorem 1. For triangle-free graphs on n vertices and m edges, the expected time for IG to find the optimal vertex clique covering is upper bounded by $\mathcal{O}(n^5m^2)$.

Proof. Based on Lemma 1, the initial clique covering contains $\mathcal{O}(n)$ more cliques than the optimum. These will determine our fitness levels.

The size of the maximum clique $\omega \leq 2$, since we have a triangle-free graph. Cliques of size one will be called 1-*cliques* and two-vertex cliques will be called 2-*cliques.* An improvement occurs if random changes cause 1-cliques move so that some pair of 1-cliques are next to each other and form a 2-clique.

Suppose that we have a fitness level with $\vartheta + d$ cliques, where $d \geq 1$. Then, the number of 1-cliques is at least $2d \geq 2$. We will now show that 1-cliques perform a fair random walk [32] on the triangle-free graph.

We first look at what happens if *block_jump* occurs. If *block_jump* is applied to a 2-clique, only the ordering of the 2-cliques can be changed. No vertex can be taken by a 2-clique, since that would create a triangle. If *block_jump* is applied to a 1-clique, the 1-clique will form a 2-clique with its nearest following neighbour in the permutation. This is due the First Fit rule, which was mentioned in Section 2.

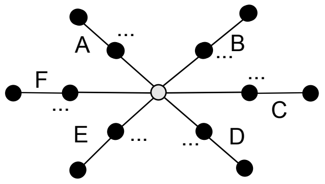


Figure 2. An illustration of the situation with a 1-clique between several 2-cliques. This picture should be perceived as a subgraph, other subgraphs can be attached to the black vertices. The direction, where the 1-clique moves when *block_jump* is applied on it depends on which of the blocks A–F comes first in the permutation.

In Figure 2, we illustrate the situation, when a 1-clique was left between several 2-cliques. Each 1-clique must necessarily have only 2-cliques around it. If it did not have, it would be joined with another 1-clique in a 2-clique.

Let X now be the waiting time until a *block_jump* of this 1-clique occurs. Before the final move of the 1-clique, there are $X - 1$ *block_jump* operations.

The direction of the movement of this 1-clique is determined by which neighbour (in Figure 2, determined by blocks A–F) comes first in the permutation. The probabilities for the directions will depend on what happens during the waiting time. More particularly, which of the blocks around the 1-clique was taken for *block_jump* as the last one. To make the proof simpler, we will pessimistically assume that the *block_jump* operations performed on the other 1-cliques during the waiting time did not lead to an improvement. Now, we will have two cases, what can happen during this waiting time.

Case 1. None of the blocks around the 1-clique jumped. Let X be the waiting time. We have that $X = 1$ (i.e., it takes only one move to choose the 1-clique) with probability $1/(\vartheta + d) \leq 2/n$. In this case, no other moves could surely be chosen. For $X > 1$, we observe that for our 1-clique vertex v with $\deg(v)$ neighbours, the probability of this event will be:

$$\left(1 - \frac{\deg(v)}{X-1}\right)^{X-1} = \left(1 - \frac{\deg(v)}{X-1}\right)^{\frac{X-1}{\deg(v)}\deg(v)} \leq e^{-\deg(v)}. \qquad (2)$$

This is because in all $X - 1$ steps in the waiting time, only non-neighbour blocks were taken. Thus, the direction of movement for our 1-clique stays the same. Therefore, this case occurs with probability, which is upper bounded by $e^{-\deg(v)} + 2/n$.

Case 2. Some block around the 1-clique jumped. We are interested in which of the $\deg(v)$ blocks was the last to jump. The probability of this case is at least $1 - e^{-\deg(v)} - 2/n$, because of the bound shown in Case 1. We will now argue that this portion of probability is distributed fairly among all $\deg(v)$ neighbour blocks. This is because the probability of *block_jump* is uniformly distributed among the blocks. Thus, for each situation, where A was the last to jump, there are equally probable situations, where the last *block_jump* was performed on B, C, etc.

Hence, the probability of changing the direction of movement of 1-clique is at least $(1 - e^{-\deg(v)} - 2/n)/\deg(v)$ for each neighbour block.

During the waiting time, the solution can be changed a lot. However, if considering the neighbours of our 1-clique only, then only 2-cliques must be around it during the whole waiting time. Otherwise, an improvement would be achieved, which is a possibility that we pessimistically exclude.

Let us now consider the event outlined in Case 1. None of the blocks around the 1-clique jumped, i.e., the direction will be determined by the block, which is currently the first in the permutation. Based on the previous arguments, the probability that one fixed neighbour block (in Figure 2, one of the blocks A–F) was the first one in the beginning of the waiting time, is uniformly distributed, too. This is implied by the fact that the initial permutation is uniformly random, and the probability of *block_jump* is also uniformly distributed among the blocks. Therefore, 1-cliques actually perform fair random walks on the triangle-free graph.

From the cover time of random walks, it takes $\mathcal{O}(nm)$ *block_jump* moves of a 1-clique to visit each vertex at least once [2]. For two such random walks, we have that it takes $\mathcal{O}(n^2m^2)$ *block_jump* moves in expectation for two 1-cliques to arrive at two adjacent vertices. $\mathcal{O}(n)$ is the time needed to obtain a *block_jump* of the 1-clique and $\mathcal{O}(n)$ is the complexity of GCC.

We have $\mathcal{O}(n)$ fitness levels, on which all this happens. Therefore, the expected time to obtain the optimum is bounded by $\mathcal{O}(n^5m^2)$. □

4 RESULT FOR SPARSE RANDOM GRAPHS

We have shown that IG finds the optimum in polynomial time for triangle-free graphs. At this point, we extend this result by studying sparse random graphs, generated by the well-known Erdős-Rényi model [16]. Consider the model in the form $\mathbb{G}(n, c/n)$, generating graphs on n vertices such that an edge is put between each pair of vertices independently with probability c/n. This model has an interesting property that for $c < 1$, the graph will consist of small components with specific properties with high probability. These properties have previously been used to prove results for iterated local search algorithms for vertex cover [41] and graph colouring [37].

Theorem 2. Let $0 < c < 1$. Then, for an Erdős-Rényi random graph G from $\mathbb{G}(n, c/n)$, the expected time for IG to find an optimal clique covering for G is upper bounded by $\mathcal{O}(n^3(\log n)^5)$ with probability $1 - o(1)$.

Proof. Bollobás [6], Sudholt and Zarges [37], and Witt [41] state that, with probability $1 - o(1)$, a random graph G from $\mathbb{G}(n, c/n)$, $0 < c < 1$, will consist of components on $\mathcal{O}(\log n)$ vertices and edges, which are trees or graphs with at most one cycle.

For a tree, or a graph with cycle with at least 4 vertices, we have that the component is triangle-free, i.e., the arguments from Theorem 1 can be applied directly. The remaining case is a component with a single triangle. We first prove that for such a component, each suboptimal solution contains at least two 1-cliques. We use enumeration based on whether the optimal/suboptimal solutions contain the triangle.

Case 1. The optimum does not contain the triangle. Hence, the optimum contains only 2-cliques and 1-cliques, i.e., overestimation can occur only by using two 1-cliques instead of a 2-clique.

Case 2. The optimum contains the triangle. If the suboptimum also contains the triangle, we have the same situation as in Case 1, since overestimation can occur only by using two 1-cliques instead of a 2-clique. Suppose that the suboptimum does not contain the triangle and it does not contain a 1-clique, too. Thus, it can only contain 2-cliques. However, such a solution cannot be improved, since a substitution of two of its 2-cliques by a triangle would leave the fourth vertex for a 1-clique. Therefore, such a solution must be the optimum.

This proves that each suboptimum contains at least two 1-cliques. We now analyse the expected time to obtain a situation when the two 1-cliques visit a configuration, in which they form a 2-clique.

If the 1-cliques are in the same subtree of the component, they need to visit $\mathcal{O}((\log n)^4)$ vertices to visit adjacent vertices simultaneously, and form a 2-clique. This is implied by the fact that a component contains $\mathcal{O}(\log n)$ vertices.

When the two 1-cliques are in different subtrees, we must explore the expected time needed for them to visit vertices of the triangle simultaneously. If we assume that events in both subtrees do not lead to an improvement, we can treat them as independent. Therefore, we have that 1-cliques need to visit $\mathcal{O}((\log n)^4)$ vertices to arrive at the triangle at the same time.

Expected waiting time for a *block_jump* of a 1-clique is $\mathcal{O}(n)$. GCC has complexity $\mathcal{O}(n \log n)$ in the worst case, since we have at most n components with $\mathcal{O}(\log n)$ edges. An improvement is obtained when $\mathcal{O}((\log n)^4)$ vertex pairs are visited by two 1-cliques in a component in expectation. Expected waiting time until an improvement to a better fitness level is therefore upper bounded by $\mathcal{O}(n^2(\log n)^5)$. We have $\mathcal{O}(n)$ fitness levels, which proves our theorem. □

5 ON THE IMPACT OF TRIANGLES

Up to this point, the analysis was only taking graphs into consideration with at most one triangle per connected component. Lemma 3 summarises the negative result for a graph with linear number of non-overlapping triangles. For graph $H_{\vartheta/2}$ depicted in Figure 3, IG will get stuck in a suboptimal clique covering with probability $1 - o(1)$.

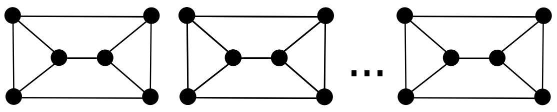


Figure 3. An illustration of the graph $H_{\vartheta/2}$, consisting of $\vartheta/2 = n/6$ connected components, for which IG does not produce the optimal vertex clique covering with probability $1 - o(1)$. This is due to the fact that if two horizontal edges are selected instead of the two triangles in at least one of the components, *block_jump* will not be able to suitably regroup the vertices [10].

Lemma 3 ([10]). For graph $H_{\vartheta/2}$, IG will not be able to produce the optimal vertex clique covering with probability $1 - o(1)$.

However, for graphs with a limited number of triangles, IG may achieve a good approximation of the optimum in polynomial time. In Lemma 4, we recall a lower bound for ϑ_n based on maximum independent set size α_n and maximum clique size ω_n. Consequently, Theorem 3 formulates the main approximation result.

Lemma 4 ([11])**.** Let α_n and ω_n be the sizes of maximum independent set and maximum clique for a class of graphs on n vertices, respectively. Then, it holds that $\max\{\alpha_n, n/\omega_n\} \leq \vartheta_n$.

Theorem 3. Let G be a graph on n vertices with τ_n triangles such that $\tau_n < n/3$. Then, IG will achieve approximation ratio:

$$1 + 6\frac{\tau_n}{n - 3\tau_n} \tag{3}$$

for G in expected polynomial time.

Proof. Let $V_T \subseteq V$ be the set of vertices in G, which are in at least one triangle. Based on the premises, we have that $|V_T| \leq 3\tau_n$. Let G_{TF} be the subgraph induced by $V \backslash V_T$, i.e. the triangle-free subgraph, which excludes the vertices in V_T and their incident edges.

For the triangle-free subgraph G_{TF}, we have that the situation around each 1-clique can be modelled using the analysis illustrated in Figure 2. Therefore, the fair random walk argument remains valid for the triangle-free "segments" between triangles.

Let $\vartheta'_n(G)$ be the number of cliques used by IG when triangle-free subgraphs are already covered optimally after $\mathcal{O}(n^5m^2)$ time in expectation, based on the arguments of Theorem 1, and let $\vartheta_n(G_{TF})$ be the clique covering number of the triangle-free subgraph G_{TF}. For the number of cliques used by IG, we have that $\vartheta'_n(G) \leq \vartheta_n(G_{TF}) + 3\tau_n$, since G_{TF} is covered optimally. The clique covering number $\vartheta_n(G)$ satisfies $\vartheta_n(G) \geq \vartheta_n(G_{TF})$. Therefore, the achieved approximation ratio is upper bounded by:

$$\frac{\vartheta_n(G_{TF}) + 3\tau_n}{\vartheta_n(G_{TF})} = 1 + 3\frac{\tau_n}{\vartheta_n(G_{TF})} \leq 1 + 3\frac{\tau_n}{(n - 3\tau_n)/2} = 1 + 6\frac{\tau_n}{n - 3\tau_n} \tag{4}$$

where the fact that $(n - 3\tau_n)/2 \leq \vartheta_n(G_{TF})$ is implied by Lemma 4 and $\omega(G_{TF}) = 2$, since G_{TF} is triangle-free. □

6 RESULT FOR BARABÁSI-ALBERT MODEL OF SCALE-FREE NETWORKS

At this point, we relate the previous result to models of real-world complex networks. Complex networks are networks with non-trivial structure. This structure is closely related to the process of their evolution. Complex networks are often statistically characterised by their degree distribution $P(k)$, which denotes the fraction of vertices, which have degree k. Many real-world networks are believed to be *scale-free*, which means that their degree distribution follows the power law, i.e. $P(k) \sim ck^{-\gamma}$, where γ is a coefficient of steepness of the distribution and c is a suitable constant.

Therefore, scale-free networks contain many vertices with low degree but also several vertices with very high degree. In real-world networks, it usually holds that $\gamma \in [2, 3]$ [1].

One of the most famous models used to explain the process of evolution of scale-free networks is the *Barabási-Albert (BA) model* [4]. Its pseudocode is given in Algorithm 2.

```
1  begin with a connected seed graph $G_0 = [V_0, E_0]$
2  for $t = (n_0 + 1) \dots n$
3      $V_t = V_{t-1} \cup \{v_t\}$
4      attach $v_t$ to vertices from $V_{t-1}$ based on preferential attachment rule
```

Algorithm 2. Barabási-Albert (BA) model of scale-free networks [4]

In BA model, we begin with a connected seed graph on n_0 vertices and m_0 edges. Then, at each time step t, one new vertex comes and brings w new edges to the network, where w is a parameter of the model, which remains constant over time. These edges are attached to the existing vertices preferentially, i.e., the probability of attachment to vertex v is $\frac{(\deg(v))_t}{2m_t}$, where $(\deg(v))_t$ is the degree of v in time step t and m_t is the number of all edges at this time step. In the context of social networks, this can be interpreted in the way that a person with a larger number of contacts is more likely to get a new contact. It is known that BA model generates networks with degree distributions, which follow the power law in form $P(k) \sim ck^{-3}$, i.e. $\gamma = 3$ [4].

Lemma 5. In BA model with w incoming edges per vertex and with a seed graph with maximum clique size at most $w + 1$, the maximum clique number ω_n satisfies $\omega_n \leq w + 1$ for any n.

Proof. We prove this by contradiction. Suppose that $\omega_n > w + 1$. Then, the last vertex of the maximum clique must have been attached to at least $w + 1$ other vertices. This contradicts the fact that we have w incoming edges per vertex. □

Lemma 6. Suppose that the seed graph for BA model is a tree. If $w = 1$, then the resulting graph will also be a tree.

Proof. From Lemma 5, we have that the maximum clique number $\omega \leq 2$, i.e., it will be triangle-free. For generation of a cycle, one would have to have at least two incoming edges for the last vertex, which "closes" the cycle. Hence, the resulting graph will be connected and acyclic, i.e., it will be a tree. □

Corollary 1. Let G be a graph on n vertices generated by BA model with 1 incoming edge per vertex and with a tree as a seed graph. Then, IG finds the optimal clique covering for G in polynomial time.

The previous results are relatively straightforward. It is more interesting to see how good solution IG produces for BA model with $w \geq 2$. We first recall a classical result on the number of triangles in BA model in Lemma 7 and Theorem 4 applies it to show that the approximation achieved by IG is asymptotically optimal.

Lemma 7 ([7])**.** Let $w \geq 1$ be fixed. The expected number of triangles in a graph on n vertices generated by BA model with w incoming edges per vertex is given by:

$$(1 + o(1))\frac{w(w-1)(w+1)}{48}(\log n)^3 \tag{5}$$

as $n \to \infty$.

Theorem 4. Let G be a graph on n vertices generated by BA model with a triangle-free seed graph and an arbitrary number w of incoming edges per vertex. Then, IG achieves approximation ratio $1 + \mathcal{O}\left(\frac{(\log n)^3}{n}\right)$ for G in expected polynomial time.

Proof. Based on Lemma 7, we have that the number of triangles $\tau_n = \mathcal{O}((\log n)^3)$. The triangle-free seed graph assures that this upper bound also holds for small n. Theorem 3 implies that IG achieves approximation ratio:

$$1 + 6\frac{\mathcal{O}((\log n)^3)}{n - \mathcal{O}((\log n)^3)} = 1 + \mathcal{O}\left(\frac{(\log n)^3}{n}\right) \tag{6}$$

in expected polynomial time. □

It is worth mentioning that this result is similar to the result of evolutionary algorithms in the NP-hard makespan scheduling problem, where asymptotically vanishing discrepancies in the obtained solutions were proven [40]. However, Theorem 3 cannot be applied to graphs with linear or superlinear numbers of triangles, which may be encountered in other network models [15]. In the next section, we investigate the impact of non-overlapping triangles on the design of a suitable algorithm for CCP.

7 RESULT FOR GRAPHS WITH NON-OVERLAPPING TRIANGLES

The previous results were mostly positive. However, Lemma 3 has also outlined the limitations of IG. At this point, we investigate the behaviour of IG for graphs with non-overlapping triangles. Consider the initial permutation being generated such that non-overlapping triangles are placed into it as blocks. The rest of the permutation is generated uniformly at random. In the following, we show that such a modification of IG guarantees that the optimal clique covering is found in expected polynomial time.

Lemma 8. Let G be a graph with maximum clique size $\omega = 3$. Let S be an optimum and let S' be a suboptimum for CCP in G. There are three cases of how IG can overestimate $\vartheta(G)$:

Case 1. Instead of a 2-clique in S, there are two 1-cliques in S',

Case 2. Instead of a triangle in S, there is a 2-clique and a 1-clique in S',

Case 3. Instead of two triangles in S, there are three 2-cliques in S'.

All possible ways of overestimation represented by S' represent compositions of these three cases.

Proof. Graphs and clique coverings generated by GCC, where the first two cases can occur, are common and can be found very easily. The existence of the third case is proven by Lemma 3. To exclude the existence of other ways, we use simple enumeration.

- Substitution of any number of 2-cliques by 1-cliques is a composition of events included in Case 1.
- Substitution of one triangle by three 1-cliques is a composition of Case 1 and Case 2.
- If we consider three triangles, the first two can be substituted based on Case 2 or Case 3 and the last triangle will remain for Case 2.
- If we consider four or more triangles, we can apply Case 2 and Case 3 iteratively. The resulting 2-cliques are further divided according to Case 1.

□

Lemma 9. Let G be a graph with maximum clique size $\omega = 3$. If there is a suboptimal clique covering S' of G, which contains more triangles than an optimum S, then S' must also contain at least two 1-cliques.

Proof. Let c_1, c_2 and c_3 be the numbers of cliques in S with 1, 2 or 3 vertices, respectively. Let c_1', c_2' and $c_3' = c_3 + d$ be the respective values for S, and for $d \geq 1$. All vertices must be covered and S must contain less cliques than S'. Hence:

$$c_1 + 2c_2 + 3c_3 = c_1' + 2c_2' + 3(c_3 + d) = n, \tag{7}$$

$$c_1 + c_2 + c_3 < c_1' + c_2' + c_3 + d.$$

From these formulas, $3d = (c_1 - c_1') + 2(c_2 - c_2')$ and $d > (c_1 - c_1') + (c_2 - c_2')$. This implies that $(c_2 - c_2') > 2d$ and, thus, $(c_1 - c_1') < -d$. The value $-d$ can be at most -1, i.e. $c_1' > 1$. □

We now split the number of 1-cliques into two values. Let an optimum S contain c_1 1-cliques. We will call this the number of *free* 1-*cliques*. If a suboptimum S' contains c_1' 1-cliques, then $(c_1' - c_1)$ is the number of *extra* 1-*cliques*. The idea now is to model the process as the minimisation of the number of extra 1-cliques, rather than the number of all cliques in the covering.

Lemma 10. Let G be a graph with maximum clique size $\omega = 3$. Let IG begin with a suboptimal clique covering S' with at least as many triangles as in an optimal clique covering S for G. Then, IG cannot get stuck in a local optimum and will be in the global optimum if the number of extra 1-cliques in the solution is minimal.

Proof. Let S contain c_1 1-cliques, c_2 2-cliques and c_3 triangles. The analogous values for S' are c'_1, c'_2 and $c'_3 \geq c_3$. The premises imply that an improvement cannot be obtained by making the number of triangles higher. Therefore, it is necessary that $c'_2 < c_2$ and $c'_1 > c_1$. Since c_1 1-cliques are present is S, the only way to obtain an improvement is to reduce the number of extra 1-cliques by 2 and increase the number of 2-cliques by 1. This holds for all suboptima, which proves the second statement.

For the first statement, suppose that IG got stuck. Then, by Lemma 8, two of the triangles must have been substituted by three 2-cliques. However, this is in contradiction with the fact that these three 2-cliques must lie between the triangles and *block_jump* cannot cause a transformation, in which vertices between 2 different blocks are regrouped to 3 blocks in between. □

Theorem 5. Let G be a graph on n vertices with maximum clique size $\omega = 3$, containing only non-overlapping triangles. Let P be the initial permutation for IG, constructed by placing the triangles into P as blocks first and the rest of vertices are placed into P uniformly at random. Then, IG will find the optimal solution in $\mathcal{O}(n^5 m^2)$ time in expectation.

Proof. Based on Lemma 9, the initial solution must be a global optimum or it contains a 1-clique. Suppose that it is a suboptimum. Lemma 10 implies that the following process is a minimisation of the number of extra 1-cliques and getting stuck in local optima is avoided.

In each time step, we have a situation, in which a 1-clique is stuck between 2-cliques and triangles, similarly to Figure 2. The probability of moving towards each direction is naturally determined by which block comes first. This is not influenced by the fact that we can have triangles. We have to examine two cases, depicted by Figure 4.

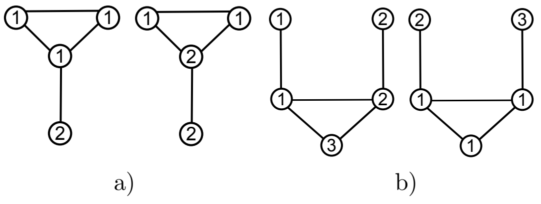


Figure 4. Illustration of the cases for the non-overlapping triangles for the proof of Theorem 5. Case a) represents a 1-clique, which can freely emerge both in suboptima and optima. Case b) illustrates a situation, when 1-clique performs a random walk by "jumping" over the triangle.

Case 1. If the 1-clique is not in a triangle, it can be surrounded by 2-cliques or triangles, to which it can be connected only by a single edge (since it is in no triangle). The 2-clique case is handled by the arguments from Theorem 1. In

Figure 4 a), we depict the situation, when it is adjacent to a vertex in a triangle block. Such a 1-clique can be freely enhanced to a 2-clique and reduced back to 1-clique afterwards. Such a transformation can occur between two optima, which shows that such a clique does not contribute to the number of extra 1-cliques.

Case 2. In this case, the 1-clique is in a single triangle. In this case, the other vertices of the triangle must be separated into different cliques. Since they cannot be in a triangle, they must each be in its own 2-clique, as shown by Figure 4 b). When *block_jump* is applied to the 1-clique, this 1-clique is transformed into the triangle, and a new 1-clique can emerge on the opposite side of the triangle. This position depends on the ordering of cliques on the other side, for which the probability is uniformly distributed, leading to validity of the fair random walk argument.

In each suboptimal solution, we have that the number of extra 1-cliques is at least 2. Therefore, by applying the same cover time arguments as in Theorem 1, we have that the expected time to obtain the optimum is upper bounded by $\mathcal{O}(n^5 m^2)$. □

Even though the assumption of non-overlapping triangles is still strong, it gives us some insight into the impact of triangles on the problem structure and design of suitable algorithms. We hope that these results may pave the way to more sophisticated analyses of heuristics for CCP, as well as other combinatorial optimisation problems for different models of complex networks and practically relevant scenarios.

8 CONCLUSIONS

We presented an analysis of an *iterated greedy* (IG) heuristic for the vertex *clique covering problem* (CCP) in several practically relevant graph classes. As our analytical results indicate, IG can be viewed as a variant of local search, with non-trivial methods needed to quantify the convergence and runtime properties of this randomised search heuristic.

The classes of graphs concerned include *triangle-free graphs*, *sparse random graphs*, scale-free networks generated by *Barabási-Albert (BA) model*, and *graphs with non-overlapping triangles.*

We have shown that for triangle-free graphs, IG finds the optimum in expected polynomial time. For sparse random graphs generated by the *Erdős-Rényi model* in its form $\mathbb{G}(n, c/n)$, where c/n is the probability of edge generation, we have shown that IG finds the optimum in expected polynomial time with high probability if $c < 1$.

For BA model, we have shown that IG achieves approximation ratio $1 + \mathcal{O}\left(\frac{(\log n)^3}{n}\right)$ in expected polynomial time.

Last but not least, we have shown that for graphs with non-overlapping triangles, putting the triangles in the initial permutation for IG as blocks helps to improve the worst-case performance of IG from getting stuck with probability $1 - o(1)$ to finding the optimum in expected polynomial time.

We believe that these results provide a valuable insight into the behaviour of heuristics, which combine ideas of classical greedy algorithms with randomised iterative improvement processes. This insight may represent a foundation of analysis for other graph classes, as well as for other problems such as graph colouring [13, 37], independent sets [11], or other similar algorithms such as the greedy randomised adaptive search procedures (GRASP) [17].

Acknowledgement

This contribution was supported by Grant Agency VEGA SR under the grant 1/0145/18.

REFERENCES

[1] Albert, R.—Barabási, A.-L.: Statistical Mechanics of Complex Networks. Reviews of Modern Physics, Vol. 74, 2002, No. 1, pp. 47–97, doi: 10.1103/RevModPhys.74.47.

[2] Aleliunas, R.—Karp, R. M.—Lipton, R. J.—Lovasz, L.—Rackoff, C.: Random Walks, Universal Traversal Sequences, and the Complexity of Maze Problems. Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS '79), 1979, pp. 218–223, doi: 10.1109/SFCS.1979.34.

[3] Auger, A.—Doerr, B. (Eds.): Theory of Randomized Search Heuristics. World Scientific, Series on Theoretical Computer Science, Vol. 1, 2011.

[4] Barabási, A.-L.—Albert, R.: Emergence of Scaling in Random Networks. Science, Vol. 286, 1999, No. 5439, pp. 509–512, doi: 10.1126/science.286.5439.509.

[5] Behrisch, M.—Taraz, A.: Efficiently Covering Complex Networks with Cliques of Similar Vertices. Theoretical Computer Science, Vol. 355, 2006, No. 1, pp. 37–47, doi: 10.1016/j.tcs.2005.12.005.

[6] Bollobás, B.: Random Graphs. Cambridge University Press, Cambridge, 2001, doi: 10.1017/CBO9780511814068.

[7] Bollobás, B.—Riordan, O. M.: Mathematical Results on Scale-Free Random Graphs. In: Bornholdt, S., Schuster, H. G. (Eds.): Handbook of Graphs and Networks. Wiley, 2005, pp. 1–34.

[8] Cacetta, L.—Purwanto, P.: Deficiencies and Vertex Clique Covering Numbers of a Family of Trees. Australasian Journal of Combinatorics, Vol. 1, 1990, pp. 15–27.

[9] Chakrabarti, D.—Faloutsos, C.: Graph Mining: Laws, Generators, and Algorithms. ACM Computing Surveys, Vol. 38, 2006, No. 1, Article No. 2.

[10] Chalupa, D.: An Analytical Investigation of Block-Based Mutation Operators for Order-Based Stochastic Clique Covering Algorithms. In: Blum, C., Alba, E. (Eds.): Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation (GECCO '13). ACM, 2013, pp. 495–502, doi: 10.1145/2463372.2463436.

[11] Chalupa, D.: Construction of Near-Optimal Vertex Clique Covering for Real-World Networks. Computing and Informatics, Vol. 34, 2015, No. 6, pp. 1397–1417.

[12] Chalupa, D.: On Combinatorial Optimisation in Analysis of Protein-Protein Interaction and Protein Folding Networks. In: Squillero, G., Burelli, P. (Eds.): Proceedings of the 19th European Conference on Applications of Evolutionary Computation (EvoApplications 2016). Springer, Lecture Notes in Computer Science, Vol. 9597, 2016, pp. 91–105, doi: 10.1007/978-3-319-31204-0_7.

[13] Culberson, J. C.—Luo, F.: Exploring the k-Colorable Landscape with Iterated Greedy. In: Johnson, D. S., Trick, M. (Eds.): Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge. American Mathematical Society, 1995, pp. 245–284.

[14] Czech, W.—Dzwinel, W.—Goryczka, S.—Arodz, T.—Dudek, A. Z.: Exploring Complex Networks with Graph Investigator Research Application. Computing and Informatics, Vol. 30, 2011, No. 2, pp. 381–410.

[15] Dorogovtsev, S.—Mendes, J. F. F.: Evolution of Networks. Advances in Physics, Vol. 51, 2002, No. 4, pp. 1079–1187, doi: 10.1080/00018730110112519.

[16] Erdős, P.—A. Rényi: On Random Graphs. Publicationes Mathematicae Debrecen, Vol. 6, 1959, pp. 290–297.

[17] Feo, T. A.—Resende, M. G. C.: Greedy Randomized Adaptive Search Procedures. Journal of Global Optimization, Vol. 6, 1995, No. 2, pp. 109-133.

[18] Friedrich, T.—He, J.—Hebbinghaus, N.—Neumann, F.— Witt, C.: Analyses of Simple Hybrid Algorithms for the Vertex Cover Problem. Evolutionary Computation, Vol. 17, 2009, No. 1, pp. 3–19, doi: 10.1162/evco.2009.17.1.3.

[19] Garey, M. R.—Johnson, D. S.: Computers and Intractability. Series of Books in the Mathematical Sciences, Vol. 29, W. H. Freeman, New York, 2002.

[20] Giel, O.—Wegener, I.: Evolutionary Algorithms and the Maximum Matching Problem. In: Alt, H., Habib, M. (Eds.): Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science (STACS '03). Springer, Lecture Notes in Computer Science, Vol. 2607, 2003, pp. 415–426.

[21] Girvan, M.—Newman, M. E. J.: Community Structure in Social and Biological Networks. Proceedings of the National Academy of Sciences of the United States of America, Vol. 99, 2002, No. 12, pp. 7821–7826, doi: 10.1073/pnas.122653799.

[22] Gramm, J.—Guo, J.—Hüffner, F.—Niedermeier, R.: Data Reduction and Exact Algorithms for Clique Cover. Journal of Experimental Algorithmics, Vol. 13, 2009, Article No. 2, doi: 10.1145/1412228.1412236.

[23] Halldórsson, M. M.: A Still Better Performance Guarantee for Approximate Graph Coloring. Information Processing Letters, Vol. 45, 1993, No. 1, pp. 19–23, doi: 10.1016/0020-0190(93)90246-6.

[24] Karp, R. M.: Reducibility Among Combinatorial Problems. In: Miller, R., Thatcher, J., Bohlinger, J. D. (Eds.): Proceedings of a Symposium on the Complexity of Computer Computations. Plenum Press, 1972, pp. 85–103, doi: 10.1007/978-1-4684-2001-2_9.

[25] Keil, J. M.—Stewart, L.: Approximating the Minimum Clique Cover and Other Hard Problems in Subtree Filament Graphs. Discrete Applied Mathematics, Vol. 154, 2006, No. 14, pp. 1983–1995.

[26] Kratsch, S.—Neumann, F.: Fixed-Parameter Evolutionary Algorithms and the Vertex Cover Problem. Algorithmica, Vol. 65, 2013, No. 4, pp. 754–771, doi: 10.1007/s00453-012-9660-4.

[27] Lehre, P. K.: Fitness-Levels for Non-Elitist Populations. In: Krasnogor, N., Lanzi, P. L. (Eds.): Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO '11). ACM, 2011, pp. 2075–2082.

[28] Leskovec, J.—Lang, K. J.—Dasgupta, A.—Mahoney, M. W.: Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics, Vol. 6, 2009, No. 1, pp. 29–123, doi: 10.1080/15427951.2009.10129177.

[29] Náther, P.—Markošová, M.: Positional Word Web and Its Numerical and Analytical Studies. Computing and Informatics, Vol. 30, 2011, No. 6, pp. 1287–1302.

[30] Neumann, F.: Expected Runtimes of Evolutionary Algorithms for the Eulerian Cycle Problem. Computers and Operations Research, Vol. 35, 2008, No. 9, pp. 2750–2759, doi: 10.1016/j.cor.2006.12.009.

[31] Neumann, F.—Wegener, I.: Randomized Local Search, Evolutionary Algorithms, and the Minimum Spanning Tree Problem. Theoretical Computer Science, Vol. 378, 2007, No. 1, pp. 32–40, doi: 10.1016/j.tcs.2006.11.002.

[32] Neumann, F.—Witt, C.: Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity. Springer, 2010.

[33] Ruiz, R.—T. Stützle: A Simple and Effective Iterated Greedy Algorithm for the Permutation Flowshop Scheduling Problem. European Journal of Operational Research, Vol. 177, 2007, No. 3, pp. 2033–2049.

[34] Schaeffer, S. E.: Graph Clustering. Computer Science Review, Vol. 1, 2007, No. 1, pp. 27–64, doi: 10.1016/j.cosrev.2007.05.001.

[35] Stanimirović, Z.—Mišković, S.: A Hybrid Evolutionary Algorithm for Efficient Exploration of Online Social Networks. Computing and Informatics, Vol. 33, 2014, No. 2, pp. 410–430.

[36] Sudholt, D.: A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms. IEEE Transactions on Evolutionary Computation, Vol. 17, 2013, No. 3, pp. 418–435, doi: 10.1109/TEVC.2012.2202241.

[37] Sudholt, D.—Zarges, C.: Analysis of an Iterated Local Search Algorithm for Vertex Coloring. In: Cheong, O., Chwa, K. Y., Park, K. (Eds.): Algorithms and Computation (ISAAC 2010). Springer, Lecture Notes in Computer Science, Vol. 6506, 2011, pp. 340–352.

[38] Sutton, A. M.—Neumann, F.: A Parameterized Runtime Analysis of Evolutionary Algorithms for the Euclidean Traveling Salesperson Problem. Proceedings of the 26th Conference on Artificial Intelligence (AAAI-12), AAAI Press, 2012, pp. 1105–1111.

[39] Welsh, D. J. A.—Powell, M. B.: An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. The Computer Journal, Vol. 10, 1967, No. 1, pp. 85–86.

[40] Witt, C.: Worst-Case and Average-Case Approximations by Simple Randomized Search Heuristics. In: Diekert, V., Durand, B. (Eds.): Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS 2005). Springer, Lecture Notes in Computer Science, Vol. 3404, 2005, pp. 44–56, doi: 10.1007/978-3-540-31856-9_4.

[41] Witt, C.: Analysis of an Iterated Local Search Algorithm for Vertex Cover in Sparse Random Graphs. Theoretical Computer Science, Vol. 425, 2012, pp. 117–125, doi: 10.1016/j.tcs.2011.01.010.

[42] Yuan, Z.—Fügenschuh, A.—Homfeld, H.—Balaprakash, P.—Stützle, T.— Schoch, M.: Iterated Greedy Algorithms for a Real-World Cyclic Train Scheduling Problem. In: Blesa, M. J., Blum, C., Cotta, C., Fernández, A. J., Gallardo, J. E., Roli, A., Sampels, M. (Eds.): Proceedings of the 5th International Workshop on Hybrid Metaheuristics (HM 2008). Springer, Lecture Notes in Computer Science, Vol. 5296, 2008, pp. 102–116.

David Chalupa received his Master's degree in software engineering and his Ph.D. degree in applied informatics from the Slovak University of Technology in Bratislava, Slovakia, in 2011 and 2014, respectively. He is currently Postdoctoral Fellow in the Operations Research Group at the Aalborg University in Denmark. Prior to that he was with the Computational Science Research Group (CSRG) at the University of Hull in the United Kingdom. His research interests include heuristics and metaheuristics, combinatorial optimisation and complex networks.

Jiří Pospíchal received his diploma degree in physical chemistry from the University of Jan Evangelista Purkyně in Brno, Czech Republic in 1984, and his Ph.D. degree in chemistry from Faculty of Chemical and Food Technologies at the Slovak University of Technology, Bratislava, in 1990. From quantum chemistry and computer assisted organic synthesis he soon transferred his interests to computer science and is now Professor of applied informatics at the Faculty of Natural Sciences at the University of Ss. Cyril and Methodius in Trnava, Slovakia. His research interests are evolutionary algorithms, artificial intelligence, neural networks, and graph theory.

Computing and Informatics, Vol. 37, 2018, 405–423, doi: 10.4149/cai_2018_2_405

COMPRESSION OF TEXTUAL COLUMN-ORIENTED DATA

Vinicius Fulber Garcia, Sergio Luis Sardi Mergen

Department of Languages and Computer Systems
Universidade Federal de Santa Maria
Av. Roraima, 1000, Santa Maria, Brasil
e-mail: {vfulber, mergen}@inf.ufsm.br

Abstract. Column-oriented data are well suited for compression. Since values of the same column are stored contiguously on disk, the information entropy is lower if compared to the physical data organization of conventional databases. There are many useful light-weight compression techniques targeted at specific data types and domains, like integers and small lists of distinct values, respectively. However, compression of textual values formed by skewed and high-cardinality words is usually restricted to variations of the LZ compression algorithm. So far there are no empirical evaluations that verify how other sophisticated compression methods address columnar data that store text. In this paper we shed a light on this subject by revisiting concepts of those algorithms. We also analyse how they behave in terms of compression and speed when dealing with textual columns where values appear in adjacent positions.

Keywords: Compression, column-oriented databases, LZ, PPM, BWT, entropy encoding, DSM, NSM, PAX

Mathematics Subject Classification 2010: 68P30

1 INTRODUCTION

Traditional relational databases use a page layout called NSM (N-ary Storage Model) where rows are stored contiguously on disk. Recent works propose a different page layout called PAX (Partition Attribute Across) where columns of relational tables are stored contiguously on disk. A similar physical organization is also employed by column-oriented databases, such as MonetDB.

This innovative data arrangement provides faster access for specific query patterns, such as those requiring a small amount of columns. The reason is that less IO is needed to retrieve the values of interest, as they fit in less data pages – no space is wasted in the page with values from columns that are not needed [12]. Additionally, all information stored in a page belong to the same domain and data type. This homogeneity allows data compression algorithms to achieve higher compression ratios if compared to physical arrangements like NSM where data inside a page is much more diverse [1].

Compression in column-oriented data is achieved in several different ways, depending on the nature of data. Sybase IQ [14] and Vertica [11], two of the industry's leading column-oriented databases, use variations of the LZ compression method when compressing high-cardinality texts. LZ is suited for data with a certain degree of redundancy, such as sentences written in natural language, where words are grammatically linked. The method is able to encode redundant parts effectively, achieving good compression associated with a low execution time.

We note that there are other compression methods suited for texts, such as PPM and BWT. Several works (e.g. [9]) report empirical results achieved when applying these methods on the Calgary Corpus, a popular collection of documents used for compression [4]. However, the redundancy in column-oriented data is naturally different than redundancy found in conventional files. Also, there is a limit on the size of a database page. The question of how such methods behave when compressing columnar text organized as pages still deserves investigation.

The goal of this paper is to compare how effective are the BWT, LZ and PPM methods with respect to compression and execution time when compressing column-oriented textual data. We start describing our motivation and running example (Section 2). From Section 3 to 6 we revisit the concepts behind the methods that are part of the evaluation. For each method we outline in general terms how the patterns found are explored to achieve compression. In the final part the paper is dedicated to reporting on the experimental results (Section 6) and presenting our concluding remarks (Section 7).

2 MOTIVATION AND RUNNING EXAMPLE

Figure 1 shows different page layouts for a table containing columns YEAR, STATUS and COMMENT. The NSM is the typical design choice of relational databases that store rows contiguously on disk. Conversely, DSM (Decomposition Storage Model) and PAX are designed to keep values of the same column together [2]. DSM splits a table into as many columns as it has. Then, each column is stored in a separate group of pages. PAX follows the same principle, but uses the concept of mini-pages inside a page. The rationale is that whole records can be read from a single PAX page. DSM is the precursor of modern columnar databases, while the general idea of

PAX is employed by some relational database vendors, like the the Hybrid Columnar Compression (HCC) used by Oracle Exadata [3].

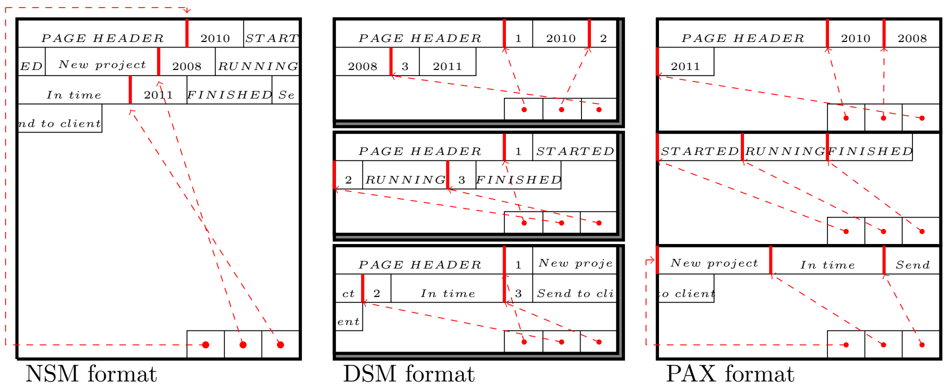


Figure 1. Three different page layouts

In what follows, we make a distinction between heavy-weight and light-weight compression methods. We use the term heavy-weight to refer to methods where the encoding of upcoming symbols relies on information gathered from the previously encoded symbols. The term light-weight is used otherwise.

The first thing to notice about the illustrated example is that the values of the STATUS and YEAR columns in the DSM/PAX page layouts are particularly suited for light-weight compression methods, such as dictionary encoding and frame of reference, respectively. In the former, one value is coded as an index that points to a dictionary where all possible status values can be found. In the latter, one value is coded as the difference from a reference value (the average stored year value).

Some light-weight methods (like the ones mentioned above) compress values into fixed-width codes. This allows fine grained decompression of single values, without the need of full-page decompression. There is also the possibility to operate directly on compressed data. Working with compressed data means that more information fits in memory and the number of cache misses is reduced if the same value is needed again. Also, the vectorized compressed data can be more efficiently handled by modern CPUs that can pipeline and parallelize instructions [19].

The shortcoming of light-weight compression methods is that they do not address well skewed and high-cardinality values, such as the ones found in the COMMENT column. Values of comments are typically sentences from a written language. In this case, the patterns that emerge are different, comprising root words, frequent sequences of contiguous words, or even a high frequency of single words, such as prepositions and nouns. Heavy-weight methods are best suited under these circumstances.

When using a heavy-weight method, the execution engine of a query processor cannot operate on the compressed data directly. The whole page/mini-page needs to be decompressed before a specific value can be accessed. The need for decompressing text obviously slows down query execution. On the other hand, the IO cost is reduced. For instance, leaving COMMENT uncompressed (or poorly compressed) in the PAX format may hinder the benefits of using light-weight methods to compress other mini-pages, resulting in less rows per page and more data transfers. Besides, a mini-page storing text occupies more space than a mini-page storing data types typically supported by light-weight methods, which makes text compression even more critical.

This is the motive that drove us into investigating compression methods suited for high-cardinality and skewed textual values. Throughout the remaining of the paper we revisit concepts of well-known heavy-weight compression methods in order to understand how different they are and why they are strong candidates to compress this sort of data.

Before we begin, we call the attention to the pages depicted in Figure 1 and the fact that data grow from the one side and auxiliary vectors grow from the other side. The vectors are useful for indirect access inside a page/mini-page. Examples are presence vectors to indicate nullable fields and offset vectors to indicate where a variable length record (in the case of a NSM) or a variable length field (in the case of PAX/DSM) begins.

Observe that the usage of heavy-weight compression methods implies that random access is not possible. Therefore, there is no need for an offset vector. We argue that, in those cases, the page/mini-page design can be simplified by dropping this kind of bookkeeping. If a column accepts texts with variable length, instead of storing offsets, a special delimiter symbol could be used to separate one value from the next.

For instance, suppose there are seven fields of the COMMENT column. The value of the forth field is 'abc' and the other fields are either nulls or blanks. Using the semicolon (;) as the special delimiter symbol, the fields put together become as follows:

$$\boxed{;\quad ;\quad ;\quad \mathrm{a}\quad \mathrm{b}\quad \mathrm{c}\quad ;\quad ;\quad ;}$$

This is the information to be encoded, and the one we use as the running example from now on. Observe the presence of two runs, for leading and trailing delimiters. This is a kind of pattern that occurs when values of a column appear in adjacent positions. It is a simple example, but it serves our purpose of illustrating how patterns are coded by the investigated methods.

Throughout the rest of the paper we use the term *message* to refer to the contents of the running example. We also use the term *symbol* to indicate each byte of the *message*. We assume the files use the ASCII encoding scheme, so that each byte maps to a different *symbol*. Compression is measured as bits per code (bpc), the average number of bits needed to code a character.

3 ENTROPY ENCODING

In information theory, the entropy of a *message* is a measure of how unpredictable the *message* is. A higher entropy means it is more difficult to predict what *symbols* are more likely to appear. For instance, *messages* where one *symbol* is much more common than the others (like the delimiter in a very sparse dataset) have a low entropy, since we can predict that *symbol* will appear quite often.

The purpose of entropy encoding is to approximate the entropy of a *message*, that is, to use the minimal amount of bits to represent information. The lower the entropy, the more compressed the *message* can be. Two of the most popular compression methods based on entropy are the Huffman coding [10] and arithmetic coding [16]. In what follows we present the differences between them.

Huffman Coding: The Huffman coding assigns to each *symbol* of the *message* a unique and unambiguous sequence of bits, reserving the smaller sequences to the most frequent *symbols*. A binary tree can be used to create the mapping through a greedy algorithm that iteratively puts the two nodes (*symbols*) with the minimum frequencies under the same parent. Figure 2 shows the tree generated based on the contents of the COMMENT *message*. Observe that coding the delimiter requires a single bit. After compression, the nine characters are transformed into a 15 bit sequence (000101101110000), giving a compression of 1.66 bpc.

Symbol Mapping

Symbol	Frequency	bit sequence
;	6	0
a	1	10
b	1	110
c	1	111

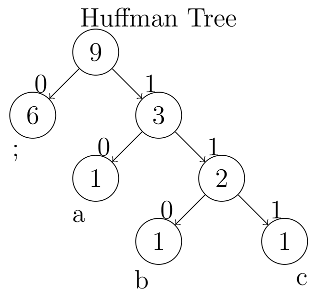


Figure 2. The Huffman tree created based on the comment *message*

Arithmetic Coding: Unlike the Huffman code, there is no unique bit sequence that determines each *symbol* in the arithmetic coding method. Instead, the bits lead to a value that indicates the probability of occurrence of that exact sequence of *symbols* being compressed. The probability ranges from zero to one. For sufficiently long *messages*, the probability would require a floating point precision higher than computers are able to express. To circumvent this architectural problem, a fixed-point precision value is used. When the value is about to overflow, the most meaningful bits are flushed out and the value is shifted to the right.

Figure 3 illustrates how the encoded probability is updated as the *symbols* are processed. Initially the probability of each *symbol* is divided into a scale ranging from zero to one and the probability range is updated as the *symbols* are processed. To simplify, the first case shows the probability distribution after the nine *symbols* of the *message* were processed. At this point, the probability of finding another delimiter is 66 %. If the delimiter is indeed found, the probability is updated as demonstrated in the second case. As it shows, the probability of finding another delimiter drops to 43 %. The probability keeps being updated as the remaining *symbols* are processed, and eventually the most significant bits are flushed.

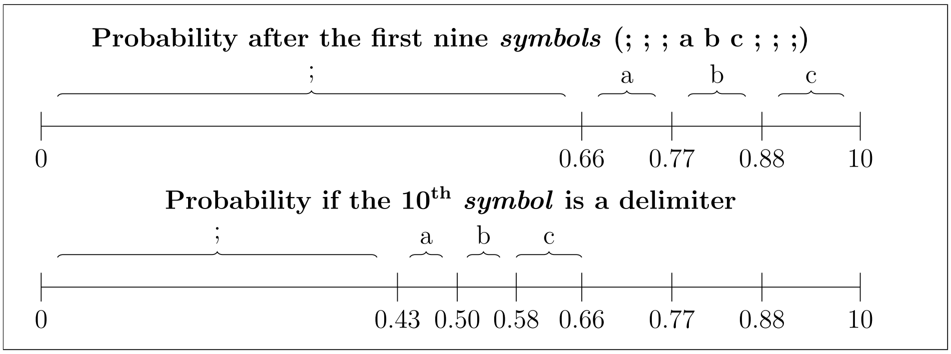

Probability after the first nine *symbols* (; ; ; a b c ; ; ;)
;
a
b
c
0
0.66
0.77
0.88
10
Probability if the 10th *symbol* is a delimiter
;
a
b
c
0
0.43
0.50
0.58
0.66
0.77
0.88
10

Figure 3. Encoding probabilities based on the running example *message*

In comparison to the Huffman code, the arithmetic coding is better at approximating the entropy of the *message*. Moreover, it simplifies the process of adaptation, where the probability of each *symbol* is updated as the *symbols* are being read, instead of having a fixed precomputed probability. On the other hand, Huffman codes allow reading from arbitrary positions of the compressed *message*, which is not possible using arithmetic coding. In either case, the effectiveness of both methods is highly dependent on the existence of very frequent *symbols*. In most scenarios the entropy coding is not used alone, but as part of a more complex method, such as BWT, LZ and PPM, as we detail next.

4 BURROWS WHEELER TRANSFORM (BWT)

The compression method proposed by [5] is divided in stages, as illustrated in Figure 4. The information flows from left to right and each stage transforms data into a format suited to the next stage.

Burrows Wheeler Transformation Stage: During the first stage, the *message* (with n characters) is copied into n rows, where row i is the same as row $i-1$

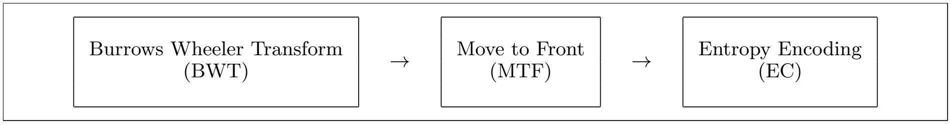


Figure 4. BWT Stages

rotated one character to the right and row 0 is the original *message*. The rows are then ordered lexicographically. From this arrangement, two data elements are passed to the next stage. The first is the content of the last column of the sorted rows. The other is the index of the sorted rows that contains the original *message*. The purpose of the latter is to reconstruct the *message* during decompression.

The reasoning behind BWT is to explore the fact that some characters usually come before other characters in many written languages. In such cases, a character that usually precedes others tends to appear in adjacent positions in the last column. As we discuss later, this is desired when it comes to compression. To illustrate, Figure 5 presents the Burrows Wheeler transformation of the COMMENT *message*. Observe that all delimiter *symbols* appear next to each other in the last column of the sorted matrix. Curiously (and mostly because the *message* is small) the leading and trailing runs of ';' were merged into a single run.

1	;	;	;	a	b	c	;	;	;		;	;	;	;	;	;	a	b	c
2	;	;	;	;	a	b	c	;	;		;	;	;	;	;	a	b	c	;
3	;	;	;	;	;	a	b	c	;		;	;	;	;	a	b	c	;	;
4	;	;	;	;	;	;	a	b	c		;	;	;	a	b	c	;	;	;
5	c	;	;	;	;	;	;	a	b	→	;	;	a	b	c	;	;	;	;
6	b	c	;	;	;	;	;	;	a		;	a	b	c	;	;	;	;	;
7	a	b	c	;	;	;	;	;	;		a	b	c	;	;	;	;	;	;
8	;	a	b	c	;	;	;	;	;		b	c	;	;	;	;	;	;	a
9	;	;	a	b	c	;	;	;	;		c	;	;	;	;	;	;	a	b
	Before the sort operation										After the sort operation								

Figure 5. Applying the Burrows Wheeler transform to the comment *message*

Move To Front Stage: The purpose of the MTF stage is to encode each character of the last column as a numeric index ranging from zero to 255. This index refers to a lookup table that contains all possible characters. The table is first built with characters occupying arbitrary positions. Then, when a character is encoded, it is moved to the front of the list. When the next character to encode is the same as the previous one, the resulting index position is zero, since the character looked up is now found in the first position of the table. At the end of the transformation, the result is expected to be formed by many consecutive zeros. Also, few index positions will be very frequent (the smaller ones). This allows entropy coding to be performed, as we detail next.

Figure 6 presents the output generated after each stage of BWT (for the running example). Line 3 shows how MTF transforms the *symbols* received from the previous stage. To get this result we assume that the characters ‘;’, ‘a’, ‘b’, ‘c’ initially occupy the positions 0, 1, 2 and 3 of the lookup table, respectively. Observe that every repeated adjacent *symbol* is encoded as zero.

1. original *message*	;	;	;	a	b	c	;	;	;
2. BWT output	c	;	;	;	;	;	;	a	b
3. MTF *symbols*	3	1	0	0	0	0	0	2	3
4. EC output	1110	10	0	0	0	0	0	110	1110

Figure 6. Encoding the running example *message* with BWT

Entropy Encoding Stage: The input for this stage is composed by a few indexed positions with a very high frequency. Entropy coders (like Huffman and arithmetic encoding) can explore this property to actually achieve compression. With respect to our running example, the Huffman tree created for the indexes would yield the sequences ‘0’, ‘10’, ‘110’ and ‘1110’ for the values 0, 1, 2 and 3, respectively[1]. Line 4 of Figure 6 shows the Huffman coding of the input received from the MTF stage. In this particular case the compressed data is three bits longer than the data compressed when only the Huffman encoding is used (Section 3). This is mostly due to the size of the example. As we demonstrate on the experimental section, the behaviour is different when dealing with larger files.

Decompression is achieved by executing the stages in opposite direction, starting from the entropy encoding and finishing with the Burrows Wheeler Transformation. All stages are reversible, including the transformation. The last column along with the index of the original text is enough information to reconstruct the original *message*.

The BWT method is effective in compressing text, especially if data is domain specific. The more specific is the domain, the shorter is the set of *symbols* that precedes characters. This translates into longer runs of the same *symbol* after the BWT stage. Runs of the delimiter *symbol* may also translate into runs of the same delimiter, as the example shows. A handicap of BWT is that it is very memory intensive, since the matrix transformation requires the whole *message* to be read. To reduce memory requirements, the input is divided into blocks, and each block is compressed separately. Higher compression ratios can be obtained when working with longer blocks, as we demonstrate in Section 7.

[1] The Huffman codes (or *symbols* frequencies) need also be saved as part of the encoded file to allow decompression.

5 LEMPEL-ZIV

This compression method compresses sequences of *symbols* of varying sizes by replacing them with a code that refers to a dictionary entry. The dictionary is formed by *symbols* of the *message* that were already processed.

The idea of using the previous *symbols* as a dictionary was originally proposed by Lempel and Ziv, which is why this kind of dictionary-based compression method is commonly referred to as Lempel-Ziv, or LZ to short. The name LZ77 is used to identify the original idea, where the dictionary is a sliding window formed by past *symbols* [17]. The LZ78 is a variant where the dictionary is explicitly built as a table, and its entries are accessed by an index [18].

Several other variations appeared, such as LZW (used in Sybase IQ) and LZO (used in Vertica). The one we have presented here is based on LZ77. It is a simplification of the method used in GZIP, where the code is formed by a pair (distance, length). The distance is an index to a position of the dictionary where a sequence starting at the current *symbol* is found. The length is the amount of *symbols* to be coded from that index position. The distance is incremented backwards from the current *symbol.* The value zero indicates that the current *symbol* was not found in the dictionary. In such cases the length is replaced by the actual *symbol.*

Figure 7 shows what codes are generated for the COMMENT *message.* The current *symbol* is circled. Symbols before the current one becomes the part of the sliding window. The first six *symbols* are encoded as literals (no dictionary entry was used). The last three *symbols* (a sequence formed by repeated delimiters) are packed as a single code. Observe that the second (or the third) *symbol* of the *message* could also point to a dictionary entry, since a delimiter is already a part of the dictionary. However, coding small sequences instead of outputting literals may actually result in higher codes.

Message									Distance	Length/Literal
$\textcircled{;}$	;	;	a	b	c	;	;	;	0	;
;	$\textcircled{;}$	;	a	b	c	;	;	;	0	;
;	;	$\textcircled{;}$	a	b	c	;	;	;	0	;
;	;	;	$\textcircled{a}$	b	c	;	;	;	0	a
;	;	;	a	$\textcircled{b}$	c	;	;	;	0	b
;	;	;	a	b	$\textcircled{c}$	;	;	;	0	c
;	;	;	a	b	c	$\textcircled{;}$	;	;	6	3

Figure 7. Encoding the comment *message* with LZ

Since the codes refer to a part of the *message* that have already been processed, during decompression it is possible to use the codes in order to reconstruct the original *message.* Decompression is much faster than compression, since there is no need to locate *symbols* from the sliding window. Instead, it only needs to copy *symbols* from the sliding window (based on the code) into the output.

Gzip implements the DEFLATE standard [7], which imposes an agreement on the code format, the maximum size of the window (32 k), the maximum length ahead (258 bytes), among others. It also establishes that the distances and lengths are further compressed using two separate Huffman trees.

Compression tools that implement this standard (like gzip) are heavily used. One reason for the popularity (apart for being a recommendation) is that it is relatively straightforward to implement a decompression algorithm that is compliant with the standard. Besides, this kind of dictionary-based compression is not only fast but also achieves good compression ratios for most of the file formats.

With respect to column-oriented textual data, the occurrence of many similar (or equal) values can also be encoded as pointers to a dictionary entry. Common expressions already processed can be used as dictionary entries for coding occurrences of other common expressions yet to come. The impact this kind of information has on a LZ based compression method is detailed in Section 7.

6 PPM

The PPM (Prediction by Partial Matching) compression method codes one *symbol* at a time. Given the *symbol* to code, PPM predicts the probability of occurrence of that *symbol.* This value is then coded using arithmetic encoding. Highly frequent *symbols* are encoded with fewer bits. The main idea can also be adapted so that other entropy coders (such as Huffman) can be used.

The probability estimation takes into account the context, which means the set of *symbols* that precedes the *symbol* being coded. Most PPM approaches use Markov models of different orders to issue a prediction. If the highest-order model is unable to predict a *symbol* (it never occurred in that context before), the next-higher-order model is used. In the worst case this goes on until reaching the zero-order model that is able to predict all *symbols.*

When a *symbol* cannot be predicted by an order, PPM issues a signal indicating that a *symbol* never seen before has appeared. This special signal is also referred to as escape *symbol.* The probability of the escape depends on the variant of the PPM used. One of the proposed ideas (called PPM-C) was to count the number of different *symbols* that occurred in that context and use it to compute the probability [13].

To illustrate, consider the comment *message* reintroduced below. The arrow points at the next *symbol* to encode and the curly bracket indicates the context to be used to determine the probability of the next *symbol.* The length of the context corresponds to the length of the highest Markov model. Empirical results reported by [13] show that compression is best when using at least four as the higher order. For the sake of presentation we use a maximum order of two.

$$; \quad ; \quad ; \quad a \quad b \quad c \quad ; \quad \underbrace{; \quad ;}_{\textit{Context}} \quad \overset{\downarrow}{?}$$

Figure 8 shows the generated context tree. A context in this tree is formed by the *symbol* of a node concatenated with all of its parents. The parenthesis indicates the frequency of that context. For instance, the leftmost leaf node shows that the context ';;;' has already occurred twice. The circles indicate the current contexts from the highest to the lowest order.

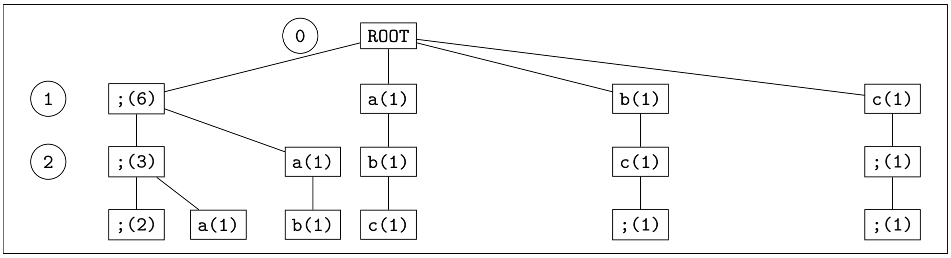


Figure 8. Context tree of the comment *message* (maximum order of two)

If the next *symbol* to encode (at the arrow position) was a delimiter, the highest order 2 (';;') would issue a probability of 40 % (two delimiters out of five occurrences: the delimiters themselves, one character 'a' and two escapes). If the *symbol* was the character 'a', the probability would be lower (20 %). If the *symbol* is new in that context, like the character 'b', an escape probability is issued (40 %) and the search continues in the order 1 (';'). Symbols with equal probability do not conflict since they occupy different intervals in the probability scale.

One technique that achieves better compression is to ignore *symbols* that exist in higher orders. For instance, if the incoming *symbol* is 'b', it would be found at order 1. In this case, the 'b' probability at order 1 would be 25 % (one out of four), if *symbols* from the upper level are ignored, and 7 % (one out of thirteen) otherwise.

The compression process is reversible. The encoding uses the context model, which is constructed based on the *symbols* already processed. Thus, given a probability, it is possible to use the current model constructed so far in order to locate the next *symbol* to be decoded.

This technique is very useful for compressing texts written in natural language, since it is able to predict with a high level of hit rate the next character of small words (or roots/prefixes/suffixes) where their length is below the maximum order. Given this, it is usually better than LZ to code small patterns. Also, it is able to encode patterns found outside the LZ window. The bottleneck is decompression speed, since PPM needs to traverse the list of children of a node to find out the one to decode. The cost can be amortized by keeping the children sorter by frequency, but the cost is still meaningful, as we discuss next.

7 EXPERIMENTAL RESULTS

The purpose of this section is to evaluate how the three investigated compression methods behave with columnar data. The experiments are divided in three parts. First we analyse how the compression varies according to the nature of textual data. Then we show how the compression and compression/decompression speed varies according to the amount of data compressed. Finally we evaluated alternative LZ based methods.

Initially, two commercially available compression tools were evaluated: BZIP2 (version 1.0.6) and GZIP (version 1.2.4) that implement the BWT and LZ methods, respectively. To avoid biased evaluations, the executables were used as is, without any kind of tuning, and the meta-data overhead was stripped from the compressed output. We also evaluated an implementation of PPM-C, implemented as part of this work. Our version uses four levels of context and it ignores symbols from higher orders in order to improve compression. All algorithms were written in C.

The algorithms were tested on a Pentium Dual-Core with 2.5 GHz. Compression speed was measured as the amount of data that is compressed by time, and decompression speed is measured as the amount of data that is decompressed by time. The time obtained is an average of 30 executions, ignoring the 10 % lesser and higher times. The experiments were held in a minimalist operating system, where functions are reduced to a bare minimum that does not sacrifice stability.

7.1 How the Data Format Impacts Compression

Throughout the paper we have seen that all of the investigated compression methods are able to handle textual data, and they explore the patterns found in the already processed *message* to achieve compression. However, text can be organized in very distinct ways, following a rigid or relaxed structure, or having no structure at all. Here we analyse how compression is affected by different text formats.

Two datasets were used, one for columnar data and the other for conventional files. The columns were taken from the TPC-H benchmark, which is a set of tables commonly used to evaluate database transaction processing features [15]. The conventional files were taken from the Calgary Corpus. As mentioned earlier, it is a set of files traditionally used to evaluate compression algorithms.

Results are presented for three high-cardinality columns from TPC-H: CUSTOMER.COMMENT, LINEITEM.COMMENT and PART.NAME. Each column was stored as a separate file, and a reserved *symbol* was used to separate one value from the next. The columnar data are compared against four files from the Calgary Corpus: a book (BOOK2), a paper (PAPER2), a source code written in Pascal (PROGP), and a list of bibliographic references (BIB). Other files were considered as well (from TPC-H and Calgary), and the variations observed were the same.

All uncompressed files were divided into chunks of 16 KB, and each chunk was compressed separately. What we report is the average compression achieved for each file, measured as bits per code (bpc). The compression is limited to chunks (and not the whole file) to emulate database compression schemes that handle one page at a time.

Figure 9 shows how the compression varies according to the nature of textual data. The results show that BZIP2 and PPM achieve better compression than GZIP in most cases. Additionally, the columnar values are more compressible in comparison to other texts written in natural language and even to the well behaved PROGP. One of the possible reasons is that it is more likely to find patterns in a list composed by small textual fields than it is in a single and longer instance of a text.

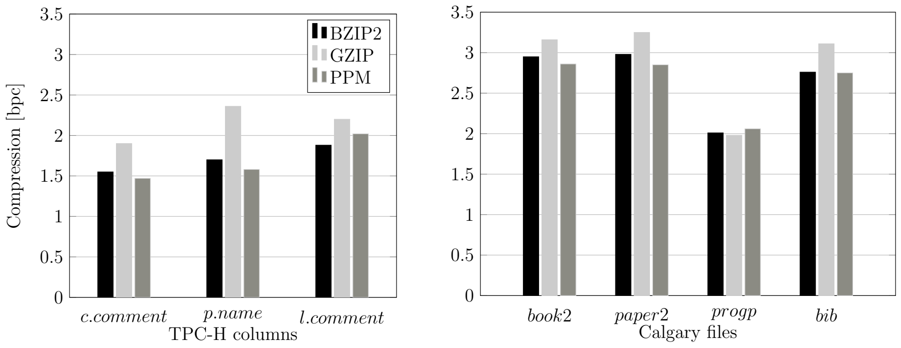


Figure 9. Compression results

Interestingly, the compression improvement of BZIP2 and PPM over GZIP is more meaningful with the columnar values. For instance, PPM uses almost 33 % less bits than GZIP when compressing PART.NAME. The improvement of PPM over GZIP with the Calgary files is less apparent. With the highly structured PROPG, PPM is actually worst. This behavior lead us to the conclusion that BZIP2 and PPM explore higher levels of redundancy better than GZIP.

7.2 How the Data Length Impacts Compression

This experiment is focused on columnar data stored in database pages. It shows how the bpc and compression/decompression speed varies according to how large the pages are. The evaluation was done by compressing data chunks of different sizes. We focused on the compression of the PART.NAME column, but similar results were obtained for other textual columns as well.

The size of a chunk is related to the amount of uncompressed data, not to the size of a page. For instance, GZIP takes 1.7 bits per code to encode 16 KB of comments, resulting in 3.481 bytes of compressed text in a page of unspecified size. What we need are measures taken from the full occupation of a page with a determined size. We did this by applying a linear interpolation of the results achieved when compressing chunks of uncompressed data. The results are shown in Figure 10. Solid lines indicate compression and dotted lines indicate compression speed (at the left) and decompression speed (at the right) as the amount of bytes coded per millisecond. The size of the pages vary from 1 KB to 128 KB.

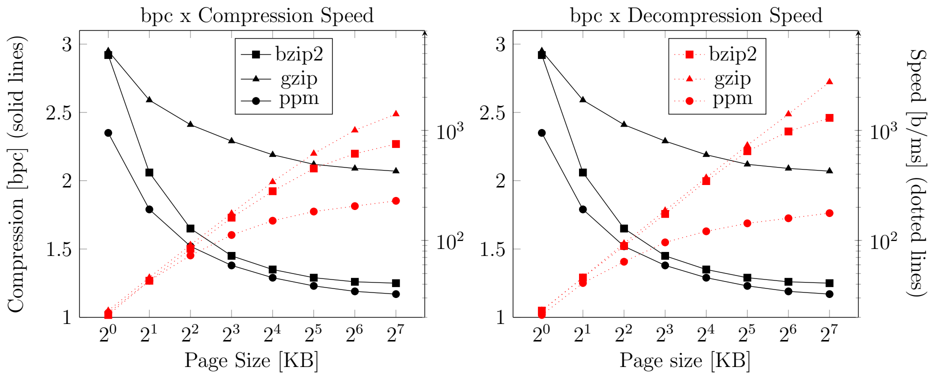


Figure 10. Measuring the compression of the name column

As the page size gets longer, GZIP becomes notably faster than BZIP2, which is in turn faster than PPM. Despite the speed, GZIP does not explore the size factor as well as the alternatives to obtain better compression. The reason is related to the 32 KB sliding window of GZIP. A longer *message* means more patterns can be found. However, if the patterns lie beyond the limits of the sliding window, they are not used to encode the incoming *symbols*.

For sufficiently long *messages*, GZIP is the clear winner. It may loose regarding the compression, but the speed compensates, especially for decompression. However, databases use pages as the transfer unit between disk and memory, and pages have a rather limited size. For the sake of comparison, MySQL uses 16 KB as the default page size and Oracle 10g uses pages from 4 KB to 8 KB. Therefore, it makes sense to consider scenarios where smaller *messages* need to be compressed. This type of scenario is analysed in the next section.

7.3 How Page-Fitting Messages Impacts Compression

As we are working with columnar compression, two page layouts are considered: DSM and PAX. These layouts subsume the main strategies adopted when values of the same column are stored together. Our intention was to evaluate the methods effectiveness when filling pages with compressed data.

When working with the DSM page layout, the whole page is dedicated to a single column. We can see from Figure 10 that BZIP2 is an interesting solution for the storage of PART.NAME values using this layout and regular page sizes (from 4 KB to 16 KB). To give a better look, Table 1 details the measurements obtained when storing small messages. BZIP2 achieves a bpc 32 % superior than GZIP when working with a 4 KB page. In contrast, GZIP is only 8 % and 6 % faster in compression and decompression speed, respectively.

When working with the PAX page layout, the page is divided into n mini-pages for the n columns it stores. Unlike DSM, only part of the page is reserved to the storage of columns whose values are textual. It means we need to look at the compression of even smaller *messages*, where the tendency is that the compression of BZIP2/PPM proportionally degrade and execution time proportionally improve, in comparison to GZIP.

To exemplify, consider an hypothetical case where the mini-page for the PART.NAME column occupies 2 KB and compare it with the storage of 4 KB of compressed values stored using the DSM layout. As Table 1 indicates, PPM is now superior than its competitors. It used 1.78 bpc to encode 2 KB of PART.NAME values, which is still almost 32 % better than GZIP. Besides, compression and decompression speeds are similar. Again, the reason is related to the sliding window of GZIP. For long *messages*, BZIP2 and PPM spend more time analysing longer parts of the *message*, whereas GZIP is bounded by the window, regardless of the *message* size. The overhead is reduced when the *message* is small, and PPM/BZIP2 become competitive in terms of execution time.

Measure	Size [KB]	bzip2 [bpc]	gzip [bpc]	ppm [bpc]
bpc	2^1	2.05	2.58	1.78
comp. [b/ms]	2^1	43	46	43
decomp. [b/ms]	2^1	46	46	41
bpc	2^2	1.64	2.40	1.52
comp. [b/ms]	2^2	84	91	73
decomp. [b/ms]	2^2	89	94	64

Table 1. Detailed measurements using pages of 2^1 and 2^2 KB

Table 2 presents some of the possible settings when 2 KB are dedicated to a single mini-page for the PART.NAME column, using PPM. For instance, if the page is 4 KB long, and each textual field occupies 40 characters, there would be enough space in the page for 230 records. For each record, 8.9 bytes of compressed data are used by the NAME field and another 8.9 bytes of compressed data are used by the fields of the remaining columns. In general, the amount of space left for other compressed columns is reasonable, especially if we consider a small amount of columns or columns that are well compressed, such as numeric values. Besides, more space for the remaining columns can be used by reducing the number of records stored or increasing the page size, as the table shows. What is worth noting here is that, if GZIP is used instead of PPM, only 68 % of the records would fit in a page, under the same conditions described in the table. If the textual values are not compressed at all, the amount is reduced to 22 %. The difference is significant, and should not be overlooked when deciding whether it is worthy to compress textual values and which compression method to use.

Page Size	Text Length	# of Records	Text Compression	Space Left
4 096	40	230	8.9	8.9
4 096	60	153	13.3	13.3
4 096	80	115	17.8	17.8
8 192	20	460	4.4	13.3
8 192	40	230	8.9	26.7

Table 2. Some settings considering PAX with a 2 KB mini-page reserved for a textual column compressed with PPM

7.4 How Other LZ Based Methods Impact Compression

The previous sections showed that GZIP is not a compelling approach for the compression of page-fitting *messages*. It is usually faster than the alternatives for large files. However, there are no remarkable differences in efficiency when files are small. Here we extend this analysis by comparing additional LZ based methods. The algorithms are presented below:

LZF (version 3.6): This method is a LZ77 variant. One of its features is that the distance/length codes produced are not further compressed using any kind of entropy encoding. Therefore, it is a fast method, especially for decompression. However, compression rate is usually compromised.

ZSTD (version 0.8.2): This is another LZ77 variant, recently proposed by Yann Collet at Facebook [6]. The entropy encoding stage is done by a fast method based on the asymmetric numeral systems (ANS) theory [8]. The author claims that ZSTD is very efficient and achieves acceptable compression.

We have also evaluated GZIP with different settings. This method can be tuned by allowing the compressor to spend more time inside the sliding window trying to find longer patterns to encode. A parameter value ranging from 1 to 9 defines how much effort should be spent. Smaller values means the search is faster, at the cost of a reduced compression. The default value (6) represents a balance between high compression and an acceptable response time. The two extremes (1 and 9) were included in the experiment, so we can verify how much compression is possible and how fast the method can be.

Table 3 brings the comparison, when compressing the PART.NAME column, considering small *messages* (that fit in a regular page) and long *messages*. There are remarkable differences between the two scenarios. The first thing to notice is that the non-LZ methods tested are not suited when *messages* are long. Despite the fact that they reach a low bpc, the processing time is much higher than the LZ alternatives. Also, GZIP-1 is an interesting choice for long *messages*. It is practically twice as more efficient that GZIP standard for compression time, and compression is only 6 % worst. Decompression time is similar, as the processing is pretty much limited to copying *symbols* from the window to the output. LZF is a little faster, as there is no entropy decoding involved.

Measure	Size [KB]	ppm	bzip2	gzip	gzip-9	gzip-1	zstd	lzf
bpc	2^1	1.78	2.05	2.58	2.58	2.74	3.01	3.73
comp. [b/ms]	2^1	43	43	46	47	47	54	49
decomp. [b/ms]	2^1	41	46	46	47	46	54	49
bpc	2^7	1.16	1.25	2.06	2.06	2.33	2.28	3.02
comp. [b/ms]	2^7	229	752	1 407	1 408	2 458	2 084	2 637
decomp. [b/ms]	2^7	177	1 300	2 751	2 776	2 781	2 326	2 912

Table 3. Comparison between lz based methods and non-lz based methods

What is worth noting is that, when *messages* are small, the compression and decompression time of all methods are levelled, and the compression achieved by PPM and BZIP2 are substantially better. In fact, PPM is particularly appealing. It is 15 % better than the second best (BZIP2) and 31 % better than the best LZ based method (GZIP). The combination of outstanding compression and competitive execution time (even for decompression) makes it a strong candidate for the compression of page-fitting columnar data.

8 CONCLUDING REMARKS

In this paper we analysed the compression of textual values from column-oriented data stored in page layouts based on DSM and PAX. We have seen that the difference in terms of number of records that fit in a page is significant when comparing the compressed and uncompressed format. This is even more important if we consider that CPUs are getting much faster compared to memory bandwidth. We argue that reducing the transfer load at the expense of having to decompress data before actually using it is an attractive trade-off.

Some of the current solutions for text compression are based on variants of the LZ method. This is a great alternative when dealing with long *messages*, since it is able to associate high compression with low execution time. However, our experiments show that other methods need to be taken into account when *messages* are short. This is the case if we consider that compression is devoted to single pages, and pages are small enough to make the BZIP2 and PPM compelling approaches.

With respect to PPM specifically, besides the applicability to text, we believe it is possible to leverage compression when dealing with short fields by adjusting the way probability is computed, especially if the fields follow a regular structure, even in the presence of outliers. For instance, dates and currency fields are expected to have the same *symbols* appearing at determined positions, and a probabilistic model is able to map this relation. This example is not only a topic we intend to further investigate, but it supports our claim that there are indeed data types that are not effectively covered by light-weight compression methods. Textual data types columns are some of them, and some others also exist.

REFERENCES

[1] Abadi, D. J.—Madden, S. R.—Hachem, N.: Column-Stores vs. Row-Stores: How Different Are They Really? Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. ACM, 2008, pp. 967–980, doi: 10.1145/1376616.1376712.

[2] Ailamaki, A.—DeWitt, D. J.—Hill, M. D.—Skounakis, M.: Weaving Relations for Cache Performance. Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01), San Francisco, CA, USA, Morgan Kaufmann Publishers Inc., 2001, pp. 169–180.

[3] Bach, M.—Arao, K.—Colvin, A.—Hoogland, F.—Osborne, K.—Johnson, R.—Poder, T.: Hybrid Columnar Compression. Expert Oracle Exadata, Springer, 2015, pp. 67–120, doi: 10.1007/978-1-4302-6242-8_3.

[4] Bell, T. C.—Cleary, J. G.—Witten, I. H.: Text Compression. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[5] Burrows, M.—Wheeler, D. J.: A Block-Sorting Lossless Data Compression Algorithm. SRC Research Report, Palo Alto, California, 1994.

[6] Collet, Y.—Turner, C.: Smaller and Faster Data Compression with Zstandard. Retrieved from `https://code.facebook.com/posts/1658392934479273/smaller-and-faster-data-compression-with-zstandard/`, 2017.

[7] Deutsch, L. P.: DEFLATE Compressed Data Format Specification Version 1.3. 1996, doi: 10.17487/rfc1951.

[8] Duda, J.—Tahboub, K.—Gadgil, N. J.—Delp, E. J.: The Use of Asymmetric Numeral Systems as an Accurate Replacement for Huffman Coding. Picture Coding Symposium (PCS), IEEE, 2015, pp. 65–69, doi: 10.1109/PCS.2015.7170048.

[9] Effros, M.: PPM Performance with BWT Complexity: A New Method for Lossless Data Compression. Proceedings of Data Compression Conference (DCC 2000), IEEE, 2000, pp. 203–212, doi: 10.1109/DCC.2000.838160.

[10] Huffman, D. A. et al.: A Method for the Construction of Minimum Redundancy Codes. Proceedings of the IRE, Vol. 40, 1952, No. 9, pp. 1098–1101, doi: 10.1109/JRPROC.1952.273898.

[11] Lamb, A.—Fuller, M.—Varadarajan, R.—Tran, N.—Vandiver, B.—Doshi, L.—Bear, C.: The Vertica Analytic Database: C-Store 7 Years Later. Proceedings VLDB Endowment, Vol. 5, 2012, No. 12, pp. 1790–1801, doi: 10.14778/2367502.2367518.

[12] Matei, G.: Column-Oriented Databases, an Alternative for Analytical Environment. Database Systems Journal, Vol. 1, 2010, No. 2, pp. 3–16.

[13] Moffat, A.: Implementing the PPM Data Compression Scheme. IEEE Transactions on Communications, Vol. 38, 1990, No. 11, pp. 1917–1921.

[14] Moore, T.: The Sybase IQ Survival Guide. Versions 12.6 through to 15.2. Lulu.com, 1st edition, 2011.

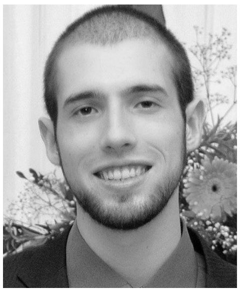
[15] Transaction Processing Performance Council. TPC-H Benchmark Specification. Retrieved from `http://www.tcp.org/hspec.html/`, 2017.

[16] Witten, I. H.—Neal, R. M.—Cleary, J. G.: Arithmetic Coding for Data Compression. Communications of the ACM, Vol. 30, 1987, No. 6, pp. 520–540, doi: 10.1145/214762.214771.

[17] Ziv, J.—Lempel, A.: A Universal Algorithm for Sequential Data Compression. IEEE Transactions on Information Theory, Vol. 23, 1977, No. 3, pp. 337–343.

[18] Ziv, J.—Lempel, A.: Compression of Individual Sequences via Variable-Rate Coding. IEEE Transactions on Information Theory, Vol. 24, 1978, No. 5, pp. 530–536.

[19] Zukowski, M.—Héman, S.—Nes, N.—Boncz, P.: Super-Scalar RAM-CPU Cache Compression. Proceedings of the 22nd International Conference on Data Engineering (ICDE '06), IEEE Computer Society, 2006, doi: 10.1109/ICDE.2006.150.



Vinicius Fulber Garcia received his Bachelor's degree from the Universidade Federal de Santa Maria (UFSM), Rio Grande do Sul, Santa Maria, Brazil. Currently he is a Master's degree student at UFSM. His research interests include low-level optimization techniques, data compression, network security and network virtualization.

Sergio Luis Sardi Mergen received his Ph.D. degree in computer science from the Database Research Group at the Universidade Federal do Rio Grande do Sul (UFRGS), Rio Grande do Sul, Porto Alegre, Brazil, in 2011. His thesis was on on-the-fly integration and querying of structured data sources available on the Web. He is currently Professor of computing science at the Universidade Federal de Santa Maria (UFSM), Rio Grande do Sul, Santa Maria, Brazil. His research interests include algorithms, data integration and data compression.

Computing and Informatics, Vol. 37, 2018, 424–456, doi: 10.4149/cai_2018_2_424

USE OF SELF-HEALING TECHNIQUES FOR HIGHLY-AVAILABLE DISTRIBUTED MONITORING

Włodzimierz Funika

AGH-UST, Faculty of Computer Science, Electronics and Telecommunication Department of Computer Science, al. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: `funika@agh.edu.pl`

Abstract. The paper addresses the self-healing aspects of the monitoring systems. Nowadays, when the complex distributed systems are concerned, the monitoring system should become "intelligent" – as the first step it can guide the user what should be monitored. The next level of the "intelligence" can be described by the term "self-healing". The goal is to provide the capability that a decision made automatically by the monitoring system should force the system under monitoring to behave more stable, reliable and predictable. In the paper a new monitoring system is presented: AgeMon is an agent based, distributed monitoring system with strictly defined roles which can be performed by the agents. In the paper we discuss self-healing in the context of monitoring. When the self-healing of the monitoring system is concerned, a good example is the case where it is possible to lose the monitoring data due to the storage problems. AgeMon can handle such problems and automatically elects substitute persistence agents to store the data.

Keywords: Monitoring, self-healing, distributed systems, reliability, high availability

Mathematics Subject Classification 2010: 68-M14, 68-M15

1 INTRODUCTION

Nowadays computer systems become more and more complicated. This statement is especially up-to-date when the distributed systems are concerned. The number of distributed systems is rapidly growing. A good example for this trend is cloud storages and cloud computing. A few years ago those terms were known only to a limited number of people who worked in IT industry or to scientists. Today, solutions which use clouds are available broadly for different kinds of end-users.

Complex distributed systems can be built with the components which cooperate together in order to achieve common goals. The monitoring of such components is especially challenging. Decomposition of a system results in a need in distributing the monitoring system itself. Moreover, the complexity of applications requires that the monitoring system provides some aspects of 'intelligence' – it should be possible to guide the user about what is the most optimal way of monitoring.

The most complex monitoring systems that are currently available are able to work in an autonomous way. It means that some or most of the operations are executed without user interaction. A monitoring system based on observations of an application can decide what action should be taken – for instance if any other monitoring should be performed or if a user interaction is required.

Similarly to *clouds*, terms like High Availability and Fault Tolerance are becoming very popular. The solutions which combine both High Availability and High Performance Computing (HPC) are becoming available for much more users [20]. This change drives changes in monitoring systems. The question is: *how can a Highly Available application be monitored?*

Increasing the complexity of a monitoring system results in a higher probability of faults in a system. In such a situation, the system should recover from a fault. In other words, it should be able to perform **self-healing**. Healing can be also considered from a perspective of an application. A good monitoring system in addition to a regular monitoring can provide a way of healing the application. Based on predefined rules, the system could take an action to help the application – for instance to restart its components or disconnect a failed resource.

There is a big number of monitoring systems available on the market. Unfortunately, the existing solutions do not provide all the features which are required by some of the modern applications. The existing monitoring systems:

- are sometimes used to monitor highly available applications or systems but they are not themselves highly available or fault tolerant,
- do not provide self-healing capabilities,
- are hard to deploy and complex to use,
- usually manifest problems when it comes to integration with applications (in order to heal the application).

In order to monitor a highly available application, monitoring system should also be highly available. This will minimize the risk of loosing important monitoring data gathered at system runtime.

The *main objective of our research* is to verify whether self-healing techniques can be used to build highly available and reliable monitoring systems needed for developing and maintaining highly available applications.

In order to achieve this goal, we introduce a new model of monitoring system. The model provides:

- *self-healing* capabilities which will help to implement high availability requirements; the system cannot loose any monitoring data gathered during a monitoring session,
- *distributed, loosely-coupled architecture* – the design of the system should be based on a distributed architecture, the system should be able to monitor distributed applications,
- *autonomicity* – it should be possible to define and deploy the rules and actions which will be executed automatically by the monitoring system,
- *capability of integration with an application* – the model should allow for integrating the monitoring system with an application in order to: heal it, or to provide monitoring data which can be used by an application in its regular functioning.

We have built a new monitoring system called *AgeMon* that was used to evaluate the proposed solution. This name will be used in further paragraphs of this paper.

The main objectives of the research are as follows:

- analyze different aspects of the High Availability and Self-Healing,
- evaluate if Self-Healing concepts can be used to provide High Availability solutions,
- select the best Self-Healing techniques which can be used in the monitoring systems,
- create a reusable and generic model of a Self-Healing monitoring system,
- create a prototype of monitoring system which will provide self-healing components. It should be possible to reuse these components in other monitoring systems.

The rest of the paper is organized as follows: Section 2 introduces the key concepts related to self-healing and reliability. Section 3 brings an overview of the challenges faced when monitoring the modern distributed systems. The motivation for a new monitoring system is introduced. Section 4 describes the architecture and implementation of the AgeMon system; later on, the HA and self-healing concepts in the context of monitoring systems. Section 5 presents the results of the system tests. The last section summarizes the paper.

2 RELATED WORK – RESEARCH BACKGROUND

There exist a considerable number of monitoring systems for distributed environments available on the market, some being more domain-adjustable like SemMon [22], while others are more specialized. Some of the monitoring systems are very well known and de-facto became industry standards, like Nagios [23] or Ganglia [24]. Various monitoring systems are described at the end of this section. We are going to start with introduction to some key concepts that are important to understand before deep diving into self-healing monitoring systems.

2.1 High Availability

The availability of a system determines if the system is able to provide the required service. When a service cannot be used by the user, it is said that there is an outage in the system. Downtime is duration of a time when the system is unavailable [17]. **Highly Available (HA)** system is designed to avoid losses of a service by reducing failures and downtimes of the system. System availability can be measured and is usually expressed as a percent of time when the system is available in a particular year. A system which provides 99.999 % percent of availability is considered as a high-availability system (the term *five-nines* is also used). The downtime of such a system should not take more than 5.5 minutes per year.

High Availability systems are reactive – emphasis is on a failover and a recovery [18]. In addition, *continuously available systems* group applications with a proactive approach. Such systems try to detect and prevent errors *in advance*.

2.2 Self-Healing

Self-healing is the ability of a system to recover from a failure state. Additionally, a self-healing system should be able to perceive that its own operation is not correct [8]. A healing action can be performed in an autonomous way or could require a user intervention (assisted-healing systems).

Self-healing [12] is also considered in the context of **self-managing autonomic systems**. In such systems human operator takes on a new role. He/she does not control the system in a direct way. Instead, the user defines general rules and policies that guide the self-management process.

The key questions when self-healing is concerned is whether the healing can be done automatically (*self*) or with a user interaction. In this paper, self-healing is understood as an action which should not involve any manual user interaction during the failure detection and recovery. Therefore, a system with the self-healing functionality should also be autonomous (self-healing components of the system should be autonomous).

An automaticity of the system does not exclude a user interaction, for instance in a system setup. While the failure detection and recovery should be completely automatic, human interaction may be needed to define high level rules for decision making, templates, to define data sources, properties, etc. [13, 25].

In Figure 1 a state diagram of a self-healing system is presented [8].

There are three states of the system from the self-healing perspective. The most desirable state is a normal, healthy state. The system in this state works correctly, and should fulfil all the requirements. Nevertheless, the system should periodically

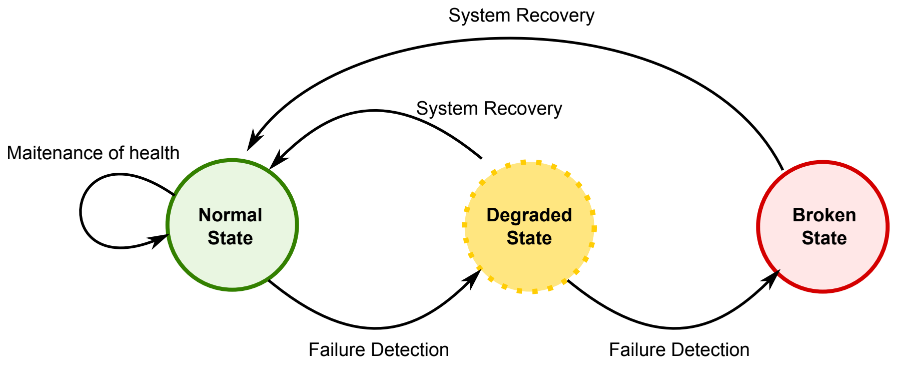


Figure 1. State diagram of the self-healing system

check its own state. It can be done by analyzing application logs. Additionally, the system needs to manage the state of its redundant components and maintain diversity – these tasks can result in failure detection, but the main intention is to keep the system healthy.

In addition to the tasks used to maintain a healthy system, there are tasks which are used to detect a failure in the system. There are multiple strategies here, for instance used to detect the missing components ([9, 10, 11]) or perform system monitoring [14].

A failure itself is a manifestation of an error caused by a system fault [3]. There is a number of classifications of errors. For instance, it is possible to classify software faults based on the circumstances needed to trigger an error [4, 5].

When a failure in the system is detected, the system is recognized as a broken one. Self-healing systems should be able to recover from this state, to heal themselves. One of the major concepts used here is the redundancy of components. With this approach, in case of a failure of a component, other components can take the responsibility of the failed component. It may decrease the performance of the system, but the system will continue to operate.

In loosely coupled environments, an additional problem is to detect a malicious fault, known also as the Byzantine Generals Problem. In order to solve such types of problems, a voting procedure should be implemented.

Other solutions for recovery are typically very specific to a particular system. Multiple attempts have been made to structure these problems. For instance, in one of the approaches, healing is done through cooperation between components. This type of recovery is becoming very popular [15, 16].

In addition to a normal and a broken state, a *degraded* state can be introduced. The system in this state can be considered working, but some of its functions could be limited. For instance, a performance indicator could be significantly degraded. If no action is performed immediately, the degraded state can be quickly turned into the broken state. As an example, let us consider an application which uses 99 % of memory. It is very probable that the system will fail soon, if no action to free the memory is performed.

2.3 Reliability

In general, software reliability is a probabilistic measure which can be defined as probability that software faults do not cause a failure during specified time [19]. From a mathematical point of view, it can be defined as the following function [19]:

$$R(t) = Pr\{T > t\} = \int_t^\infty f(x)\,\mathrm{d}x \tag{1}$$

where $R(t)$ = reliability, T = working time without failure, t = required (assumed/specified) working time without failure, $f(x)$ = failure probability density function.

From the industry perspective, there are two major metrics that are used to evaluate the reliability of a software:

- Mean Time Between Failures (MTBF) – elapsed time between failures in the system. This assumes that the system can be recovered (manually or automatically) from the failure (or the failure does not affect the overall functionality of the system).
- Failure Rate – frequency of the failures in the system.

The following equation defines correlation between MTBF and Failure Rate (λ):

$$\mathrm{MTBF} = \frac{1}{\lambda}. \tag{2}$$

Measuring the reliability is especially useful during the process of developing software. It can be used as a metric determining the current state of software and see if there are improvements during different phases of testing (unit testing, acceptance testing, integration testing, soak and stress testing).

2.4 Existing Monitoring Systems

Over years a considerable number of monitoring facilities were released. Below we give an overview of some representative monitoring systems.

2.4.1 Ganglia

Ganglia [24] is a scalable distributed monitoring system. It is deployed on more than 500 clusters over the world. It is designed to work in the high-performance computing like clusters and Grids. Its hierarchical design is based on the federation of clusters. Inside the cluster communication uses the multicast, while the communication between clusters within the federation is based on tree point-to-point connections.

Ganglia provides some aspects of high availability system. For instance, it is possible to specify multiple sources for the data in the `gmetad` components. It is used to failover in case one source in invalid. It is very scalable, but it does not provide advanced self-healing capabilities, and it is not possible to integrate this system with the monitored system in order to heal the system.

2.4.2 Autopilot

AutoPilot [26] provides an infrastructure for real-time, adaptive monitoring of distributed applications/systems. This monitoring tool can adaptively apply the data reduction based on the fuzzy logic. Autopilot allows also optimizing the application at runtime as the result of the application state observed by the monitoring system. This could also be used to *heal* the monitored system.

Autopilot is based on the Pablo Toolkit. It supports rule definition and some basic concepts of healing the application. It does not ensure high availability (it has a single point of failures) and does not provide any self-healing capabilities. It is also quite hard to use. The sensors and, especially, monitor tasks, need to be created from scratch for any new application. It does not support a common monitoring infrastructure like JMX.

2.4.3 GEMINI

GEMINI [27] is a Grid monitoring framework that fulfils space between resources monitoring components and monitoring services clients. GEMINI combines applications, infrastructure and Grid middleware resources monitoring by providing unified interfaces for data sources. As for monitoring, GEMINI performs measurements using a set of loadable modules called sensors which retrieve monitoring data by itself or using external, legacy applications for this purpose. GEMINI was developed within the K-Wf Grid Project but it is not intended to cooperate with this Grid system only; the idea behind providing a generic framework is that it could be easily adapted to various Grid environments. GEMINI is written in Java and is based on the Globus Toolkit libraries and services.

Monitors and sensors provide web-services which can be used by the clients to access the data. This solution allows developers to write extensions in an easy way. While GEMINI is very flexible, it does not allow for interactions with the monitored system. It is not a high availability system and it does not provide self-healing capabilities.

2.4.4 Aksum and JavaPSL

Aksum [28] is part of the Askalon [29] project, aiming to simplify the development and optimization of applications that can harness the power of Grid computing.

Aksum automatically searches for performance bottlenecks based on the concept of performance properties. In contrast to many existing systems, performance properties are normalized (values between 0 for the best case and 1 for the worst case), enabling the user to interpret the resulting performance behaviour.

Aksum is highly customizable, which allows the user to build or define an own performance tool. Performance properties are defined in JavaPSL [30], and may be freely edited, removed from or added to Aksum in order to customize and speedup the search process. The performance properties found can be grouped, filtered, and displayed in several dimensions as long as more experiment data become available.

The Askalon is focused on monitoring workflows of the applications deployed on the grid. It uses Globus infrastructure, therefore it is not possible to simply deploy this monitoring tool outside the grid. It does not provide an infrastructure to dynamically call the application logic, therefore its healing functionality is limited. It does not provide self-healing.

2.4.5 SemMon

The SemMon [22] is a monitoring system for distributed applications which enables adaptive monitoring. It is focused on monitoring distributed Java applications, but it can also be used to monitor OS specific capabilities.

The system is able to learn what is important to monitor in the current situation. The knowledge is gathered based on the previous user decisions. For instance, if the user decides that in the current state of the system, a specific capability needs to be monitored, the system will store this information, and use it in future to help the user or start the measurement automatically.

The definition of the monitoring capabilities (the capabilities which can be monitored by SemMon like CPU usage, number of threads) and metrics are expressed with the semantic description. They are described in the ontology, which brings an additional abstraction layer and could be used to improve the adaptation of the system. Based on the semantic dependencies between metrics, the system can automatically provide a hint to the user what needs to be monitored in the specific situation.

The SemMon system is a complete implementation of a robust system with semantics, which is not biased to any kind of underlying ‘physical’ monitoring system, giving the end-user the power of intelligent and computer-aided monitoring features like automatic metrics selection and collaborative work. On the other hand, it is not a high-available system. It has several single points of failures – e.g. reasoners or database with results. In addition, it does not provide any self-healing features.

2.4.6 Dynamic Monitoring Framework

Dynamic Monitoring Framework [32] is a solution which tends to address multiple challenges related to monitoring of the SOA based products. Service Oriented Architecture (SOA) is an architecture approach used in a wide area of solutions. It focuses on implementing business requirements as a service. One of the biggest advantages of such an architecture is the ability to support frequent changes of the requirements at runtime. For instance, it is possible to update a specific service without shutting down other services. It is also possible to change dependencies between services at runtime. While this approach addresses a lot of business challenges, it also adds complexity to the management and monitoring layers.

The Dynamic Monitoring Framework provides an interesting capability – ability to monitor a dynamically changing environment. Unfortunately, it does not provide any self-healing capabilities.

3 MONITORING SYSTEM MODEL

The model of monitoring system under discussion is designed to be based on the distributed, agent-based architecture. The distribution of the system enables the system to be more flexible and allows to dynamically fit into a complex *monitored* system. Since the monitored systems are often distributed, or even the monitoring system is used to monitor distributed values (like *network flow*) the distribution of the monitoring system becomes one of the most important requirements.

The distributed monitoring system usually consists of the following components: monitoring service/sensor and the user interface used to present the monitoring data to the user. It could be extended by additional components like database used to persist results or rule engines used for decision making based on the monitoring results.

One of the possible implementations of the distributed architecture may involve the agent-based approach [7]. This type of architecture brings a lot of values to the system. For instance, the scalability of an MAS system is provided very naturally. If one needs to have more resources in the system it is all about adding additional instances of agents. A similar situation is with concurrency. MAS systems are designed to be flexible and adaptable to the changing environment.

In addition, the attributes which are usually [6] discussed when MAS are concerned are *fault-tolerance* and *reliability*. The system does not have a single point of failures. The reliability is achieved collectively by all the agents of the system.

The main goal for the system model under discussion is not focused on the fully autonomous agents. Therefore, in the first implementation of the system, agents will have a limited extent of autonomy. Due to that limited autonomy, the reasoning coordination can be simplified. At the same time, in the AgeMon we address some of the common challenges with the agent based approach: communication between agents, cooperation or problem decomposition are the aspects that are implemented in the system. The agents are physically distributed, and need to cooperate in order to achieve the monitoring goals. Therefore the system under discussion is a *distributed and multi-agent* system.

3.1 Self-Healing, Distributed Monitoring System Architecture

The design of the monitoring system with the self-healing capability is considered at the very first stage. In fact, this requirement is the *driver* of the design.

There are two main requirements which need to be addressed by high-level design in order to provide self-healing of the monitoring system:

- redundancy of key components,
- high availability of the communication layer.

The redundancy of key components is the basic concept when the *high availability* feature is implemented. The redundancy is usually realized by physical or programmatic duplication of the resources/components in the system. This is a simple and straightforward approach used in different systems. Unfortunately, it has a huge disadvantage – it does not scale. For instance, if we have a system with 2 database components, and 2 oracles, addition of four more components to provide duplication would not impact the overall system performance. But in case there are 50 database components adding more 50 ones just to fulfil a redundancy paradigm could drastically degrade the system performance.

The redundancy of components is a key concept that needs to be introduced when the high availability is considered. Unfortunately, it is not enough – it is possible that the system consists of duplicated components, but these components cannot communicate with each other. Therefore, the communication layer between components/agents in the system should also feature high availability.

The design of the system under discussion should address these issues. The new monitoring system – AgeMon is intended to provide self-healing capabilities together with allocating the resources dynamically depending on the system state. The redundancy of the components should be available, but should only be used when no other way of providing high availability is feasible.

The AgeMon system is a distributed, agent based, self-healing monitoring system. The key concept in the system are *roles*– which are used to group functionality. For instance, the persistence role groups all the functionalities related to storing monitoring data in a persistent way. Use of this approach will simplify design and implementation and will also help define the best self-healing techniques for each role.

All the agents in the system are able to perform one or more roles. There are five types of the roles:

Monitoring Role – used to retrieve monitoring data and interact with the monitored system,

GUI Role – interacts with the system administrator and displays monitoring data,

CLI Role – enables advanced capabilities used by the system administrator,

Persistence Role – stores monitoring results in a persistent way (e.g. in a database),

Rule Role – dynamically reacts on changes in the system environment and performs actions based on the monitoring results.

To provide the dynamic self-healing capability in a scalable way, the roles can be automatically enabled or disabled by the agent in response to changes in the system. This is a very important aspect of the system – it is not required to have duplicated agents/roles since it is possible to start a new role only when needed. The communication between agents is based on a reliable messaging protocol. The stub of the protocol is provided by the Agent Communication Layer.

A sample deployment diagram is given in Figure 2.

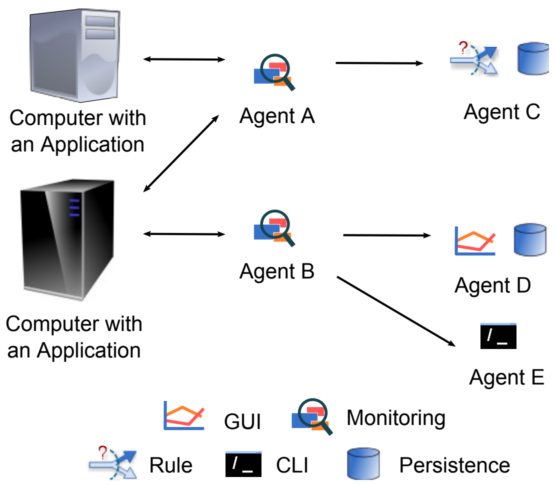


Figure 2. AgeMon System – a sample deployment

In this example scenario, two applications are going to be monitored. The arrows in the diagrams present the flow of the monitoring results which are sent between agents. There are two agents with the monitoring role enabled – **A** and **B**. Agent **B** is monitoring one application while the second agent is used to monitor two applications. The monitored data is sent to other agents. In the example, agent **C** is concurrently performing two roles – Rule and Persistence. It will be used to store the monitoring results in the database. At the same time the agent can perform user-define checks to test if the monitoring results should trigger an action. Agent **D** is also used to store the monitoring results in the database. In addition to this, it can present the monitoring data to the user. The agent **E** can be used by the system administrator to validate the state of the monitored system.

The above simple example illustrates some key-concepts of the system. It is possible that more than one role is performed by a single agent. On the other hand, it is possible that in the system there are more agents performing the same role. The agent is able to send monitoring results to more than one agent.

The next subsections provide a description of the Roles in the AgeMon System.

3.2 Monitoring Role

The monitoring role is used to retrieve the data from the monitored system and to *interact* with that system. Agents which perform this role act like sensors – they periodically ask for a new monitoring value and deliver it to the monitoring system. There are two types of measurements which can be done by the monitoring role: OS-related measurements and application specific measurements.

In the Java world, the Java Management Extensions (JMX) is a broadly accepted industry standard for monitoring and managing Java Applications [1, 2, 21]. This is not limited to the application – JMX can be used to monitor an Operating System, Networks or Devices. AgeMon utilizes this flexibility – support for JMX based monitoring is built-in. It is also possible to integrate it with other monitoring environments.

User is meant to select what he/she wants to monitor (select a *monitoring capability*) and create a monitoring task through a GUI. A monitoring agent will send a monitoring value to the destination agent(s) at the predefined intervals. Agent is also capable of *buffering* the results – it is possible to define the size of the packet used to send the data.

Whenever the monitoring data cannot be sent to the destination agent an election of a substitute agent (failover) will be triggered.

While probing the system under monitoring can be considered a fundamental task of the monitoring role – participation in handling the ‚feedback' procedure enables the *healing* of the monitored system.

Currently the system supports two *direct* ways of passing the information to the monitored system. With the first one the AgeMon system can call a method on an MBean. The MBean should be registered in the default platform MBean server. With the second way, the call can be performed on any static method defined by the user.

3.3 GUI Role

The GUI role is used to present monitoring data to the user. It is also meant to perform administrative tasks – for instance to define and deploy rules. The other rules provide important capabilities like persistence or healing, but it is possible to use the basic features of the monitoring system without them. On the other hand when the monitoring role and GUI role are not present, the system cannot operate in an efficient way.

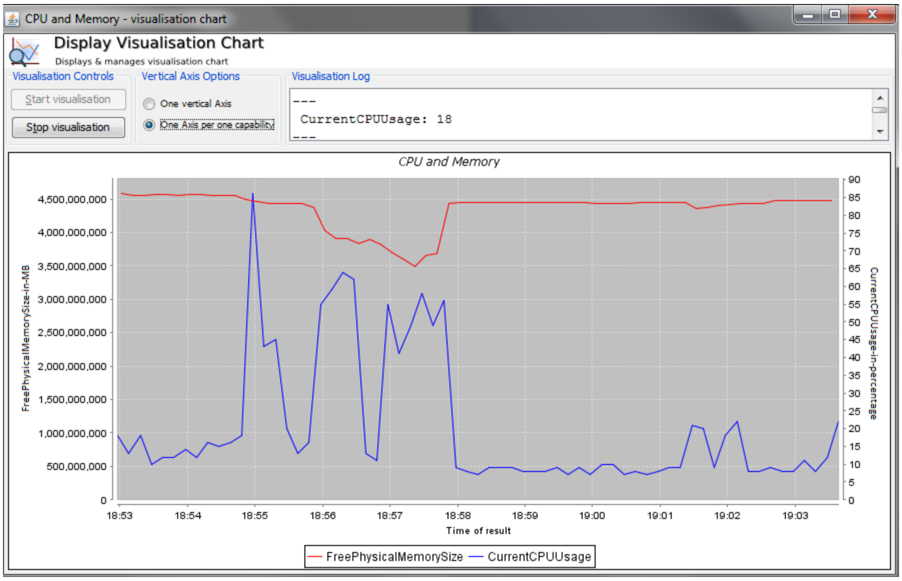
The central point of the GUI role is *Agent Graph.* It is a directed graph where each vertex represents an agent connected to a monitoring group, and each edge shows a monitoring connection created between two agents. It represents a publisher-subscriber dependency between two agents, therefore it is always directed. It is designed to execute a bunch of actions directly from a graph – for instance it is possible to stop an agent, create a monitoring and visualisation.

The *Monitoring Component* allows the user to select measurable capabilities, define a monitoring name or specify a polling interval. In addition, as a result of the creation of a monitoring task, a monitoring link between two agents is established.

The *Rule Component* provides an ability to create, manage and delete the rules. It can be only displayed in the context of the Rule Agent. A detailed description of the Rules is provided in Section 3.4.

Managing of the Persistence Agent can be done through the *Persistence Component*. The main functionality provided by this component is focused on enabling the browsing of the monitoring data in a straightforward way. It is possible to list measurement results and filter them, e.g. by a monitoring name.

The *Visualisation Component* allows the user to create and manage visualisations. It enumerates the available monitoring sessions and allows the user to create a new visualisation or attach to the existing one.



One vertical Axis [✓ One Axis per one capability]

```
---
CurrentCPUUsage: 18
---
```

Figure 3. Example visualisation – two measurements: CPU utilization and memory usage are presented on the same chart

A sample visualisation is presented in Figure 3. It presents an ability to display different monitoring results on one chart. In this example *CPU Usage* and the *Free Physical Memory* is presented at the same time. Due to it, the chart contains two different vertical axes – one per each capability. This component provides more advanced options like a dynamic scale, dynamic rendering, zooming or printing.

3.4 Rule Role

The Rule role introduces automation into the system. This is a very important feature, especially when complex, distributed systems are under monitoring. The user will not be able to capture all the issues, as well as to check all the results in every minute over a long period of time. Carefully designed and implemented rules will introduce a lot of new functionalities without additional cost.

A short definition of the *Rule* states that Rules are used to identify a change in the state of the monitoring and monitored systems. For instance, it can identify that a new Agent is attached. It can be also used to detect that CPU usage is higher than 70 %. This definition is not complete, and will be extended later in this section.

In order to *react* to the changes, a Rule is always connected with an appropriate *Action*. The action is executed by the Rule when the criteria defined by the Rule are met. There are multiple types of actions; most of them will be described later in this section.

The system provides two types of rules: Value Expression Rule and System Event Rule. The first one can be used to monitor the value of a selected capability, for instance to react if the CPU usage is higher than 90 %. It is possible to create a rule condition using Java Script language.

Another type of rules is the System Event Rule which allows to react on the changes occurring in the Monitoring System. For instance, it is possible to monitor whenever an agent is attached or detached to/from the system.

In order to react, AgeMon provides five different actions out of the box (it is also possible to define new actions). The first, simplest action is a *Console Action*. When this action is triggered, the message will be logged to a log file. A more complex action is *Email Action*. When it is triggered, an email message will be sent to a predefined email address.

The next action available in the AgeMon system provides the ability to run an external command. When an *ExecuteScript* action is defined, the user can provide the name of a script and path with the location of the script.

The next two actions are used to directly interact with the monitoring system. They allow for execution of a specified code from the *monitored system* – depending on the action's type, the processor will execute a static method or invoke a method on an MBean.

Typically, a rule is deployed in a single Rule Role. This is the most common situation, but AgeMon provides a second type of rule deployment called the *cooperative* mode which brings the *high availability* functionality. In this mode, the rule is deployed in multiple Rule Agents. Agents are able to synchronize the evaluation and execution of the rules, so the action will be executed only once. The following assumptions for the cooperative mode are listed below:

- All the Rule Roles Agents have a running instance of the Cooperative Rule (**CR**). *Running* means that the rule is executed against the monitoring data on each of the agents.
- When the rule condition is met, only ONE role executes a Cooperative Action (**CA**). The election algorithm will be used to select an agent which can perform a **CA** (the algorithm is similar to the one used in finding the substitute agent).
- If the **CA** fails due to the internal reasons (e.g. not because of the exceptions thrown from the monitored system) a next agent will try to re-execute a **CA**.
- All the roles have to have access to the monitoring data used in a definition of the condition. This means that whenever a rule is deployed, the additional monitoring links need to be created automatically.
- If a new rule agent joins the network, the **CR** will be automatically deployed on it and the corresponding monitoring links will be created.

3.5 Persistence Role

The Persistence Role is used to store the monitoring results in a persistent way. By default, this Role will use an in-memory database. Due to this reason, it is not needed to install any third-party applications in order to enable persistence.

3.6 CLI Role

The CLI Role is the second role used by the user to interact with the Monitoring System (and System Under Monitoring). This role is complementary to the GUI – it will provide additional capabilities for the advanced user, while some of the other features will only be available in the regular GUI. For instance, with the CLI it will not be possible to create a visualisation instance, whereas it is meant to enable creating customized queries used to export the monitoring data.

3.7 Communication in the System

Messaging is the key concept of the system. Agents interact with each other by exchanging the messages. The system uses a state-of-the-art communication library which is built by analogy to a stack. The bottom layer introduces a discovery of all the agents in the same group. Multicast as well as gossip server(s) can be used to detect members of a group. The low-level communication protocol is also implemented in this layer. All the messages are exchanged in a reliable way. The system allows to use point-to-point communication as well as point-to-many. In the second approach, a message is sent with the UDP/multicast protocol. Thanks to that messages are sent in single operation to all agents. This reduces the network usage in contrast to plain TCP protocol. At the same time, additional components built on top of protocol ensure reliability (e.g. failure detection, retransmissions etc.). Due to the discovery protocol, it is possible to start the system in a typical environment without any additional configuration.

The second layer of the communication library brings an agent abstraction which simplifies sending the messages to the agent. The agents' topology is reflected in an object-oriented fashion. An agent object contains methods to send messages as well as to subscribe for possible notifications.

The top layer defines the monitoring capabilities of the system. As mentioned above, the communication between agents is based on exchanging the messages: there are more than 40 types of them, gathered into two groups: foundation messages and regular messages.

The library is highly extensible, it is allowed to define new types of messages as well as corresponding processors. The library can also be used in other systems, since it is not specific to the system under discussion.

3.8 Logging in the System

Having a mechanism for correct logging of the events in the application is an important feature of any modern systems. It can be used to validate the state of the system as well as to track the issues that occurred during runtime.

AgeMon uses the log4j library, by default logging only important messages that are presented on the console. It is possible to enable a very detailed logging and redirect it to files. In order to reduce performance degradation when tracing is enabled, the logging system uses an asynchronous appender. It is also possible to rotate logs. Nevertheless in case a full, detailed logging is enabled, it may have a significant impact on the agent performance. During the conducted tests, when the TRACING level of the logger is enabled, CPU utilization was up to 50 % higher than when the default logging settings are used. Tracing is very detailed – it is possible to see the details of each message that is exchanged as well as all connections that are being made on the TCP/UDP level.

In addition to the regular logging, it is possible to easily integrate the system with more *centric* logging environments that can be used to profile the performance of the system. We have successfully used `logstash` (for pushing data), `elastic` (as data storage) and `kibana` (presentation & BI layer) to store and analyse AgeMon's performance.

4 HIGH AVAILABILITY AND SELF-HEALING

The AgeMon was designed as a self-healing, high availability monitoring system. It is used as a proof-of-concept to verify which techniques can be used to build self-healing monitoring systems. Below we outline the key-features implemented in AgeMon which enable high availability and self-healing.

4.1 Automatic Discovery

The automatic discovery of all the agents in the group has an impact on the high availability. It is not required to have any kind of central registry where all agents have to be registered – therefore there is no single point of failure. The automatic discovery can be also considered as one of the self-healing techniques. Agent, which is detached from a group due to a network issue, is able to automatically re-attach to the group when the issue is resolved.

4.2 Reliable Transport Protocols

The protocols used by the AgeMon system are *reliable*. This means that the transport layer will notify other layers that the message has been *or* has not been delivered successfully to the recipient. The successful delivery means that the recipient sent back the ACK information in order to confirm that the message had been received. The reliability of the protocol provides also additional features like preserving the correct order of messages or enforcing the coherency of messages.

4.3 Network Failures Tolerance

The AgeMon system can operate when there occurred a network failure. The first line of defence is the ability to cache the messages in the designated buffer on the agent side. If the monitoring results cannot be sent to the destination agent, the results will be stored locally and a second phase of resolving the problems will be triggered – finding a substitute agent (it will be described later in this section).

4.4 Lack of Single Point of Failure

The single point of failures is usually the biggest pain point which prevents the system from being highly available. There are two aspects of dealing with single points of failure:

- *avoid* the problem by introducing the correct design decisions. The Agent-based approach guarantees that the system is decentralized. All the agents are homogeneous, except the roles. On the other hand, the roles can be dynamically started in order to resolve the issues in the system.
- add *redundancy* for the components. Whenever it is not possible to avoid the problem, all important components should be duplicated.

4.5 Roles Redundancy

Where it is not possible to avoid the single point of failure, one of the solutions is to enhance redundancy in the system.

AgeMon is by its nature agent-based, and all the agents are the same – the only difference between the agents are the roles which they play. There are multiple ways of achieving the redundancy in the AgeMon System. The first one is quite simple and is presented in Figure 4.

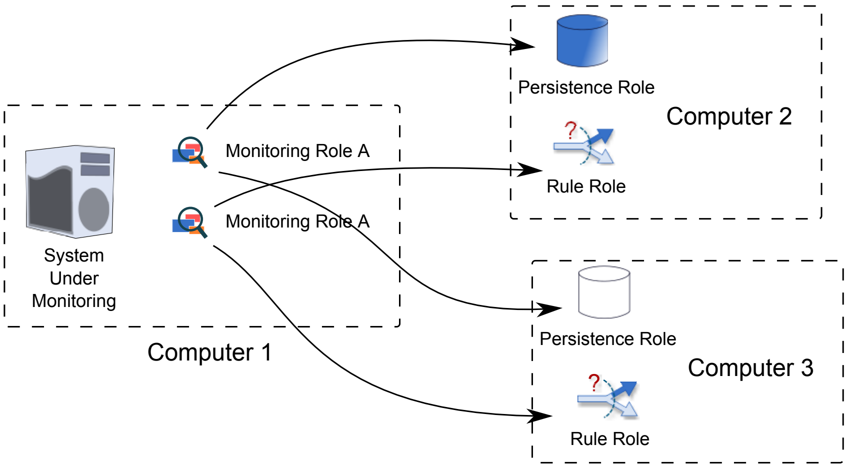


Figure 4. Redundancy of key components in AgeMon system

In this example, AgeMon is used to monitor the OS specific values. All the agents are duplicated – there are two monitoring agents, two persistence roles and two rule roles. The agents are running on different computers in case of a physical failure.

While AgeMon supports duplication of the agents, it has several disadvantages. The complexity of deployment is very high in case of multiple monitoring agents, and requires a lot of redundant connections. It requires a significantly higher network bandwidth. Therefore this type of deployment is not recommended and in most situations it is not needed.

AgeMon provides much more sophisticated solutions which enable high availability deployment – for instance it can automatically respond on changes in the system and add the redundancy 'on-the-fly' by electing the substitute agents. This approach is described in the next subsection.

4.6 Substitute Agents – Failover

The Agents Substitution is a key feature implemented in AgeMon, which enables high availability in the Monitoring System. It is used to elect the substitute agent that will persist the data in the database while the original destination agent is down. Figure 5 depicts an example usage.

In this example (Figure 5 a)), the user wants to monitor two OS capabilities: CPU Usage and Memory Usage. After some period of time *Agent 2* is killed (Fig-

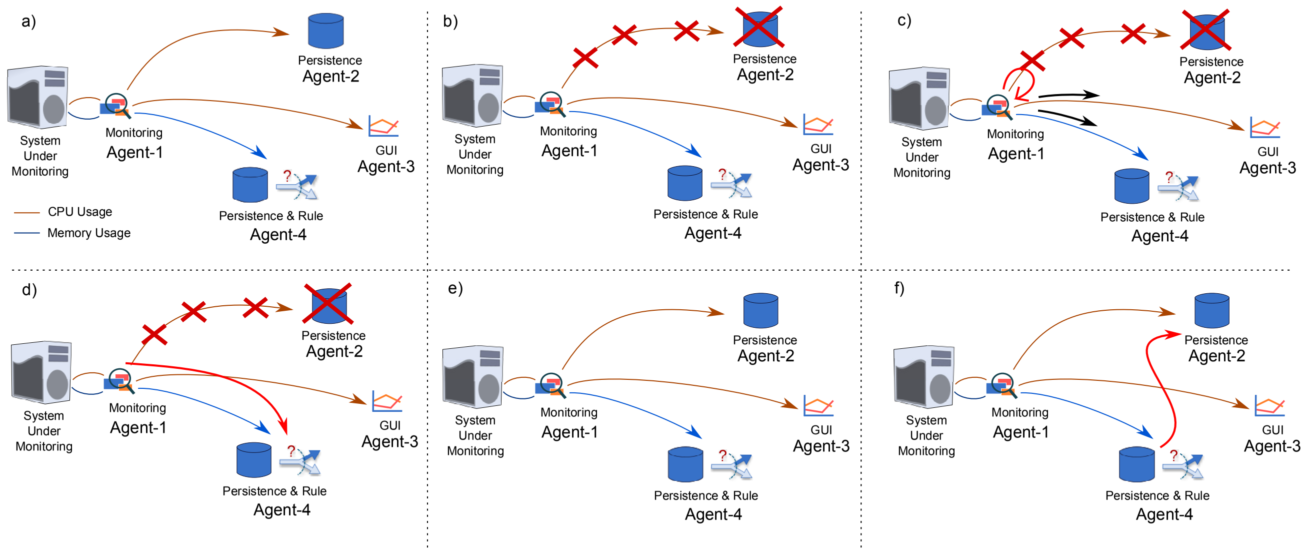


Figure 5. Overview of the election algorithm

ure 5 b)). The system detects such a situation and starts the failover procedure which consists of two steps.

In the first step (Figure 5 c)) the monitoring agent creates a *local buffer* where the monitoring data will be stored until the problem is resolved. This prevents from losing any monitoring data.

The second step is focused on finding the *substitute agent* that can handle the monitoring data. The *election algorithm* is used in order to fulfil this task (it is possible to use different election *policies*). The failover monitoring link will be automatically created and the monitoring data will be sent to the selected substitute agent (Figure 5 d)).

At the same time, the rule agent was configured to monitor the state of the monitoring agent. Since Agent 2 was killed, it sends an email to the administrator. After some time, Administrator was able to free some space, and restart the persistence agent (Figure 5 e)). The failover link will be deleted, and all the new monitoring data will be sent directly to Agent 2.

In the last step of the failover/substitution algorithm the monitoring data stored in the Substitute Agent (Agent-4) is transferred to the original destination agent (Figure 5 f)). After this step, Agent 2 contains the complete and coherent monitoring data.

The substitute agents are an important piece of the AgeMon system. Owing to the election algorithms, the system can provide high availability capabilities. It can also be considered as a self-healing strategy. While this technique does not lead to a complete system recovery, it helps with resolving the problem. The monitoring data will never be dropped if at least one agent can function.

4.7 Advanced Rules – Self-Healing

The rules can be triggered if the state of the Agent Group is changed, for instance, when one of the agents is down. This simple rule can be used to define complex actions and enable *self-healing* of the monitoring system. For instance, the system can detect that one of agents is detached. Based on the log entries, the system can decide that the agent is down due to the system restart, and try to start it again (e.g. by executing a remote command on the server).

This is one of examples – of course, since there is no limitation for external scripts, there are plenty of possible self-healing actions, starting with restarting the network adapter to solve connectivity issues between agents, to ending with removing the logs to gain free space. All these cases can be already allowed for AgeMon.

4.8 High Availability/Self-Healing Strategies – A Summary

The roles performed by the agents have different requirements from the high availability perspective. For instance, we can use redundancy to setup the Monitoring

Role. The Persistence Role can also be setup redundantly, but it will not be efficient from the disk usage perspective. A similar problem is when the Rule Role is concerned – the same rule can be executed on multiple agents in parallel, but the action can be triggered only on one agent.

Due to these reasons for each Role type, a different High Availability strategy should be used:

Monitoring Role – for this type of the role, the best strategy is to combine both agent *redundancy* and *self-healing* actions. The redundancy of the agent's instances will decrease the probability of failure. Even if such unexpected situation occurs, the self-healing rules and actions can restart the problematic agents.

Rule Role – the *cooperative* rules should be used for the high availability environment. These rules can be executed/evaluated in parallel. The AgeMon system will ensure that the action will be performed only by one instance of the agents.

Persistence Role – the approach with *Substitute Agents* is the best when this type of the role is considered. In case of any type of failure (agent failure, connection failure) the AgeMon system will elect the substitute agent which will be used to persist the data. This will prevent from losing any monitoring results.

GUI Role and CLI Role – these types of the roles do not need to have a separate high availability mechanism. They are not storing important data and do not have any automated processing which should be parallelized.

Due to the roles approach used in AgeMon it is possible to develop different high availability algorithms for different requirements. As is evident, each rule has a different nature of operating and requires a different treatment in order to enable high availability. Splitting the functionalities into the roles encapsulates them and provides a good stub for implementing specific high availability algorithms.

5 SYSTEM TESTS

The first part of this section that is dedicated to testing the AgeMon System is focused on High Availability. We will verify if the system can be used if some of its components fail. These types of tests are referred to as destructive tests. We will use them to validate if the self-healing techniques used in the AgeMon system are good enough to ensure High Availability.

In the second part of the section, performance tests are presented. These types of tests are used to determine the responsiveness and stability of the system under different values of load. The performance tests deliver information about the scalability or reliability of the system. They are followed by a description of the tests which were used to verify what is the minimum time in which the system can react to changes in the monitoring system.

5.1 Self-Healing Tests/High Availability Tests

In order to verify the High Availability of the AgeMon, two types of tests were executed. The first group consists of a set of destructive tests. These tests attempt to cause a failure in some components (e.g. random components, critical components) in order to verify the robustness and reliability of the system. The second type of the tests which were executed is the soak test. It is used to verify if the system works correctly under significant load over a significant period of time.

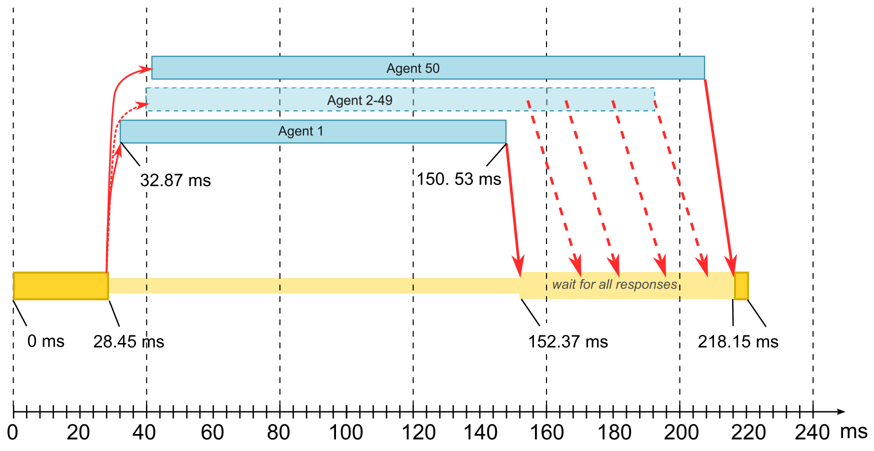


Figure 6. Process of the election of a substitute agent

In the **first test**, a system that consists of 50 agents was under observation. A half of the agents were executing monitoring scenarios, while others were used to persist results. The configuration of the agent allows all agents to perform a persistence role.

The test simulates a failure in the persistence agent. The self-healing mechanism implemented in the AgeMon should detect this situation and try to elect a substitute agent which can be used to persist monitoring results until the original agent is available.

The election procedure (Figure 6) starts when a monitoring agent is not able to send a message to a destination agent. Owing to the Agent Communication Layer, each agent possesses the list of the active agents. If the destination agent is not present, the election procedure is started (the election procedure is also started if there was a failure when sending the message). At the same time all the messages with monitoring results are stored by the agent in a local cache. The messages will not be lost as long as there is a free memory space. A single message typically needs 1–2 kB – it means that it is possible to keep half a million messages with a 1 GB heap size.

At the moment of 28 ms after the failure was detected, a request to start an election process is sent to all the agents in the group. The agent processes the request, and based on the election policy it responds with the information which will be used to elect an agent from the group. The monitoring agents are waiting for all responses (or until a predefined time elapses). An average response time was between 120 and 180 milliseconds. After all the data is collected, the agent elects a substitute agent and advertises this information to all agents. All messages cached in the monitoring agent are sent to the elected agent.

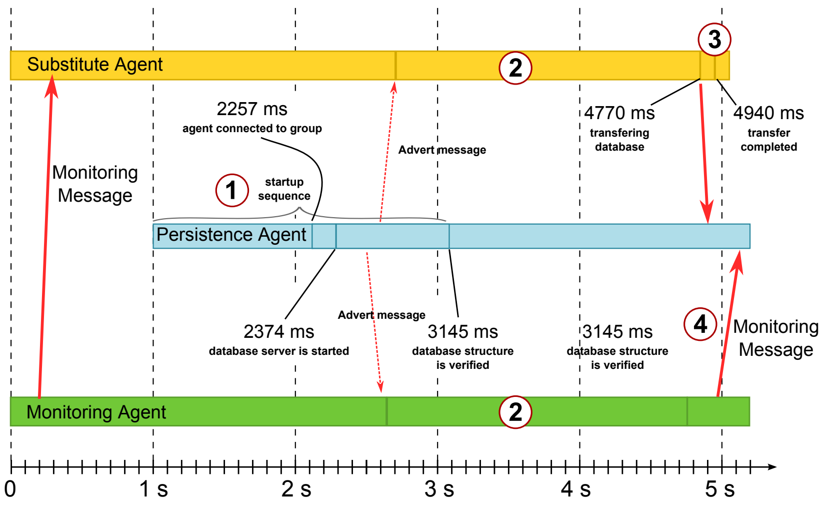


Figure 7. Operations performed after Persistence Agent is recovered

The **second test** scenario covers a situation when Monitoring Agent failed (e.g. due to the machine restart) and needs to be restarted. It presents the capability of the monitoring system to auto-detect failures in the agent group and the flexibility of a rule agent to execute a customized script.

For this scenario (Figure 7), a group of 20 agents is prepared. A half of the agents are performing monitoring, 9 ones are persisting data and one is used as a rule agent. There is one rule deployed in the system – this rule will execute an external script in case one of the agent is detached. The script is used to connect to the remote sever where the Monitoring Agent is installed and will try to execute the restart script.

During the test, Monitoring Agent was killed with `kill -9` command. After about 100 ms, Rule Agent was notified by the Transport Layer about a missing agent. During the next 140 ms the agent was executing a bash script. Monitoring Agent was started after additional 50 ms. Therefore, the total time when the agent was down is **295 ms**. After additional 170 ms all agent's services were started again, and the agent was reconnected to the group.

5.2 Reliability

The total number of the messages sent during all tests is higher than 40 million. There were different tests conducted – we tested the system under regular load, then we performed a stress test of the system. After that the destructive tests were conducted followed by the soak test.

In all these tests, we verified the reliability of the system (together with testing other non-functional requirements like performance or scalability). These tests are very promising – there was no single monitoring message missing. The system could work with reduced performance (e.g. when load was too high) but we did not observe any dropping messages. The tests evidence that transport layer is reliable – all messages were delivered. In case there was a problem with network, the message was not acknowledged – and it was retransmitted again.

There are only two cases when isolated messages can be lost. The first one can occur when there is a failure in a monitoring agent before the message was broadcasted to the other agents. The second situation happens when there is a failure of the persistence agent during message processing (but before it is persisted in the database). The theoretical solution for this problem assumes that the persistence agent should acknowledge to the monitoring agent the fact that the message is persisted. If there is no ACK message, the monitoring agent can try to:

1. retransmit the message, or
2. start an election procedure to find a substitute agent.

The drawback of both solutions is that it could reduce the overall performance of the system (additional synchronization in the monitoring agent, additional message in the communication layer). Therefore, we did not decide to implement it.

Fortunately, the system architecture allows avoiding both situations. For instance, to avoid losing the messages in the monitoring agent we proposed duplication of those agents. The same idea applies to the persistence agents. Of course this type of the deployment should only be considered when persisting all messages is critical. The conducted tests have proved that losing a message is very unlikely to happen.

There was no single message lost in all the tests – all messages (10 M during destructive tests and 32 M during soak tests) were successfully delivered.

5.3 Performance Tests

To test the performance of AgeMon, a sample testing environment was prepared. In this setup, there are multiple monitoring agents (the exact number depends on the specific test scenario) installed on multiple-machines. They are monitoring CPU usage. The monitoring data is sent to one persistence agent – the interval between each send operation (we call *delay*) is defined in a scenario.

By measuring the total time of the test and comparing it with the expected time, it will be possible to calculate overheads. If the overheads are high (cannot be explained by network delays or serialization/deserialization of the messages) it means that the persistence agent was not able to process so many requests.

In all the test scenarios the following servers/computers were used (connected with Gigabit Ethernet):

- 5 × SunOS 5.10, 48 GB of memory, 2 × Intel® Xeon® CPU X5650 @ 2.67 GHz (later referred to as Sun),
- 5 × Linux RedHat 5.5 (Tikanga), 30 GB of memory, 24 × Intel® Xeon® CPU X5650 @ 2.67 GHz (later referred to as Linux),
- 1 × Windows 7 Enterprise, 16 GB of memory, 1 × Intel® Core™CPU i5-3527U @ 2.30 GHz (4 cores) (later referred to as Win7).

During the test some principal performance metrics were gathered like CPU Usage, Memory Usage, Threads Count, and Garbage Collector executions. The system was monitored with a second instance of the AgeMon system running on different ports with a different agent group. This second 'monitoring' system instance was built with multiple monitoring agents and one persistence agent. The monitoring data was extracted through a GUI agent.

Some 12 tests with different numbers of agents and messages were performed in order to measure the differences between 'ideal processing' and the observed performance. We assume here that more than 20 % difference between 'ideal processing' and the actual performance should be considered unacceptable.

The results were confronted with the 'ideal processing'. The ideal processing was calculated based on the number of messages and the delay between sending the messages. It does not include any network delays or time used by the agent to persist the data. Therefore, the ideal processing cannot be reached in a live system. On the other hand, in a high performance system, the results should be close to the ideal ones.

In order to verify if the performance of the Persistence Agent is correlated with the number of agents, different types of configurations were used. Figure 8 presents the results of the tests. The figure depicts the differences between the ideal situation and the observed results (the difference is marked by pink and red colour). We can draw multiple conclusions:

- The performance is directly related to the number of messages to be processed. A huge amount of messages results in longer processing delays of each message.
- Persistence Agent can process 950 messages with acceptable performance. Tests with more than 950 messages show a performance degradation (the difference between 'ideal processing' and the observed performance was more than 20 %).

If AgeMon is used in an environment with a large number of messages (greater than 1 000 messages per second), the performance may be unacceptable. In order to resolve this issue, the user has two possible ways: multiply the number of the persistence agents or deploy a persistence agent in the environment with a more powerful CPU.

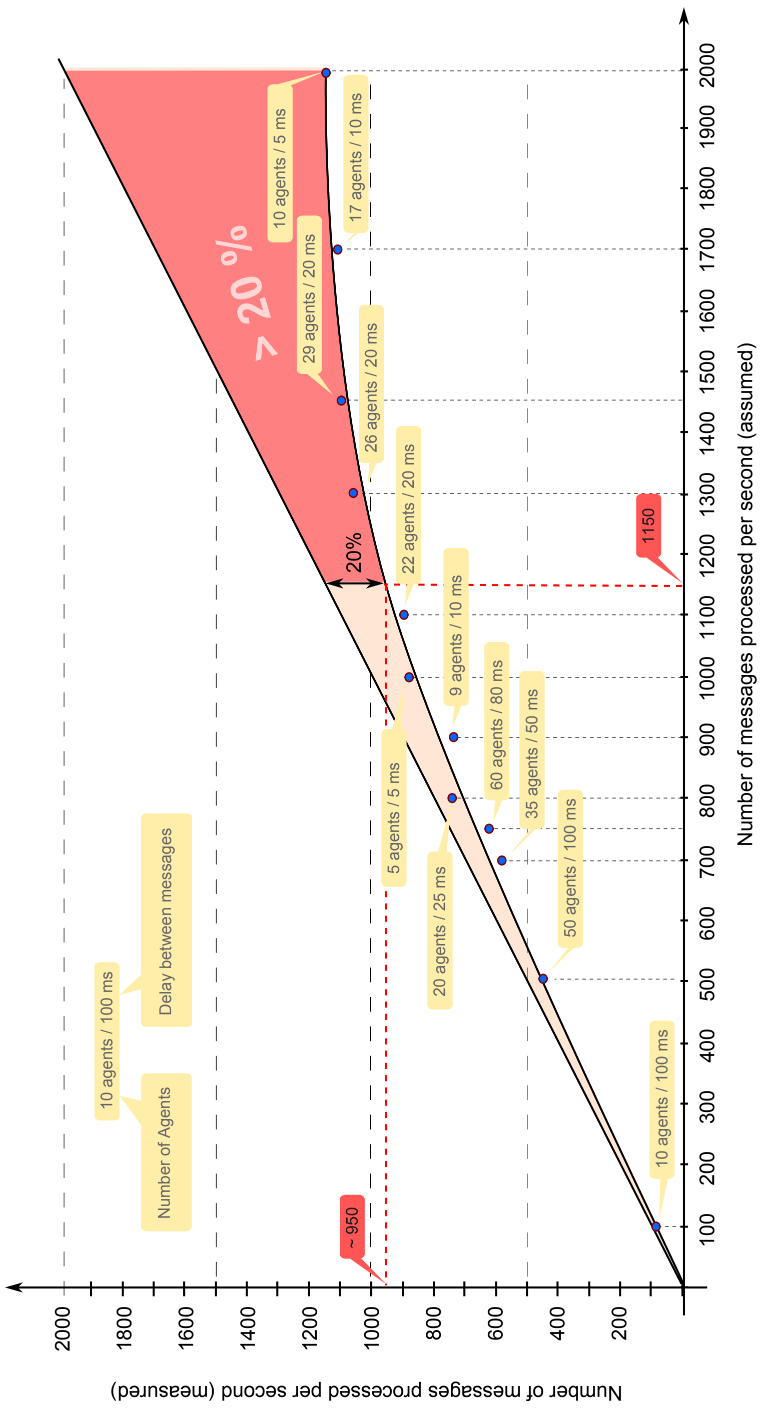


Figure 8. Performance of Persistence Agent

The tests results show that the system can be used to persist a big number of messages per second. One persistence agent can successfully process over 1 000 messages per second – it should be enough for most of the monitoring cases.

In order to verify the *scalability* of AgeMon, some additional tests were performed. In this test scenario a large number of agents were used. The first cycle of the test started with one persistence agent and 20 monitoring agents. The monitoring agents send messages to the persistence agent every 25 ms. In the next cycle the number of persistence agents and monitoring agents is increased. For each cycle, the performance of the system is measured. The results are given in Figure 9.

The AgeMon system can scale horizontally – as described above, the CPU could be regarded as the biggest bottleneck. Therefore, in order to increase the number of messages that can be processed by the system, the user may consider adding new agents running on dedicated physical machines.

System can scale to a greater extent than that presented in Figure 9, but due to the resource limitations we were not able to conduct more extensive tests. In order to estimate if the system can be used in a deployment scenario with a significantly greater number of agents instantiated we measured the overhead of the messages that were not used to exchange the monitoring data.

From the high-level perspective, the agents exchange two types of messages. The first group consists of messages used to transfer monitoring data between agents (in a typical case among the monitoring agents and persistence agents). Another type of messages is used to synchronize, detect failures and keep up-to-date the group of agents. These messages are vital to keep the group of the agents together within a single monitoring system.

Fortunately, the overhead generated by these messages is almost negligible. In our tests, these messages were responsible for 3 % of the network traffic, and 8 % of CPU. Based on that, we can draw a conclusion that the system should be able to handle at least 12 times more agents (up to 1 200) in a single group compared to the maximum number of agents used in the tests.

5.4 Latency in the System

Below we discuss two test scenarios which were executed to measure the latencies in the AgeMon system. The first one is quite interesting – it answers the question: "How long it takes the system to make a decision based on the state of the System Under Monitoring?" The second test is important as well, since it is used to check when the monitoring data is persisted.

The average decision time (Avg_{dt}) is one of the most important aspects of AgeMon. It defines the time which is required to take a decision. This factor describes what the maximum speed of changes in the monitored system is, which can be successfully detected and allows the system to make a correct, healing decision.

If the monitored system is faster than Avg_{dt}, the monitoring system may not be able to make a correct decision. An action made by the system could not be accurate to the prevailing conditions. In the test scenarios the observed decision in the system was performed **36 milliseconds** after a change in the measured capability had been observed.

In the next scenario, the persistence time was measured. The goal was to measure the average time between receiving the monitoring value and storing this information in the monitoring agent. The configuration in this scenario is similar to what was introduced in the previous test. The only difference is that a persistence agent is substituted for the rule agent. In this scenario the time needed to commit will be measured.

The tests show that the Monitoring Agent requires 33 milliseconds to send a message to the Persistence Agent. After additional 11 milliseconds the message is committed to the database by the Persistence Agent. The full process took 47 milliseconds.

The system can operate with millisecond-long latencies. The system requires 36 ms to make a decision and 47 ms to commit a result into the database. Prior to defining the rules, network tests need to be performed (e.g. with the `PING` command) in order to measure the latency.

The biggest contribution to the latency is caused by the message serialization. It is planned to be improved in the next versions of the system (the message can be constructed in a binary form instead of the use of the standard Java Serialization mechanism).

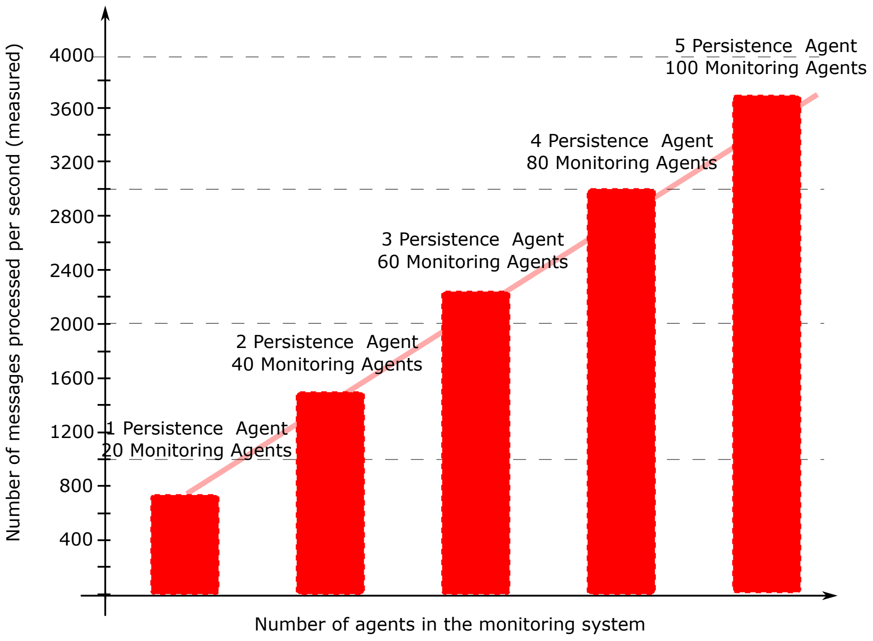


Figure 9. Scalability in the AgeMon system

6 CONCLUSIONS

The main goal for our research was to design a model and evaluate the algorithms and techniques which can be used to build a highly available monitoring system. In the paper a set of approaches has been described. Our work focuses also on the practical verification of the requirements. In order to do this, a new monitoring system called AgeMon was developed. This system uses an agent based, distributed approach. Each agent can perform different roles in the system. A role groups similar functionalities.

Role is an important concept to build a *highly available*, *self-healing system*, since different functionalities need different strategies to solve the problems related to high availability. The Persistence Role in the system uses substitute agents to persist data when one of the agents fails. The Rule Role can cooperate with other roles to provide a reliable way of executing the actions. The Monitoring role can be dynamically restarted in case of a failure.

The model lacks any single point of failure. The communication between agents is based on a reliable and fast protocol and provides auto-discovery. Due to it, there is no need to have a gossip/rendezvous server. Each component in the system could be redundant, but it is not a "must".

The system is a proof of concept for a more generic model. This model can be applied to build high availability systems (it does not apply only to the monitoring systems). Such concepts like roles, communication layer, and agent abstraction can be successfully applied in other systems.

The *monitoring mechanism* used by the AgeMon is very flexible. By default it can be used to monitor OS and any Java application. If such an application uses JMX to provide monitoring information it can be automatically used by the AgeMon. For any other type of monitoring, an appropriate adapter needs to be developed. In order to implement such an adapter, a single interface needs to be implemented.

An important part of the monitoring system is providing **visualisation** of the results. Charts in AgeMon support a single or multiple sources. Owing to this it is possible to compare two different measurements (e.g. CPU or memory) on a single chart. It is possible to zoom, scroll and print charts.

The system supports multiple types of *rules* and *actions*. AgeMon provides built-in rules like "agent attached", "agent detached" and rules based on the monitored value. It is also possible to create a custom rule by providing an implementation of a particular interface in Java. Similarly, some number of built-in actions is available (console, email, execute script, execute static call) while other actions can be implemented by the user.

The system is fully *distributed*. The system supports different types of deployment – for instance it is possible that a large number of agents are installed on a single computer. It is also possible to have one agent per physical machine. Agents can be distributed across a local network. The distribution over WAN or Internet is also supported. In this case the tunnelling and gossip servers need to be used.

The Persistence Role provides the *persistency* of monitoring results. AgeMon provides an in-memory, relational database out of the box (HSQL). If needed, other SQL or NoSQL databases [31] can be used.

The system can be *extended* and *integrated* with the monitored application or other monitoring systems in a straightforward way.

The use of AgeMon is very facile. It does not require any configuration or sophisticated *deployment*. The system is built inside one file (a jar file) and no installation is required. In the default configuration (default agent group, in memory database) no changes to the configuration is needed – the user can start agents which will automatically discover themselves and build the monitoring system.

To summarize, it is possible to build a high-available monitoring system. The designed self-healing approach can be very helpful in achieving this goal.

Novel concepts introduced. The AgeMon system is one of the first designed and implemented monitoring systems with self-healing capabilities. It is also a highly available system; it does not have a single point of failure and supports the redundancy of all its components.

The model used to build the system, which is based on roles, can be used to build other self-healing systems. This approach can be applied to other types of applications, so it is not limited only to the monitoring systems. The self-healing strategies defined in the model can be successfully used for other types of the components.

One of the most important achievements of our work is the evaluation of a number of HA and self-healing strategies. Different problems require different solutions, the role-based approach helps with identifying similar functionalities with the same healing strategies.

Additionally, the communication model designed for the system can be adopted and used in many other applications with distributed components. It supports auto-discovery, reliable and efficient communication. The model is built with three different layers – a physical communication layer, an agent layer, and a monitoring layer. The monitoring layer can be easily adapted to support other types of messages, specific to a system under design.

Acknowledgment

The Author is indebted to several people, first of all to Dr. Piotr Pęgiel for valuable help and to Mr. Paweł Koperek for inspiring discussions. The research presented in this paper received financial support from AGH University of Science and Technology Statutory Project.

REFERENCES

[1] Perry, J. S.: Java Management Extensions: Managing Java Applications with JMX. O'Reilly, 2002. ISBN 0-596-00245-9.

[2] Fleury, M.—Lindfors, J.: JMX: Managing J2EE with Java Management Extensions. Sams Publishing, 2002. ISBN 0-672-32288-9.

[3] Laprie, J. C. (Ed.): Dependability. Basic Concepts and Terminology. Springer-Verlag, New York, 1992, doi: 10.1007/978-3-7091-9170-5.

[4] Gray, J.: Why Do Computers Stop and What Can Be Done About It? Technical Report 85.7., Tandem Computers, 1985, pp. 17–19.

[5] Rentschler, M.—Kehrer, S.—Zangl, C. P.: System Self Diagnosis for Industrial Devices. Proceedings of 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2013), September 10–13, 2013, doi: 10.1109/ETFA.2013.6648019.

[6] The Complex Adaptive Systems (CAS) Group: Multi-Agent System. `http://wiki.cas-group.net/index.php?title=Multi-Agent_System`, 2011.

[7] The Intelligent Software Agents Lab. `http://www.cs.cmu.edu/softagents/intro.html`.

[8] Ghosh, D.—Sharman, R.—Rao, H. R.—Upadhyaya, S.: Self-Healing Systems – Survey and Synthesis. Decision Support Systems, Vol. 42, 2007, pp. 2164–2185, doi: 10.1016/j.dss.2006.06.011.

[9] George, S.—Evans, D.—Marchette, S.: A Biological Programming Model for Self-Healing. First ACM Workshop on Survivable and Self-Regenerative Systems (SSRS '03), 2003, pp. 72–81, doi: 10.1145/1036921.1036929.

[10] Aldrich, J.—Sazawal, V.—Chambers, C.—Nokin, D.: Architecture-Centric Programming for Adaptive Systems. Proceedings of 1st Workshop on Self-Healing Systems (WOSS '02), 2002, pp. 93–95, doi: 10.1145/582128.582146.

[11] Dabrowski, C.—Mills, K. L.: Understanding Self-Healing in Service Discovery Systems. Proceedings 1st Workshop on Self-Healing Systems (WOSS '02), 2002, pp. 15–20, doi: 10.1145/582128.582132.

[12] IBM. The IBM Autonomic Computing Initiative: `http://www-03.ibm.com/systems/z/os/zos/features/sysmgmt/autonomic/index.html`.

[13] Raz, O.—Koopman, P.—Shaw, M.: Enabling Automatic Adaptation in Systems with Under-Specific Elements. Proceedings 1st Workshop on Self-Healing Systems (WOSS '02), 2002, pp. 55–60, doi: 10.1145/582128.582139.

[14] Blair, G. S.—Coulson, G.—Blair, L.—Duran-Limon, H.—Grace, P.—Moreira, R.—Parlavantzas, N.: Reflection, Self-Awareness and Self-Healing in OpenORB. Proceedings 1st Workshop on Self-Healing Systems (WOSS '02), 2002, pp. 9–14, doi: 10.1145/582128.582131.

[15] Georgiadis, I.—Magee, J.—Kramer, J.: Self-Organizing Software Architectures for Distributed Systems. Proceedings 1st Workshop on Self-Healing Systems (WOSS '02), 2002, pp. 33–38, doi: 10.1145/582128.582135.

[16] de Lemos, R.—Fiadeiro, J. L.: An Architectural Support for Selfadaptive Software for Treating Faults. Proceedings 1st Workshop on Self-Healing Systems (WOSS '02), 2002, pp. 39–42, doi: 10.1145/582128.582136.

[17] Weygant, P. S.: Clusters for High Availability: A Primer of HP Solutions. Prentice Hall, 336 pp., 2001. ISBN 978-0-13089-355-0.

[18] Richards, M.: The Secret to Bulding Highly Available Systems. No Fluff Just Stuff, Vol. 2, 2011, No. 5. `http://wmrichards.com/ha.pdf`.

[19] Goel, A. L.: Software Reliability Models: Assumptions, Limitations, and Applicability. IEEE Transactions on Software Engineering, Vol. 12, 1985, pp. 1411–1423, doi: 10.1109/TSE.1985.232177.

[20] Longbottom, C.: Make a High-Performance Computing and High-Availability Datacentre. Computer Weekly, 2013. `http://www.computerweekly.com/feature/Making-your-datacentre-fit-for-high-performance-computing-and-high-availability`.

[21] Oracle: Java Management Extensions (JMX) Technology. `http://www.oracle.com/technetwork/java/javase/tech/javamanagement-140525.html`.

[22] Funika, W.—Godowski, P.—Pegiel, P.— Król, D.: Semantic-Oriented Performance Monitoring of Distributed Applications. Computing and Informatics, Vol. 31, 2012, No. 2, pp. 427–446.

[23] Nagios Web Site: `https://www.nagios.org/`.

[24] Ganglia Web Site: `http://ganglia.sourceforge.net/`.

[25] Shehory, O.—Martinez, J.—Andrzejak, A.—Cappiello, C.—Funika, W.—Kondo D.—Mariani, L.—Satzger, B.—Schmid, M.: Self-Healing and Recovery Methods and Their Classification. In: Andrzejak, A., Geihs, K., Shehory, O., Wilkes, J. (Eds.): Proceedings Dagstuhl Seminar 09201 "Self-Healing and Self-Adaptive Systems", May 10–15, 2009, Dagstuhl, Germany, Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, Germany, 2009. ISSN 1862-4405.

[26] Ribler, R. L.—Vetter, J. S.—Simitci, H.—Reed, D. A.: Autopilot: Adaptive Control of Distributed Applications. Proceedings of the Seventh International Symposium on High Performance Distributed Computing, July 1998, doi: 10.1109/HPDC.1998.709970.

[27] Baliś, B.—Bubak, M..—Łabno, B.: GEMINI: Generic Monitoring Infrastructure for Grid Resources and Applications. Proceedings of the Cracow '06 Grid Workshop "The Knowledge-Based Workflow System for Grid Applications", ACC Cyfronet AGH, Krakow, 2007, pp. 60–73.

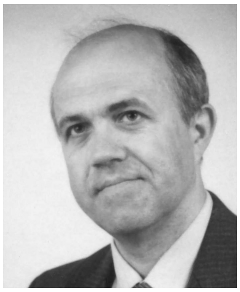
[28] Fahringer, T.—Seragiotto, C.: Automatic Search for Performance Problems in Parallel and Distributed Programs by Using Multi-Experiment Analysis. In: Sahni, S., Prasanna, V. K., Shukla, U. (Eds): High Performance Computing – HiPC 2002. Springer Verlag, Lecture Notes in Computer Science, Vol. 2552, 2002, pp. 151–162.

[29] Fahringer, T.—Jugravu, A.—Pllana, S.—Prodan, R.—Seragiotto, C. Jr.—Truong, H.-L.: ASKALON: A Tool Set for Cluster and Grid Computing. Concurrency and Computation: Practice and Experience, Vol. 17, 2005, No. 2-4, pp. 143–169, Wiley, Inc., 2005.

[30] Fahringer, T.—Seragiotto, C.: Performance Analysis for Distributed and Parallel Java Programs. IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005) 2005, `http://dps.uibk.ac.at/local/conferences/ccgrid/2005/pdfs/115.pdf`.

[31] Gąsiorowska, K.—Trybała, D.—Funika, W.: Storage of Monitoring Data in NoSQL Database. Procedia Cracow Grid Workshop '16, Krakow, ACC CYFRONET AGH, Krakow, 2016, pp. 39–40. ISBN: 978-83-61433-20-0.

[32] Żmuda, D.—Psiuk, M.—Zieliński, K.: Dynamic Monitoring Framework for the SOA Execution Environment. Proceedings of International Conference on Computational Science (ICCS 2010). Procedia Computer Science, Vol. 1, 2010, No. 1, pp. 125–133.



Włodzimierz Funika works at the Institute of Computer Science of the AGH University of Science and Technology in Krakow (Poland). His main research interests are in distributed programming, tools construction, performance analysis and visualization, machine learning. Involved in EU Cross-Grid, Core-GRID, K-WfGrid, ViroLab, GREDIA, UrbanFlood, MAPPER, VPH-Share projects as well as in Polish-wide projects PL-Grid, PLGrid PLUS, PLGrid Core, and PLGrid NG.

Computing and Informatics, Vol. 37, 2018, 457–475, doi: 10.4149/cai_2018_2_457

MODELLING OF DIAGNOSTICS INFLUENCE ON CONTROL SYSTEM SAFETY

Karol Rástočný, Juraj Ždánsky, Mária Franeková

Faculty of Electrical Engineering, University of Žilina
Univerzitná 8215/1, 010 26 Žilina, Slovakia
e-mail: {karol.rastocny, juraj.zdansky, maria.franekova}@fel.uniza.sk

Iveta Zolotová

Faculty of Electrical Engineering and Informatics, Technical University of Košice
Letná 9, 040 00 Košice, Slovakia
e-mail: iveta.zolotova@tuke.sk

Abstract. If the control system besides the standard control functions also realizes the functions (known as safety functions), failures of which can influence safety of the controlled process, then the control system may be a source of risk for assets, that are within the scope of the controlled process. Early detection of these failures and subsequent negation of their effects can have a significant influence on the safety integrity level of the safety function and thus also on the elimination of risks related to the controlled process. Therefore, the diagnostics is the means which, if appropriately applied, can increase not only the availability, but also the safety of the control system. The paper deals with using the homogeneous Markov chains to influence the evaluation of on-line diagnostics on the hardware safety integrity of the safety function, depending on the application method of several simultaneously operating diagnostics mechanisms and their basic parameters – the failures diagnostic coverage coefficient and the failure diagnostics time.

Keywords: Safety integrity, safety function, diagnostics, analysis, Markov process

1 INTRODUCTION

There are many cases in industry, when the controlled process can be a source of a significant danger that can result in personal injury, environmental damage or other undesirable consequences. If the risk related to the controlled process is bigger than the acceptable risk, then it is necessary to use appropriate measures to minimize this risk at least to the acceptable risk level [1, 2]. One of the technical measures is also the use of safety functions (SF), that are implemented by the safety related control system (SRCS). SF is function (risk reduction measure), that is intended to achieve or maintain a safe state for the equipment under control (EUC), in respect of a specific hazardous event [1].

From a safety point of view it is important to detect and negate any dangerous failure as soon as possible (negation – enforcement of a safe state following the detection of a failure). As the dangerous failure (in this paper) is considered the failure that causes the SRCS transition into a dangerous state or increases the probability of the SRCS transition into a dangerous state. The failure detection and subsequent negation of the failure consequences have significant influence on the safety integrity level (SIL) of the SF [1, 12, 13]. Therefore, the SRCS generally contains on-line functional diagnostics in addition to on-line test diagnostics (the SRCS is automatically on-line tested), especially if SFs are implemented with the SIL3 or SIL4 [4, 15].

To analyze the failures consequences on the hardware safety integrity, the methods that were originally intended for the analysis of the reliability – RBD (Reliability Block Diagram) and FTA (Fault Tree Analysis) are very often used. However, these methods do not allow a complex influence assessment of multiple properties of the SRCS on the hardware safety integrity of the SFs, which are realized by the SRCS. From this point of view, it is more appropriate to use for example the methods using the continuous-time Markov chain (CTMC), either alone [3, 14], or in a combination with the discrete-time Markov chain (DTMC) [6]. Also it is possible to use other methods, for example methods based on the Petri nets [7, 8, 9].

The publications, dealing with the failures on-line diagnostics influence on the hardware safety integrity, generally consider only one failure detection mechanism that operates continuously in time [5, 9, 15]. The failure detection mechanism, which operates discreetly in time, is considered only in case of periodic maintenance or repair of the system [4, 9, 10, 11, 16]. However, in practice it is possible that the SRCS contains more mechanisms for the failures detection, which vary by their parameters and the character of the activity. This paper deals with the analysis of the simultaneous operation of several mechanisms of the failures detection and their influence on the hardware safety integrity of the SF. The usage of the proposed method is presented on the SRCS with dual structure based on composite fail-safety with fail-safe comparison.

2 MARKOV CHAINS

Let us consider a stochastic process that fulfills the Markov property

$$\Pr\left\{X\left(t\right) \le x \mid X\left(t_0\right) = x_0, X\left(t_1\right) = x_1, \ldots, X\left(t_n\right) = x_n\right\}$$
$$= \Pr\left\{X\left(t\right) \le x \mid X\left(t_n\right) = x_n\right\}$$

where $X(t)$ is the random variable, $t \in T$ (T is the time range) is the time parameter and is valid, that $0 \le t_0 < t_1 < \cdots < t_n < t$.

If the value, which is acquired by $X(t)$, is called state and if the set of states is countable, then Markov process forms the Markov chain (MC). We distinguish two basic types of the MC:

- discrete-time Markov chain (DTMC);
- continuous-time Markov chain (CTMC).

The MC can be homogeneous or nonhomogeneous. In this paper a premise is accepted that the considered MC are homogeneous.

For the homogeneous DTMC the transition probability of the system from state i to state j can be calculated as the conditional probability, that the system in time $t = n + 1$ goes to state j, under the condition, that the system in time $t = n$ was in state i, i.e.

$$p_{ij} = \Pr\{X_{n+1} = j \mid X_n = i\}. \tag{1}$$

The homogeneous DTMC is completely defined, if the transition matrix (2) and the initial distribution (3) are defined.

$$\mathbb{P} = (p_{ij}) \text{ for } i, j \in \{1, \ldots, m\}, \tag{2}$$

$$\overrightarrow{P_0} = \overrightarrow{P_0(t=0)} = \{p_1(t=0), p_2(t=0), \ldots, p_m(t=0)\} \tag{3}$$

where $\overrightarrow{P_0}$ is the initial distribution at time $t = 0$, $p_i(t = 0)$ is the probability of state i at the time $t = 0$. The DTMC distribution in time $t = k + 1$ for $k \in \{0, \ldots, n\}$ is

$$\overrightarrow{P_{k+1}} = \overrightarrow{P_k} \cdot \mathbb{P}. \tag{4}$$

For the homogeneous CTMC the transition probability of the system from state i to j state can be calculated as the conditional probability, that the system in time $t = t + \Delta t$ goes to state j, under the condition, that the system in time t was in state i, i.e.

$$P_{ij}(t + \Delta t) = \Pr\{X(t + \Delta t) = j \mid X(t) = i\}. \tag{5}$$

The homogeneous CTMC is completely defined, if the transition rate matrix (6) and the initial distribution (3) are defined.

$$\mathbb{Q} = (q_{ij}) \text{ for } i, j \in \{1, \ldots, m\} \tag{6}$$

where q_{ij} is the transition rate from state i to state j and $q_{ii} = -\sum_{j=1,j\neq 1}^{m} q_{ij}$ is the sojourn rate in state i. If the CTMC is homogeneous, the transition rates are constant.

The CTMC distribution in time t can be calculated as a solution of the differential equations system (7) for the initial distribution (3)

$$\frac{\overrightarrow{\mathrm{d}P(t)}}{\mathrm{d}t} = \overrightarrow{P(t)} \cdot \mathbb{Q}. \qquad (7)$$

3 GENERAL VIEW ON THE FAILURES DIAGNOSTICS

The SRCS can contain one or more failure detection mechanisms. The failure detection mechanism can be characterized by the failure detection time t_d and the diagnostic coverage coefficient c. For the hardware safety integrity evaluation the diagnostic coverage coefficient of the dangerous failures is relevant

$$c_D = \frac{\lambda_{dD}}{\lambda_D} \qquad (8)$$

where λ_{dD} is the dangerous detectable hardware failure rate and c is the diagnostic coverage coefficient and λ_D is the dangerous failure rate.

In general, it is valid, that $\lambda_D = k \cdot \lambda, k \leq 1$. If the value k cannot be exactly proved for the application, it is necessary to choose the value k in accordance with the requirements of the relevant standards for the applications area.

If the SRCS contains one failure detection mechanism, this mechanism in principle can work by a schedule where the failure diagnostics operates:

- periodically and discreetly in time – always at the end of the diagnostic cycle (Figure 1 a)), while $t_{cd} \gg t_{td}; t_{cd}$ is the diagnostic cycle time (it can be identified with the maximum time of the failure detection) and t_{td} is the operation time of the failure detection mechanism (the testing time); or
- periodically and continuously in time (Figure 1 b)).

If the SRCS contains two failure detection mechanisms, it is necessary to assume, that these mechanisms differ from each other by the failure detection time and the diagnostic coverage of the failures.

Let the system contains two failure detection mechanisms:

- the "rapid"detection mechanism (RM), which is characterized by the detection time t_{Rd} and the diagnostic coverage coefficient of the dangerous failures c_{DR};
- the "slow"detection mechanism (SM), which is characterized by the detection time t_{Sd} and the diagnostic coverage coefficient of the dangerous failures c_{DS}.

The Figure 2 shows the influence of these two diagnostics mechanisms on the overall diagnostic coverage of the dangerous failures. It shows, that

$$\lambda_D = \lambda_{uD} + \lambda_{dD\_R} + \lambda_{dD\_X} \qquad (9)$$

where λ_D is the dangerous failure rate, λ_{uD} is the undetectable dangerous failure rate, λ_{dD_R} is the dangerous failure rate, which are detectable by the RM, λ_{dD_X} is the dangerous failure rate, which are detectable only by the SM.

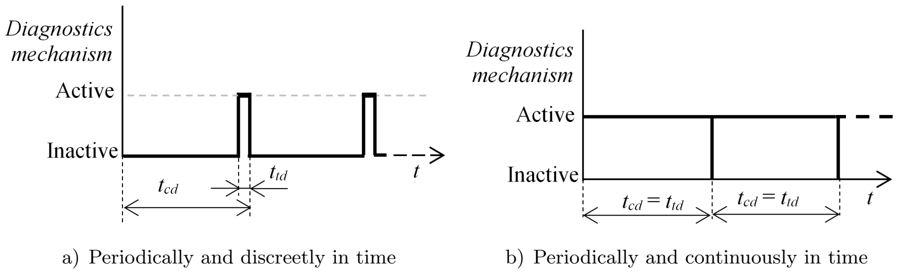


Figure 1. Operation of the failure detection mechanism

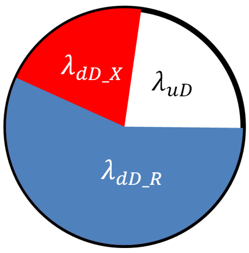


Figure 2. Diagnostic coverage of failures – two failure detection mechanisms

The probability of the failure detection by the SM, which was not detected by the RM, can be calculated according to the equation

$$Px = \frac{c_D - c_{DR}}{(c_D - c_{DR}) + (1 - c_D)} = \frac{c_D - c_{DR}}{1 - c_{DR}} \tag{10}$$

where c_D is the diagnostic coverage coefficient of the dangerous failures, which are detectable by the RM or SM and c_{DR} is the diagnostic coverage coefficient of the dangerous failures, which are detectable by the RM.

In reality it is necessary to assume, that some of the dangerous failures are covered by both failure detection mechanisms and also that there can be a part of failures, which are not covered by any failure detection mechanism. In general, the diagnostic coverage of the failures covered by the SM can be significantly lower than the diagnostic coverage of the failures covered by the RM, because the SM may be intended for detection of certain failures, which are not detectable by the RM.

Generally, it is possible to say, that:

- $c_D > c_{DR}$; if $c_D = c_{DR}$, it does not make any sense to apply the SM and $P_X = 0$;
- if $c_{DR} < 1$ and at the same time $c_D = 1$, then $P_X = 1$.

If the SRCS contains more failure detection mechanisms, a simplified method can be also used as the parameters estimation of the failure diagnostics, based on pessimistic premise that

$$c_D = \max\{c_{Di}\} \text{ for } i = \in \{1, \dots, n\},$$
$$t_{cd} = \max\{t_{cdi}\} \text{ for } i = \in \{1, \dots, n\} \qquad (11)$$

where c_{Di} is the diagnostic coverage coefficient of dangerous failures, which are detectable by i^{th} failure detection mechanism; t_{cdi} is the maximum time of the dangerous failure detection, which is detectable by i^{th} failure detection mechanism and n is a number of failure detection mechanisms.

4 THE HARDWARE SAFETY INTEGRITY OF THE DUAL STRUCTURE

In practice, SRCS with dual structure based on composite fail-safety are often used with fail-safe comparison. The standard [1] requires to realize the hardware safety integrity evaluation not for the system, but individually for each SF. Due to clarity reason of this paper it is assumed that the SRCS comprises two hardware identical and physically independent units – unit R and unit L (Figure 3), which control the EUC. Let both these units participate in realization of one SF and each unit may contain several elements – for example sensors, logic, actuators. On this premise, the dangerous state of the SRCS can be identified with dangerous failure of the SF.

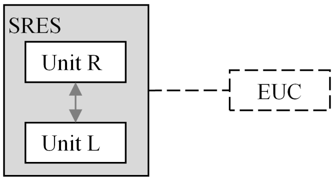


Figure 3. Block diagram of a general dual structure

In general, the SRCS with this structure has the RM, which is based on mutual data comparison of the units R and L after each operation cycle (comparative mechanism) and in many cases also the SM, which is focused on the failures, which are not detectable by the comparative mechanism.

As it is the dual structure with identical units, it is valid:

$$\lambda_L = \lambda_R = \lambda \tag{12}$$

where λ_L is the hardware failure rate of the unit L and λ_R is the hardware failure rate of the unit R.

5 THE DANGEROUS FAILURE PROBABILITY OF THE SF

In general, the SF can be performed in the low demand mode of operation or in the high demand mode of operation (in continuous mode of operation) [1].

If the SF operates in the continuous mode, then as the dangerous state of the SRCS is considered the state, which terminates the ability to realize its SF in compliance with the safety requirements specifications. In this case, the analysis of the SRCS failure consequences ends when the dangerous state is reached – it is necessary to identify the dangerous state of the SRCS with the dangerous state of the EUC.

In this paper it is considered, that the SF operates in the continuous mode and occurrence of the electronic elements failures can be regarded as the continuous random process, which follows the exponential distribution law. It is also taken into account the pessimistic assumption, that

$$\lambda_D = \lambda, \qquad c_D = c, \qquad c_{DR} = c_R, \qquad c_{DS} = c_S \tag{13}$$

where c is the overall diagnostic coverage coefficient of the failures, $c_R(c_S)$ is the diagnostic coverage coefficient of the failures covered by the RM (SM).

A. Influence of One Diagnostics Mechanism – Continuous Mode of Operation

If the method based on the MC is used for the hardware safety integrity evaluation, then for the dual structure (Figure 3) with hardware identical units can be used the CTMC, which is shown in Figure 4.

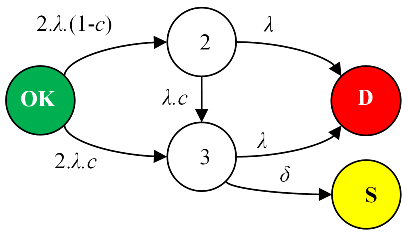


Figure 4. The Markov model for the dual structure with the time-continuous failure detection – continuous mode of operation and with one failure detection mechanism

It is a simplified approach to the hardware safety integrity evaluation, which can be used just when the SRCS has the failure detection mechanism. This failure detection mechanism operates continuously in time and the SRCS operates in the continuous mode.

The characteristic of the states in the model in Figure 4 is listed in Table 1.

State	Characteristic
OK	SRCS is functional; neither one unit has the failure.
2	Unit R or unit L has only the undetectable failures (one or more).
3	Unit R or unit L has the detectable failures (one or more); units can have also the undetectable failures (one or more).
S	The safe (dysfunctional) state – the state after detection and negation of the failure. The EUC is in state, which is not dangerous.
D	The dangerous state – both units have the failure.

Table 1. States of the model in Figure 4

The SRCS can go from the state OK to the dangerous state D on a trajectory, which depends on the sequence of the failures occurrence and their detectability (detectable or undetectable).

The characteristic of the transitions in the model in Figure 4 is listed in Table 2.

Transition	Characteristic
OK $\rightarrow$ 2	Transition is realized, if the undetectable failure occurs in unit L or unit R.
OK $\rightarrow$ 3	Transition is realized, if the detectable failure occurs in unit L or unit R.
2 $\rightarrow$ D	Transition is realized due to the failure occurrence in the unit (L or R), which is without failure.
2 $\rightarrow$ 3	Transition is realized due to the detectable failure occurrence in the unit, which already has the undetectable failure.
3 $\rightarrow$ D	Transition is realized due to the failure occurrence in the unit (L or R), which is without failure.
3 $\rightarrow$ S	Transition is realized due to the detection and negation of the failure occurrence.

Table 2. Transitions in the model in Figure 4

The transition rate from the state 3 to the state S can be expressed by the equation

$$\delta = \frac{1}{t_d/2 + t_N} \tag{14}$$

where δ is the failure detection and negation rate, t_d is the failure detection time (in this case $t_d = t_{cd}$, where t_{cd} is the duration time of one diagnostic cycle) and t_N is the time needed to the detected failure negation. Using the mean value of the failure detection time is not accurate, but acceptable in practice [1, 2]. In case of the pessimistic approach, the transition rate from the state 3 to the state S can be expressed by the equation

$$\delta = \frac{1}{t_d + t_N}. \tag{15}$$

CTMC in Figure 4 can be described by the transition rate matrix (16) and the differential equations system (17):

$$\mathbb{Q} = \begin{pmatrix} -2\lambda & 2\lambda \cdot (1-c) & 2\lambda \cdot c & 0 & 0 \\ 0 & -\lambda \cdot (1+c) & \lambda \cdot c & 0 & \lambda \\ 0 & 0 & -\lambda - \delta & \delta & \lambda \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{16}$$

$$\begin{aligned} p'_{OK}(t) &= -2\lambda \cdot p_{OK}(t), \\ p'_2(t) &= 2\lambda \cdot (1-c) \cdot p_{OK}(t) - \lambda \cdot (1+c) \cdot p_2(t), \\ p'_3(t) &= 2\lambda \cdot c \cdot p_{OK}(t) + \lambda \cdot c \cdot p_2(t) - (\lambda + \delta) \cdot p_3(t), \\ p'_s(t) &= \delta \cdot p_3(t), \\ p'_D(t) &= \lambda \cdot p_2(t) + \lambda \cdot p_3(t). \end{aligned} \tag{17}$$

If in the time $t = 0$ the SRCS is in the state OK (Figure 4), then the initial vector

$$\overrightarrow{P_0(t=0)} = \{1, 0, 0, 0, 0\}, \tag{18}$$

and the dangerous state probability [5]

$$\begin{aligned} p_D(t) = e^{-2\lambda.t} - 1 &+ \frac{2\delta}{(\lambda \cdot c - \delta) \cdot (1+c)} (e^{-\lambda \cdot (1+c) \cdot t} - 1) \\ &- \frac{2\lambda^2 \cdot c}{(\lambda \cdot c - \delta) \cdot (\lambda + \delta)} (e^{-(\lambda+\delta) \cdot t} - 1). \end{aligned} \tag{19}$$

If $c = 0$, then

$$p_D(t) = 1 - 2e^{-\lambda \cdot t} + e^{-2\lambda \cdot t}. \tag{20}$$

If $c = 1$, then

$$p_D(t) = \frac{e^{-2\lambda \cdot t} \lambda \cdot (\delta - \delta \cdot e^{2\lambda \cdot t} + \lambda + \lambda \cdot e^{2t \cdot \lambda} - 2\lambda \cdot e^{-(\delta - \lambda) \cdot t})}{(\lambda + \delta)(\lambda - \delta)}. \tag{21}$$

At the latest in time, when the probability value of the state D achieves the critical limit (the maximum allowed value related to the acceptable risk), it is necessary to terminate the SRCS operation and execute the proof-test. The influence of the proof-test on the hardware safety integrity of the SRCS is described in [9, 10, 11, 16].

If the SRCS operates in the continuous mode and the failures diagnostics operates periodically and discreetly in time – always at the end of the diagnostic cycle (Figure 1 a)), then the diagnostics influence on the hardware safety integrity of the SRCS can be modelled using the multi-phase Markov model – combination of the CTMC and the DTMC.

The failures influence on the hardware safety integrity of the SRCS in time, when the failure diagnostics mechanism is not active, can be described by the model in Figure 5. The failures occurrence is continuous in time, but the failure detection is not possible and therefore the transition from the state 3 to the state S is not possible. The state S in the model in Figure 5 is mentioned only by the reason of representation of a compatibility with the model in Figure 4.

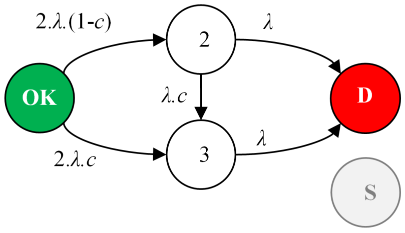


Figure 5. The reduced CTMC model for the dual structure with the time-continuous failures detection – continuous mode of operation and with one failure detection mechanism in time, when it is not active

The model in Figure 5 can be described by the differential equations system

$$\begin{aligned} p'_{OK}(t) &= -2\lambda \cdot p_{OK}(t), \\ p'_2(t) &= 2\lambda \cdot (1-c) \cdot p_{OK}(t) - \lambda \cdot (1+c) \cdot p_2(t), \\ p'_3(t) &= 2\lambda \cdot c \cdot p_{OK}(t) + \lambda \cdot c \cdot p_2(t) - \delta \cdot p_3(t), \\ p'_s(t) &= 0, \\ p'_D(t) &= \lambda \cdot p_2(t) + \lambda \cdot p_3(t). \end{aligned} \tag{22}$$

The failure detection mechanism influence (Figure 1 a)) on the hardware safety integrity of the SRCS provided, that $t_d \to 0$ (theoretical, but acceptable assumption) can be modelled using the DTMC (Figure 6) and described by the transition probability matrix

$$\mathbb{P} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}. \tag{23}$$

Using the matrix (23), is possible to calculate the initial distribution for solution of the differential equations system (22) for the next diagnostic cycle

$$\overrightarrow{P_{n+1}(t=0)} = \overrightarrow{P_n(t=t_{cd})} \cdot \mathbb{P} \tag{24}$$

where n is the test cycle order ($n = 1$ after putting the SRCS into operation), $\overrightarrow{P_n(t=t_{cd})}$ is solution of the differential equations system (22) with the initial distribution (18) in the time $t = 0$ or the initial distribution calculated according to (24) in the time $t = t_{cd}$ (after the execution of n^{th} test cycle).

$$\overrightarrow{P_n(t)} = \left\{p_{OK}^{(n)}(t), p_2^{(n)}(t), p_3^{(n)}(t), p_s^{(n)}(t), p_D^{(n)}(t)\right\}. \tag{25}$$

The initial distribution for the next $(n+1)$ cycle

$$\overrightarrow{P_{n+1}(t=0)} = \Big\{p_{OK}^{(n)}(t=t_{cd}), p_2^{(n)}(t=t_{cd}), 0, p_3^{(n)}(t=t_{cd}) + p_s^{(n)}(t=t_{cd}), p_D^{(n)}(t=t_{cd})\Big\}. \tag{26}$$

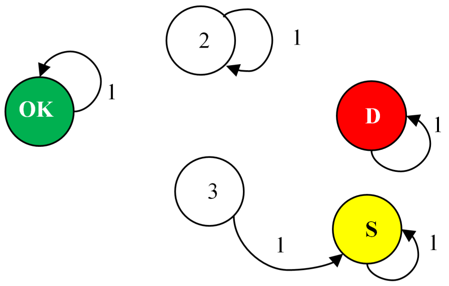


Figure 6. The DTMC model for the dual structure with the time-discrete failures diagnostics – continuous mode of operation and with one failure detection mechanism

B. Influence of Two Diagnostics Mechanisms – Continuous Mode of Operation

If the SRCS contains two failure detection mechanisms, then it is possible to consider various combinations of operation of these failure detection mechanisms according to the parameters and the operation mode. Generally, it is such a combination of the failure detection mechanisms, that one mechanism is intended for the detection of the maximum number of failures in the shortest possible time interval (the RM) and the second mechanism (the SM) is intended for the detection of a certain group of failures, which are not covered by the first mechanism.

The analysis of the failure consequences on the hardware safety integrity can be based on the same principles as in the case of one failure detection mechanism. Although it is possible to proceed in a number of ways, the most suitable are the following ones:

1. If the failure detection time is approximately the same for both the mechanisms, thus it is possible to proceed in such a way, as if the SRCS contains only one failure detection mechanism with the diagnostic coverage coefficient, which can be calculated according to the (10) (if the P_x is unknown, is necessary to choose $P_x = 0$) and the failure detection time, which can be determined according to the (11).
2. It is possible to divide the set of failures on two subsets (X, Y) and make the analysis for each of them separately. The final probability of the dangerous state
$$p_D(t) = p_{DX}(t) + p_{DY}(t) - p_{DX}(t) \cdot p_{DY}(t), \tag{27}$$
provided, that the dangerous state occurrence probability in consequence of the failures from the first subset $p_{DX}(t)$does not influence the dangerous state occurrence probability in consequence of the failures from the second subset $p_{DY}(t)$ and vice versa.
3. If the $t_{Sd} \gg t_{Rd}$, it is possible to proceed in such a way, that the RM operates continuously in time and the SM operates discretely in time.

If it is valid, that $t_{Rd} \ll t_{Sd} \ll t_{proof}$ (t_{proof} is the maximum time value between two proof tests; in extreme cases that can be identified with the useful life of the SRCS), not only the failures diagnostics of the RM, but even the failures diagnostics of the SM can be considered as the continuous-time process. Then the SRCS reaction to the failures occurrence can be described by the CTMC, which is shown in Figure 4.

The transition rate from the state 3 to the state S (Figure 4) can be determined according to the (14) or (15). If the failure is detectable by the RM, then $t_d = t_{Rd}$. If the failure is detectable by the SM, then $t_d = t_{Sd}$. Real value of the $t_d \in \langle t_{Rd}, t_{Sd} \rangle$. In case of pessimistic approach, it can be assumed, that $t_d = t_{Sd}$. The diagnostic coverage coefficient of the failures can be determined according to the (10). The dangerous state probability can be calculated according to the (19) for the time interval $t \in \langle 0, t_{proof} \rangle$ and if the proof-test is perfect, this curve will be repeated periodically [9].

6 THE EXPERIMENTAL RESULTS AND DISCUSSION

Let us suppose that the SF is implemented by the dual structure based on composite fail-safety with fail-safe comparison, as it is shown in Figure 3. The SRCS in compliance with functional specification of the SF controls EUC. The unit R and the unit L are hardware identical. Their failures rate $\lambda = \lambda_L = \lambda_R = 2 \times 10^{-5}$ h^{-1}. The functional specification of the SF is irrelevant from the view of hardware safety integrity analysis of the SRCS. Let the supposed time interval, in which the dangerous failure probability of the SF ($p_D(t)$) will be calculated, be 1 year (it can be, e.g., the time interval between the proof tests).

The SRCS operates in such a way that if the failure is detected, the safety reaction is triggered and the SRCS transits to the state S (interruption of the SRCS operation). The transition rate of the SRCS to the state S is determined according to the (15).

A. One Failure Detection Mechanism

Let the SRCS have one failure detection mechanism with the diagnostic coverage coefficient of the failures $c = 0.99$, which operates in such manner that the diagnostic test is triggered every 0.5 h. The time duration of test and the time of reaction to the failure is negligible with respect to the considered time interval 0.5 h.

If the SF is performed in continuous mode of operation and the diagnostic cycle time (the failure detection time) is significantly less than the time between two proof tests ($t_{cd} \ll t_{proof}$), the dangerous failure probability of the SF can be calculated according to the relation derived for the model in Figure 4. The time dependence ($p_D(t)$) is shown in Figure 7.

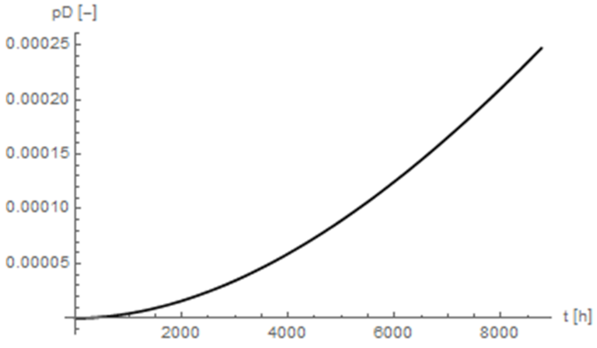
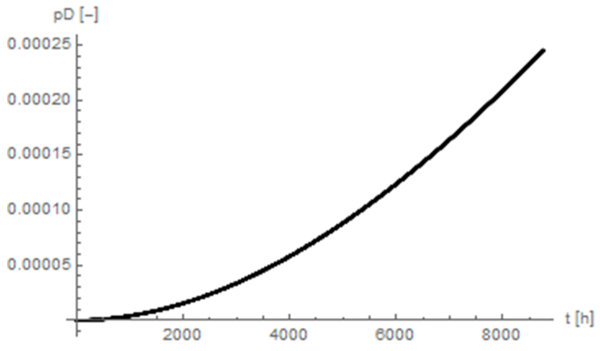


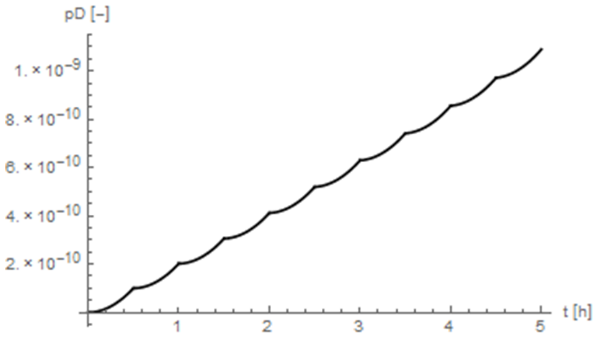
Figure 7. The dangerous failure probability of the SF; one failure detection mechanism; continuous mode of operation – calculation using the CTMC model (Figure 4)

The dangerous failure probability of the SF can be calculated also using the relations derived from models in Figure 5 and Figure 6 (CTMC/DTMC combination). The time dependence $p_D(t)$ is shown in Figure 8 a). Modelling of the dangerous failure probability of the SF using the CTMC/DTMC combination is closer to reality, but the calculation is significantly more time-consuming than in case of using only the CTMC model.

Figure 8 b) shows only the shortened time frame of the dangerous failure probability of the SF calculated using the CTMC/DTMC, to achieve observability of the influence of the time-discreet diagnostic method on the monitored variable $-p_D(t)$.



a) The whole time frame



b) The shortened time frame

Figure 8. The dangerous failure probability of the SF; one failure detection mechanism; continuous mode of operation – calculation using the CTMC/DTMC model (Figures 5, 6)

The failures diagnostic coverage influence on the dangerous failure probability of the SF can be seen in Figure 9. The results confirm the known fact, that failures diagnostic coverage under 60 % does not significantly influence the hardware safety integrity of the SF. The failures diagnostic coverage influence is significant, if $c \to 1$.

B. Two Failure Detection Mechanisms

Let us assume that the SRCS has two failure detection mechanisms. Let one failure detection mechanism (RM) operate in such manner that the diagnostic test is triggered every 0.5 h and its diagnostic coverage coefficient of the failures $c = 0.9$.

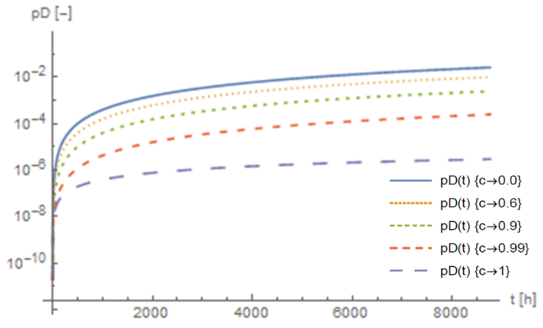


Figure 9. The failures diagnostic coverage influence on the dangerous failure probability of the SF; one failure detection mechanism

Let the next failure detection mechanism (SM) operate in such manner that the diagnostic test is triggered every 10 h. Let the SM cover 90 % of the failures, which are not covered by the RM ($c_x = 0.09$).

If the SF is performed in continuous mode of operation and the diagnostic cycle time (the failure detection time) is significantly less than the time between two proof tests ($t_{cd} \ll t_{proof}$), the dangerous failure probability of the SF can be calculated according to the relation derived for the model in Figure 4. The time dependence $p_D(t)$ is shown in Figure 10. The comparison of the graphs in Figure 7 and in Figure 10 shows that even if the diagnostic coverage coefficient is the same in both cases, the SRCS with two failure detection mechanisms has worse safety properties. Deterioration of the safety properties is caused by the bigger failure detection time.

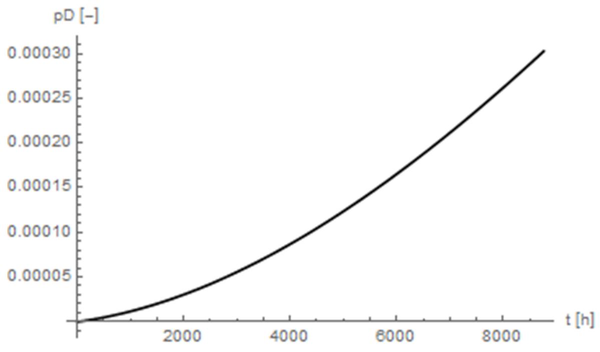


Figure 10. The dangerous failure probability of the SF; two failure detection mechanisms; continuous mode of operation – calculation using the CTMC model (Figure 4)

7 CONCLUSIONS

In the paper, the basic idea how to solve problem related to the evaluation of differently operating failure detection mechanisms to the hardware safety integrity of the SF is analyzed. Solution of this problem is based on the appropriate CTMC and DTMC combination. All the factors were respected in the presented models that significantly influence the hardware safety integrity of the SF.

In order to underline the significance of the problem, the obtained results are presented as a simple dual structure with two elements. In practice, there are often much more complex structures, when the SF is realized by bigger number of elements, which are not only on the process level of control, but also on the higher levels. This leads to the fact, that the SF is realized by parts of the SRCS with different structures. In such cases the model creation is difficult – the number of states in the model increases markedly, thus showing a tendency to increase the probability of mistakes made by the analyst. A successful solution of this problem lies in the decomposition of the SF hardware realization on modules with simple structures and in appropriate using of combination of the different analysis methods when integrating the partial results.

The dangerous failure probability of the SF can be calculated based on the models presented in this paper. In practical use, on the basis of knowledge of the dangerous failure probability, it is necessary to calculate the variable, which is required by relevant standards given to the application area. For example, [1] requires to determinate the average probability of dangerous failure on demand of the SF (PFD_{avg}) in low demand mode of operation, or the average frequency of dangerous failure of the SF (PFH) in high demand mode of operation or continuous mode of operation.

The ideas mentioned in this paper have practical importance and they can be properly used for the hardware safety integrity evaluation of the SF, which has more complex hardware structure and several (at the same time operating) failure detection mechanisms.

Acknowledgment

This work was supported by the Educational Grant Agency of the Slovak Republic (KEGA) No. 034ŽU-4/2016 "Implementation of Modern Technologies Focusing on Control Using the Safety PLC into Education" (45 %) and particularly by the project No. 008ŽU-4/2015 "Innovation of HW and SW Tools and Methods of Laboratory Education Focused on Safety Aspects of ICT within Safety Critical Applications of Processes Control" (45 %), and by the grant VEGA 1/0663/17 (10 %).

REFERENCES

[1] EN 61508:2010. Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems.

[2] EN 61511:2004. Functional Safety – Safety Instrumented Systems for the Process Industry Sector.

[3] Holub, P.—Börcsök, J.: Advanced PFH Calculations for Safety Integrity Systems with High Diagnostic. XXII International Symposium on Information, Communication and Automation Technologies (ICAT 2009), Bosnia, 2009, pp. 1–8. ISBN 978-1-4244-4220-1, doi: 10.1109/ICAT.2009.5348449.

[4] Iden, J.: Assessing the Effects of Diagnostic Failures on Safety-Related Control Systems. International Automatic Control Conference (CASC), Taiwan, 2014, pp. 23–28. ISBN 978-1-4799-4586-3, doi: 10.1109/CACS.2014.7097156.

[5] Ilavský, J.—Rástočný, K.—Ždánsky, J.: Common-Cause Failures as Major Issue in Safety of Control Systems. Advances in Electrical and Electronic Engineering, Vol. 11, 2013, No. 2, pp. 86–93. ISSN 1804-3119, doi: 10.15598/aeee.v11i2.748.

[6] Mechri, W.—Simin, C.—BenOthman, K.: Switching Markov Chains for a Holistic Modeling of SIS Unavailability. Reliability Engineering and System Safety, Vol. 133, 2015, pp. 212–222. ISSN 0951-8320, doi: 10.1016/j.ress.2014.09.005.

[7] Liu, B.—Ghazel, M.—Toguyéni, A.: Model-Based Diagnosis of Multi-Track Level Crossing Plants. IEEE Transactions on Intelligent Transportation Systems, Vol. 17, 2016, No. 2, pp. 546–556. ISSN 1524-9050, doi: 10.1109/TITS.2015.2478910.

[8] Liu, Y.: Discrimination of Low- and High-Demand Modes of Safety-Instrumented Systems Based on Probability of Failure on Demand Adaptability. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, Vol. 228, 2014, No. 4, pp. 409–418. ISSN 1748-006X.

[9] Kleyner, A.—Volovoi, V.: Application of Petri Nets to Reliability Prediction of Occupant Safety Systems with Partial Detection and Repair. Reliability Engineering and System Safety, Vol. 95, 2010, No. 6, pp. 606–613. ISSN 0951-8320, doi: 10.1016/j.ress.2010.01.008.

[10] Rástočný, K.—Ilavský, J.: Effects of a Periodic Maintenance on the Safety Integrity Level of a Control System. In: Schnieder, E., Tarnai, G. (Eds.): Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems (FORMS/FORMAT 2010). Springer-Verlag, 2010, Part 2, pp. 77–85. ISBN 978-3-642-14261-1.

[11] Rástočný, K.—Ilavský, J.: Effects of Recovery on the Safety of a Safety-Related Control System. IEEE International Conference on Applied Electronics (AE), Pilsen, Czech Republic, 2011, pp. 321–324. ISBN 978-80-7043-865-7.

[12] Rástočný, K.—Franeková, M.—Zolotová, I.—Rástočný, K. Jr.: Quantitative Assessment of Safety Integrity Level of Message Transmission Between Safety-Related Equipment. Computing and Informatics, Vol. 33, 2014, No. 2, pp. 343–368. ISSN 1335-9150.

[13] Rástočný, K.—Franeková, M.—Holečko, P.—Zolotová, I.: Modeling of Hazards Effect on the Safety Integrity of Open Transmission Systems. Computing and Informatics, Vol. 35, No. 2, 2016, pp. 470–496. ISSN 1335-9150.

[14] Innal, F.—Dutuit, Y.—Rauzy, A.—Signoret, J.-P.: New Insight into the Average Probability of Failure on Demand and the Probability of Dangerous Failure Per Hour of Safety Instrumented Systems. Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability, Vol. 224, 2010, No. 2, pp. 75–86. ISSN 1748-006X.

[15] Ustoglu, I.—Kaymakci, O. T.—Börcsök, J.: Effects of Varying Diagnostic Coverage on Functional Safety. International Symposium on Fundamentals of Electrical Engineering (ISFEE), Romania, 2014, pp. 1–6. ISBN 978-1-4799-6820-6, doi: 10.1109/ISFEE.2014.7050581.

[16] Velten-Philipp, W.—Houtermans, M.: The Effect of Diagnostic and Periodic Proof Testing on the Availability of Programmable Safety Systems. Proceedings of the 10th WSEAS International Conference on Communications, Athens, 2006, pp. 180–186. ISBN 960-8457-47-5.

Karol Rástočný graduated at the Department of Signalling and Communication Systems of the Faculty of Mechanical and Electrical Engineering, Technical University of Transport and Communications, Žilina, Slovakia in 1982. He defended is Ph.D. in the field of safety analysis in 1995. Since 2008 he has been working as Professor at the Department of Control and Information Systems at the Faculty of Electrical Engineering, University of Žilina. His professional orientation covers solving problems of functional and technical safety of safety related control systems, preferably oriented to railway domain.

Juraj Ždánsky graduated at the Department of Information and Safety Systems of the Faculty of Electrical Engineering, University of Žilina in 2003. He received his Ph.D. degree at the University of Žilina in 2007 in the field of automation with specialization on safety control. Since 2014 he has been working as Associated Professor in the Department of Control and Information Systems at the Faculty of Electrical Engineering, University of Žilina. His professional orientation covers solving problems in functional and technical safety of safety related control systems, preferably oriented to industry.

Mária Franeková graduated at the Department of Telecommunications of the Faculty of Electrical Engineering, Slovak Technical University of Bratislava, Slovakia in 1985. She defended her Ph.D. in the field of channel coding applications in 1995. Since 2011 she has been working as Professor at the Department of Control and Information Systems at the Faculty of Electrical Engineering, University of Žilina, Slovakia. Her scientific research is focused on secure and safety-related communication systems, safety analysis, safety and cryptography techniques used within control of safety-critical processes in transport (railway and road) and in industry.

Iveta Zolotová graduated at the Department of Technical Cybernetics of the Faculty of Electrical Engineering, Technical University of Košice, Slovakia in 1983. She defended her C.Sc. in the field of hierarchical representation of digital image in 1987. Since 2010 she has been working as Professor at the Department of Cybernetics and Artificial Intelligence, Faculty of Electrical Engineering and Informatics, Technical University of Košice, Slovakia. Her scientific research is focused on networked control and information systems, supervisory control, data acquisition, human machine interface and web labs. She also investigates issues related to digital image processing.

Computing and Informatics, Vol. 37, 2018, 476–508, doi: 10.4149/cai_2018_2_476

A STEREO APPROACH TO WILDFIRE SMOKE DETECTION: THE IMPROVEMENT OF THE EXISTING METHODS BY ADDING A NEW DIMENSION

Toni Jakovčević, Marin Bugarić, Darko Stipaničev

Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture
University of Split, R. Boškovića 32, 21000 Split, Croatia
e-mail: {toni.jakovcevic, marin.bugaric, dstip}@fesb.hr

Abstract. In this paper, we present a novel approach to visual smoke detection based on stereo vision. General smoke detection is usually performed by analyzing the images from remote cameras using various computer vision techniques. The literature on smoke detection shows a variety of approaches, and the focus of this paper is the improvement of the general smoke detection process by introducing stereo vision. Two cameras are used to estimate the distance and size of the detected phenomena based on stereo triangulation. Using this information, the minimum size and overall dynamics of the detected regions are further examined to ensure the elimination of false alarms induced by various phenomena (such as the movement of objects located at short distances from the camera). Such false alarms could easily be detected by the proposed stereo system, allowing the increase of the sensitivity and overall performance of the detection. We analyzed the requirements of such system in terms of precision and robustness to possible error sources, especially when dealing with detection of smoke at various distances from the camera. For evaluation, three existing smoke detection methods were tested and the results were compared to their newly implemented stereo versions. The results demonstrated better overall performance, especially a decrease in false alarm rates for all tested methods.

Keywords: Smoke detection, artificial intelligence, stereo vision, false alarms, wildfires

Mathematics Subject Classification 2010: 68T45, 68U10

1 INTRODUCTION

Wildfires, unlike many other natural disasters, are perhaps the only ones that can be largely prevented. Nevertheless, the consequences of uncontrolled wildfires are catastrophic and can lead to significant material damage and can have detrimental impacts on both human safety and health. Regardless of whether the wildfires started accidentally or caused by arson, the hazard can be avoided if they are promptly identified and quickly extinguished.

A lot of efforts have been invested in the early detection of wildfires. Traditional detection is usually based on observers who monitor the surrounding environment in search for a smoke. This is due to the nature of wildfires, where in most cases smoke is visible long before the flame. This is particularly evident for environments with dense vegetation such as forest areas, where fire is not visible until it catches the crowns. The observers are located on observation towers positioned at carefully chosen locations, such as hills, in order to have a better view of the surroundings. Unfortunately, wildfires often occur during extreme conditions such as drought and high temperatures, leading to concentration difficulties and a decreased ability to focus on the actual recognition of smoke.

This issue was partly solved by introducing a camera-based surveillance, allowing the observer to control several observation posts from a single remote location. Unfortunately, practice has shown that these systems still require a long-term attention. To further improve the effectiveness of such systems, automatic smoke detection methods have been developed. Reliability of these methods relies on advanced computer vision algorithms that take into account many different smoke characteristics in order to properly identify the smoke in the image taken from the camera. The system raises an alarm if it determines the presence of smoke in the image. However, a final confirmation by a human operator is still required to distinguish real threats from false alarms.

There is a relation between false alarm and correct detection rates for a given smoke detection system. Although most existing systems have the ability to change the detection sensitivity using detection parameters, this relation between false alarms and correct detections is not significantly affected. An increase in detection sensitivity usually leads to a higher number of correct detections, but unfortunately also to a higher number of false alarms. On the other hand, lowering the detection sensitivity can result in a missed detection, making the system unusable and unreliable. It would be of great benefit to find a solution to reduce the number of false alarms while simultaneously maintaining the acceptable number of correct detections.

In this paper we propose the improvement of the existing smoke detection methods by introducing stereo vision to the detection system. Application of stereo vision in smoke detection systems has already proven to be useful, as shown in [1, 2]. However, in those solutions, stereo vision is primarily used to extract foreground objects from the background. Foreground objects are then further analyzed: i.e., in [1] wavelet transform and discrete cosine for feature extraction and recognition based on fuzzy-neural networks is used, while in [2] additional image features are extracted (contrast, brightness, edge strength, etc.) in order to validate smoke regions. This is different from our approach, where we use stereo vision to estimate the distance to the object or phenomenon visible in the image, what is further used to analyze the detected regions; i.e., to check the minimum size and overall dynamics (changes in size over time) of the detected candidate region. If those regions do not meet specified requirements, they can be discarded as false alarms. Please note that except from introducing this feature to the detection process, the original smoke detection methods are not modified in any way. However, lower false alarms rate achieved by the proposed improvement allows us to raise the detection sensitivity. In other words, it is possible to achieve not only lower rate of false alarms, but also to increase the rate of correct detections as well as the coverage area of the used smoke detection method.

Depth information of the scene visible in the image has also been used to improve existing smoke detection algorithms in recent work presented by Bugarić et al. [3]. However, in this solution distances are estimated using the precise virtual terrain model. The main disadvantage of this system is that previous calculations are mandatory to operate correctly. Calculating a depth map is done for the entire image, which is time-consuming and can last up to ten minutes for high-resolution images. Therefore, such calculations are done only for predefined camera preset positions.

The approach presented in this paper uses stereo vision to process only the candidate regions detected as smoke. In this way we eliminate the need for calculating the depth map for the entire image, meaning that we do not significantly influence the execution time of the original smoke detection method. Also, with this approach, there is no need for using predefined preset positions and the camera can move freely. Please note that using stereo vision, the calculated distance represents the actual distance from the camera to the detected phenomenon, rather than the distance to the terrain behind it, as it is the case in [3].

The rest of the paper is organized as follows: in Section 2 we give an overview of existing visual smoke detection systems where we investigated the most common phases for the detection process. In Section 3 we present our solution to visual smoke detection based on stereo vision, Section 4 deals with possible improvements over standard smoke detection approaches. Finally, a thorough evaluation is carried out in Section 5 where we implemented stereo versions of three existing visual smoke detection methods and compared them to their standard versions.

2 OVERVIEW OF VISUAL SMOKE DETECTION

There are various approaches to visual smoke detection, and in this section we will cover the most common methods and aspects of the detection process. The research in this field has begun over twenty years ago, with one of the first papers [4] dealing with smoke detection based on a live stream from a surveillance camera. The developed systems have been constantly improving since that period, and the field is very alive with new methods constantly emerging. However, most of the methods share several phases that are common to the general smoke detection process. We can divide the detection process in several common phases and cover each phase separately. It is important to emphasize that these phases do not have to be executed sequentially, or in a specific order. In some approaches they are executed simultaneously, and this process is method dependent. Most commonly used phases in smoke detection are: motion detection, region analysis, dynamics analysis and the decision phase. In the following subsections we will briefly cover these phases.

2.1 Motion Detection

Motion detection is the most common phase in smoke detection. Wildfire smoke is dynamic, and this property is used to isolate only the moving regions from the entire scene. This phase generally acts as filter for subsequent phases to reduce the amount of data for further computation, so only the detected regions are forwarded for further analysis. There are many approaches to motion detection for smoke detection purposes such as adaptive background estimation [5, 6], block mean difference [7, 8] or motion history image [9, 10]. Adaptive background estimation approaches in [5, 6] are similar and use a background estimation model based on rules for stationary and moving pixels. In the case that the pixel intensities significantly deviate from the background model they are considered as moving pixels and forwarded to the subsequent phases for further analysis. Another motion detection approach is described in [11] where the background model is initialized dividing the input in 16×16 size blocks. After the setup phase, the model is updated using a selective temporal median with a fixed k-sized circular buffer. The difference between the input image and the background model is computed and then binarized using a low and high thresholds to identify small and high intensity variations. Those pixels that are present in both of the masks generated by low and high thresholds are considered as moving pixels. Extracted objects are then validated jointly using color shape and gradient information to remove noise and artifacts. Further processing phases involve shadow and ghost removal.

Motion detection approach using motion history image described in [9] represents motion in successively layered image differences. Moving objects create silhouettes that represent patterns of motions. The intensity in a motion history image (MHI) represents the recency of motion in the observed scene. In this way, the motion from several frames can be encoded in a single image. Using this approach, it is possible to capture the gradient of smoke motion including orientation and direction.

2.2 Region Analysis

Another very important phase in the general smoke detection process is the region analysis. This phase is often executed after motion detection, where region candidates are extracted. Regions are analyzed based on different smoke characteristics such as color, texture, shape and size. Different methods use different approaches, but it is generally a combination of several characteristics. However, all of the methods rely on color as one of the most important features. When dealing with color analysis, one of the first steps is to choose a color space for analysis. Research described in [12] deals with the effect of various color spaces on the performance of different classifiers in smoke detection. One of the goals of the research was to find a particular color space with highest separability between smoke and non-smoke pixels. Different color spaces were used in the analysis, such as RGB, YCrCb, CIELab, HSI, and a HS'I which is an derivative of HSI. The results suggest that the performance is a distinctive feature of the classifier itself, rather than the classifier-color space combination, and that the favorable color space candidates are HSI and its derivative, as well as RGB color space. Regardless of the color space, all of the methods take into account specific chromatic characteristics of smoke. Smoke color varies based on fuel type and moisture content but it is generally manifested as a light to dark shade of gray. This means that the smoke pixels are positioned diagonally in the RGB color space and it is possible to dismiss the pixels that have high deviation from the diagonal of the color space as implemented in [8, 13]. Another smoke-specific feature is low chrominance in the smoke affected region. Detection of a significant drop in chrominance could be a possible indicator of smoke appearance. Another step in region analysis is using information about the texture of a given region. There are various approaches to texture analysis, such as wavelet analysis [14, 15] or the gray-level co-occurrence matrix (GLCM) [16]. Another important factor in region analysis is the information about the morphological characteristics of the region. Smoke regions have a rather convex contour based on irregular shape with erratic silhouette [5]. Most common approach to isolate these types of regions is to calculate the disorder parameter and compare it to the reference values [17].

2.3 Dynamics Analysis

Another common phase in smoke detection process is the analysis of the dynamics of the candidate regions. The candidate regions are tracked over a certain period of time to ensure they exhibit smoke-like behavior. Regions that do not conform to motion characteristics of smoke can be rejected in this phase of the process. One of the basic characteristics is the growth rate of smoke. Different methods use certain thresholds to isolate only the regions that satisfy a set of predefined conditions [18, 6]. Another aspect is the direction of smoke movement. Smoke usually exhibits upward as well as lateral motion, however, this factor heavily impacted by the wind speed and direction and is rather difficult to predict. Work presented in [19] deals with the analysis of the direction of smoke motion using accumulative motion orientation. The aim of the analysis is to discover regions that are moving upwards due to high temperatures. The method involves calculation of the temporal motion orientation histogram. By removing the regions that do not exhibit growth and gradual upward motion it is possible to reduce the overall number of false alarms and increase the performance of the detection method.

2.4 Decision Phase

The decision phase is the final phase of the detection process. All the information accumulated through the previous phases of detection is now taken into account in order to make the final decision about raising an alarm. There are various approaches to the decision-making based on the data from the previous phases of detection, such as the Bayesian approach [11], neural networks [6, 20], random forests [21], support vector machine [22], the mechanism of thought [23], and others. Based on the available information and the specific decision process, the system makes the final decision whether to raise an alarm for the given situation.

3 INTRODUCING STEREO VISION TO SMOKE DETECTION

Smoke is a phenomenon that has no clearly defined characteristics such as shape, color, etc. This is precisely the reason why there are many different approaches to smoke detection. However, analyzing the size of the smoke region candidates rarely occurs among these approaches. This is due to limitations of the systems that use a single camera, where the only option is to express this size in the number of pixels. Using only this information, it is difficult to estimate the actual size of the smoke phenomenon in the real world, as it largely depends on the camera parameters and the distance of the phenomenon from the camera.

Stereo vision triangulation allows us to estimate a three-dimensional position of the detected smoke in the real world, and also to estimate the real-world size of that region and express it in standard units of measurement. In this way we can further analyze the candidate regions detected as smoke based on their real-world sizes, and thus reduce the number of false alarms. One such example would be identifying false alarms caused by the uncontrolled movement of the vegetation in the close vicinity of the camera based on the estimated size and overall dynamics of the detected objects. Moreover, eliminating such types of false alarms allows us to increase the detection sensitivity, eventually leading to the increase in correct detections rate and coverage area of the used smoke detection method.

Any existing smoke detection system could be upgraded with this stereo vision approach. Given that the smoke detection requires an estimation of relatively large distances, a stereo vision system with a wide baseline is required. In the following sections we analyze the problems that may arise and describe the guidelines that have to be followed when upgrading the existing smoke detection systems.

3.1 Wide-Baseline Stereo Vision

In order to improve smoke detection performance, a stereo system should correctly match the images captured by the cameras and obtain accurately triangulated three-dimensional data. It is important to emphasize that the occurrence of smoke is usually located several hundred meters away from the observation post where the system is installed. Stereo systems capable of estimating such relatively large distances have already been proposed [24, 25], however a study concerning advantages and disadvantages of such a system as a part of a smoke detection was not carried out to date.

Our system requires two cameras with a known horizontal displacement, whose optical axes are parallel. In order to maintain the accuracy of a stereo system, the horizontal displacement between these two cameras should be relatively wide. The actual smoke detection is conducted on the images taken by one of the stereo cameras (either left or right). Stereo vision techniques are applied only for the candidate regions detected as smoke as the additional verification phase. In our solution, the process of determining the distance of the detected phenomenon from the camera is divided into two steps:

- Selecting a point that represents a smoke region candidate in the original image and finding a corresponding point in the image of the other camera (correspondence problem), and
- three-dimensional reconstruction (stereo triangulation).

Each of these steps is further explained in the following subsections.

3.1.1 Correspondence Problem

The correspondence problem is the problem to match image points from two stereo images which are projections of the same point in a three-dimensional space. The main requirement while taking a pair of stereo images in our solution is that the pictures are taken at the same time instant. Smoke is a dynamic phenomenon, constantly changing in shape and size. If this requirement is not met, projection of the smoke on the image plane can be significantly different in two stereo images, making the matching process difficult. As we show later in this section, even a single pixel error in this phase could lead to significant errors when estimating the real-world sizes of the smoke phenomenon. Please also note that any difference in perspective or lighting of two stereo cameras, or any object occlusions can further accentuate the correspondence problem.

Stereo cameras should, therefore, be mutually synchronized in order to capture the images at the same time instant [26]. Automatic brightness adjustment should also be avoided, as it may result in different lightning in two stereo images.

The points of interest would be the points that the smoke detection algorithm determines as the potential candidates. The detection algorithm isolates the regions of interest, and for each region a single point is selected for correspondence. The selected point basically represents the geometric center of the region. Now, the task is to find the corresponding point in the other image. There is a significant number of existing algorithms for stereo correspondence. However, the main problem is not to generate the disparity map for the given scene, but rather to calculate the actual depth for a very sparse set of points that represents the candidate regions in each frame.

One possible way of achieving this task is to use normalized cross correlation. It is a rather straightforward approach where a region around the given point is used as a template. The aim is to find the best matching position of that template in the second image. The search is not performed on the complete second image, but rather on a limited region that covers the space were the corresponding point could be in the second image given the constraints of the geometry of the stereo system. Given the possible variation in brightness due to exposure or lightning conditions it is necessary to perform the normalization of the regions. The point with the highest correlation coefficient is used as the corresponding point in the second image. The process of normalized cross correlation could be computationally demanding when performed on large sets of correspondence points, but for a very sparse set that is generally obtained from the candidate regions the computing load is negligible. We can examine this in more detail. Specific implementation of normalized correlation calculation is based on [27] as follows

$$\gamma(u,v) = \frac{\sum_{x,y}\left[f(x,y)-\bar{f}_{u,v}\right]\left[t(x-u,y-v)-\bar{t}\right]}{\left\{\sum_{x,y}\left[f(x,y)-\bar{f}_{u,v}\right]^2\sum_{x,y}\left[t(x-u,y-v)-\bar{t}\right]^2\right\}^{0.5}} \tag{1}$$

where f is the second image where we search for a match, $\bar{t}$ is the mean of the template, and $\bar{f}_{u,v}$ is the mean of $f(x,y)$ in the region under the template. This calculates the normalized correlation coefficient for the template placed at a point (u,v) in the second image. We can see that the time complexity of calculation for a single point would be $O(M_t N_t)$ where M_t and N_t represent the size of the template. In case we would like to calculate the correspondence of each point in the image of size $M_I \times N_I$ the time complexity would be $O(N_I M_I M_t N_t)$. It is evident that the computational cost of type of calculation is rather high, so it is necessary to perform the calculation only for a set containing the points that represent the candidate regions in order for the algorithm to work in real-time. For example, we can take that the time constraint for real-time processing is 1 frame per second in order to capture smoke dynamics. Average time to calculate normalized cross correlation for a single point, where the template dimensions are 40×20 pixels, and the search space in the second image is 900×100 pixels, on a machine with 3 GHz CPU is approximately 14 ms. This implies that in order for the algorithm to be able to work in real time we must use a sparse set of correspondence points that are most probable candidates for detection.

3.1.2 Stereo Triangulation

The distance of the observed object or phenomenon from the camera is computed using the disparity of two corresponding pixels. In our model, each camera is reasonably approximated by a pinhole camera model, thus ignoring camera lens distortion and other optical nonlinearities. We consider lens to be a point through which all incoming rays of light pass. This means that all objects, regardless of their distance from the camera, project to a single point on the image plane.

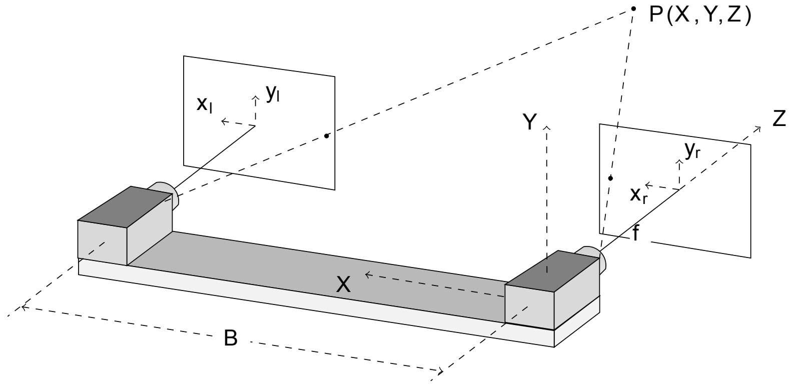


Figure 1. Stereo system

Figure 1 illustrates the configuration of a stereo system that we use to improve the existing smoke detection methods. Two cameras that have parallel image planes are separated in the X direction, and the baseline length is denoted by B. Both cameras share the same focal length that we denote by f. The focal point of the right camera is chosen as the origin of the three-dimensional world coordinate system. Please note that, in order to minimize the measurement inaccuracy, these two cameras have to be carefully aligned and well calibrated. Also, the cameras should be securely mounted to ensure that the alignment is persevered with continuous use.

In this model, a randomly chosen point $P(X, Y, Z)$ in a three-dimensional world is projected onto two camera image planes. These projections in the left and right image planes are represented by (x_l, y_l) and (x_r, y_r), respectively.

The relation of the point $P(X, Y, Z)$ with respect to its image projections is given by:

$$x_r = \frac{fX}{Z}, \quad x_l = \frac{f(X - B)}{Z}. \tag{2}$$

Therefore, using simple trigonometry, we can estimate the depth Z of the point $P(X, Y, Z)$:

$$Z = \frac{fB}{x_r - x_l}. \tag{3}$$

From Equation (3) we see how the distance from the camera can be computed using the disparity $x_r - x_l$. It can be seen that y_l and y_r coordinates do not influence the actual calculation of the depth information. Please note that the aforementioned model does not take into account the following irregularities: different focal lengths of left and right camera, differences in principal point coordinates, skew and distortion parameters. Therefore, in order to use Equation (3) in practice, inputs into this equation should first be normalized. However, even without such irregularities, Equation (3) will not always provide us with the correct depth information. This occurs for several reasons, so in the following of this chapter we will investigate possible errors and discuss how they affect the proposed improvement of the existing smoke detection methods.

3.2 Different Types of Errors

First, we must accept that the depth resolution of a stereo vision system is limited due to the discrete nature of the imaging system [28, 24]. The projection of the point $P(X,Y,Z)$ onto the image planes is approximated to the nearest pixels with coordinates $(\hat{x}_l, \hat{y}_l)$ and $(\hat{x}_r, \hat{y}_r)$. The error that is a result of this approximation is referred to as a discretization error.

The discretization error generates an uncertainty polyhedron when we perform the stereo triangulation. Due to the aforementioned approximation, the actual distance cannot be accurately determined, since it lies somewhere inside this polyhedron. Figure 2 illustrates the discretization error in two dimensions, where uncertainty areas are represented as diamond shapes.

The discretized image points that cause the error are at most within half a pixel of the actual projection, that is:

$$\hat{x}_r = x_r \pm \frac{x_{pix}}{2}, \quad \hat{x}_l = x_l \pm \frac{x_{pix}}{2} \tag{4}$$

where x_{pix} represents the distance between two adjacent pixels along the X direction. From this it follows that the maximum value of the observed depth $\hat{Z}$ can be calculated using Equation (5):

$$\hat{Z} = \frac{Z}{1 \pm Z\frac{x_{pix}}{fB}}. \tag{5}$$

It can also be shown that the maximum value of the discretization error (Z_{disc_err}) is given by:

$$Z_{disc\_err} = |Z - \hat{Z}| = Z^2 \cdot \frac{x_{pix}}{fB \pm Zx_{pix}}. \tag{6}$$

From Equation (6) we can see that the discretization error (Z_{disc_err}) increases as the depth Z increases. Several other interesting facts useful for a better estimation of relatively large distances arise from this equation as well. First, since the focal length of the calibrated camera is not a parameter that can be easily modified, from Equation (6) it follows that we can improve distance estimation by increasing the length of the baseline B. Second, image resolution affects the discretization error as well, since it defines the distance between two adjacent pixels (x_{pix}). From Equation (6) it is obvious that the higher the resolution of the image, the smaller the discretization error.

Nevertheless, it is important to notice the expression $\pm Zx_{pix}$ in the denominator of Equation (6). Uncertainty areas represented as diamond shapes in Figure 2 are bounded by rays that go through the middle of each pixel in the image plane. Therefore, the distance from the intersection of rays to the center of the diamond shape corresponds to the maximum discretization error. Recall, the maximum discretization error is achieved by moving half a pixel in a certain direction (in both left and right image planes). As seen from Figure 2 this movement can result in the estimated distance that is larger or smaller then the actual distance. Therefore, the discretization error is measured from the intersection of rays to the center of the diamond shape that is either further away or closer to the stereo cameras. Let us denote these errors as a maximum positive and a maximum negative discretization error, respectively. Maximum positive discretization error is defined by the negative prefix in the expression $\pm Zx_{pix}$, while the maximum negative discretization error is defined by the positive prefix. Maximum discretization error (either positive or negative) represents a worst case scenario, and in most cases the value of the discretization error is smaller. It can also be noticed that the maximum positive discretization error is larger than the maximum negative discretization error.

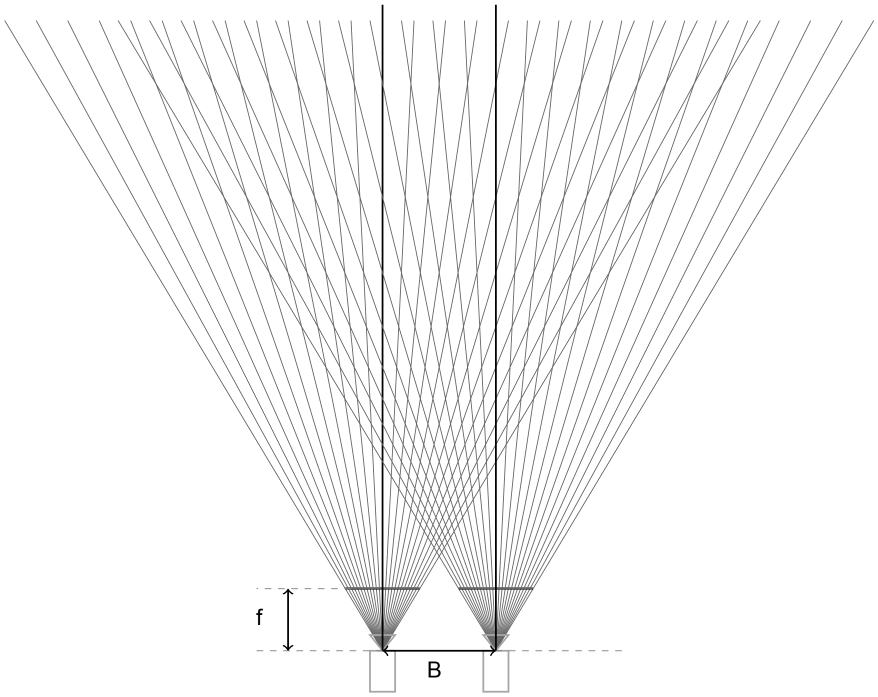


Figure 2. The discretization error as the result of the discrete nature of the imaging system. Diamond shapes represent uncertainty areas, i.e., the system returns the same distance for all the points inside the same diamond region.

It is also very important to consider possible camera alignment errors and their impact on the depth estimation. Once the stereo rig is constructed, the cameras are calibrated in order to obtain intrinsic and extrinsic camera parameters. Using the calibration information we can model the possible misalignments of the individual cameras relative to each other. However, the perfect calibration is not possible and a certain degree of error will still be present. Additionally, the camera setting is prone to slight misalignment due to continued usage and transport, so we also have to take into account the possible errors that arise from this sort of misalignment. We will consider four types of errors; e.g. error due to roll between cameras, error due to pitch between cameras, error due to yaw between cameras and finally the error due to lens distortion. These types of errors are explained in detail in [29] so we will only cover them briefly. The first type of error is the depth error due to roll of the second camera, while the first camera is correctly aligned. Using trigonometry relations we can arrive at the the following equation:

$$Z_{roll\_err} \simeq Z\frac{X_2(cos\theta - 1)}{B} \tag{7}$$

where Z_{roll_err} is the depth error, Z is the true depth of the object, X_2 is the true 3-D coordinate of the object in the x dimension in the coordinate frame of the second camera, θ is the roll angle of the camera, and B is the baseline.

Another type of possible error arises when the second camera is not on the same level as the first, and rotates about a line which is parallel to the bar (pitch error). This type of error can be calculated as follows:

$$Z_{pitch\_err} \simeq -\frac{1}{2}\frac{X_2 Z(\tan\alpha)^2}{B} \tag{8}$$

where α is the pitch angle between the two cameras.

The third type of error arises when the second camera is rotated about an axis perpendicular to the epipolar plane and through the center of projection (yaw error). This type of error can be approximated using the following equation:

$$Z_{yaw\_err} \simeq -\frac{\tan\beta(Z^2 + X_2^2)}{B} \tag{9}$$

where β is the yaw angle between the two cameras. It is important to emphasize that this type of error is the most common, and contributes the most to the error in measured depth.

Another type of error arises from the possible lens imperfections. Although we use a pinhole camera model, we can also consider the impact of this type of error in real world applications. As covered in [29], using the radial lens distortion model, the depth error can be calculated as

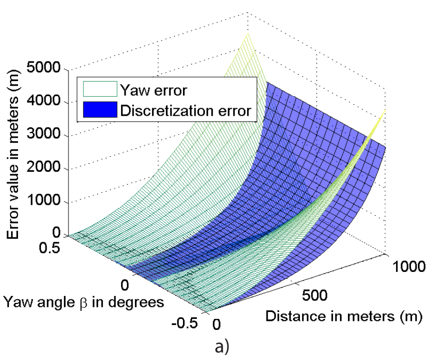
$$Z_{lens\_err} = Z - \frac{1}{\frac{1}{Z} - \frac{f^2}{Z^3B}[\kappa_2 X_2^3 - \kappa_1(X_2 - B)^3]} \tag{10}$$

where κ_1 and κ_2 are the lens distortion coefficients.

Now, the accumulated error can be approximated as:

$$Z_{acc\_err}(Z,\alpha,\beta,\theta,\kappa) \simeq Z_{disc\_err}(Z) + Z_{roll\_err}(\theta) + Z_{pitch\_err}(\alpha) + Z_{yaw\_err}(\beta) + Z_{lens\_err}(\kappa). \tag{11}$$

We can compare the effects of different types of errors on the accumulated error. Figure 3 a) shows the comparison of the absolute yaw error and the discretization error for variable distance of the observed object Z and yaw angle β.



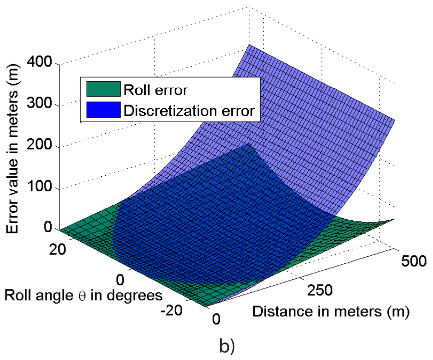


Figure 3. a) Absolute yaw error $|Z_{yaw\_err}|$ (green), and maximum absolute discretization error Z_{disc_err} (purple), b) absolute roll error $|Z_{roll\_err}|$ (green), and maximum absolute discretization error Z_{disc_err} (purple) (for a stereo system with $B = 2\,\text{m}$, $f = 4.655\,\text{mm}$, $x_{pix} = 0.0071\,\text{mm}$, and $X2 = 3\,\text{m}$)

When calculating the discretization error we use the maximum positive discretization error (the expression $\pm Zex$ in Equation (6) with a negative prefix) since it generates a greater overall error. It is possible to see from the figure that the slight misalignment of the camera in the yaw direction generates a significant error (the graph shows the yaw error in the interval of ± 0.5 degrees). On the other hand, the effects of the roll and pitch errors are negligible compared to the discretization error (Figure 3 b)) shows the comparison of the roll and discretization error). Since the stereo setup for smoke detection will be working with relatively large distances we can discern the roll and pitch errors that have an insignificant effect on the accumulated error when compared to the discretization error.

For example, given a stereo setup with baseline $B = 2\,\text{m}$, the focal length of both cameras $f = 4.655\,\text{mm}$, pixel dimensions in the image plane $x_{pix} = 0.0022\,\text{mm}$, where the observed object is located at distance of $Z = 500\,\text{m}$, with a shift in the x dimension of $X2 = 7.5\,\text{m}$ relative to the second camera. In case that the detection system requires the precision that would allow the depth estimation error to be no more than 100 m for this distance, we can calculate the maximum allowed error from different sources. Using the Equation (6) we can calculate that the maximum discretization error for the given distance would be $Z_{disc\_err} = 66.9\,\mathrm{m}$, so the rest of the error from other sources would be 33.1 m. It is important to emphasize that this is the maximum discretization error or the worst case scenario for the given distance. If we assume that each of the remaining types of errors (roll, pitch and yaw error) contribute evenly to the rest of the error it would mean that the maximum error from each of these sources would be less than 11.03 m. This means that the alignment accuracy for the given setup for specific rotations would be $\alpha < 6.19°$ (pitch error), $\beta < 0.0051°$ (yaw error) and $\theta < 6.21°$ (roll error). We omit the contribution of the lens distortion error for practical reasons.

From this example we can understand which calibration alignment types require more attention when constructing a stereo setup for smoke detection. Since we cannot influence the discretization error (except by widening the baseline or increasing the resolution of the image), the main key points are to ensure a thorough calibration of the system, and especially a very precise yaw orientation of the both cameras since it has a significant impact on the overall error.

4 IMPROVEMENT OF STANDARD DETECTION APPROACHES USING DEPTH INFORMATION

Standard visual smoke detection systems are usually equipped with a single rotating camera or a setup of several cameras pointed in different directions covering the 360° area around the detection post. The images acquired from the camera are analyzed in order to detect potential occurrence of smoke in the scene. In Section 2 we have covered the main phases of smoke detection process that are common to most smoke detection systems. However, using this kind of setup it is not possible to reliably estimate the distance of the detected phenomena from the camera. This additional information could be very useful in several phases of detection and help to improve the overall reliability of the system.

There are two main phases that could be improved using the information about the distance of the detected phenomena: region analysis phase and dynamics analysis phase. As covered in Section 2, the region analysis phase deals with the analysis of the candidate regions based on different smoke characteristics such as color, texture, shape and size. In order to eliminate noise from the detection process, most of the methods define a minimal size threshold for the detected region. This size is often expressed in the number of pixels that constitute the region. However, the number of pixels in the region does not provide the actual information about the physical size of the detected object. A group of pixels could represent a small object close to the camera or a large object at a great distance. One aspect of the improvement using stereo vision is to provide this additional information about the actual distance of detected objects. With this information it is possible to estimate the actual physical size of the detected objects. Objects or phenomena that do not satisfy minimal size constrains can now be eliminated from the detection process. This leads to a lower number of false alarms and a more reliable and robust detection process.

Another detection phase that could be improved is the dynamics analysis phase. This phase deals with the behavior of the candidate regions over time, where the regions are tracked over a certain period to verify that they exhibit smoke-like behavior. One of the main aspects of smoke dynamics is the growth rate of smoke regions in the image. In the incipient phase of wildfire, smoke gradually appears in the scene and continues to grow until reaching the full size. The growth rate and smoke size depend on many different parameters such as the fuel type, moisture content, wind speed and wind direction. However, it is possible to establish certain growth thresholds based on empirical data. By estimating actual physical size of the objects it is possible to eliminate regions in the image that exhibit very rapid growth that is much faster than the natural expansion of wildfire smoke. Eliminating such regions affects the performance of the system, by improving the reliability of the detection and reducing the number of false alarms.

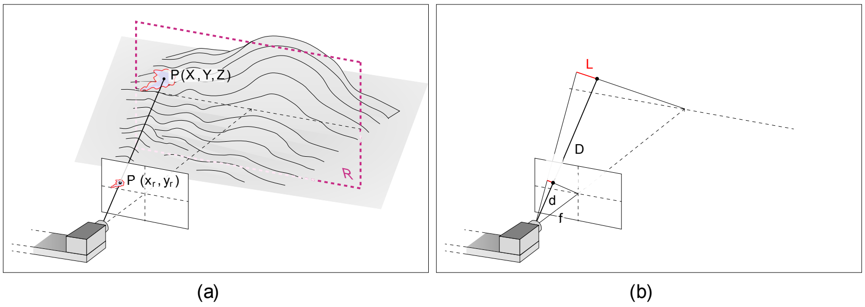


Figure 4. Calculating pixel coverage using: a) the relation between the point $P(X, Y, Z)$ in the real world and $P'(x_r, y_r)$ in the image plane, and b) the relation between shift L in the real world and the shift ℓ in the image plane

As already mentioned, due to the perspective of the camera, a group of pixels visible in the camera's image plane can represent a small object close to the camera, or a large object at a greater distance. Accordingly, we believe that it is better to express the size of the detected objects in standard units of measurement (such as meters) rather then in number of pixels.

Figure 4 illustrates the method used for estimating physical sizes of the objects visible in the scene. Using a stereo vision system introduced in previous sections, we can determine the depth (Z), as well as the actual distance (D) of the detected phenomena from the camera. Please recall, in this scenario the focal point of the right camera is chosen as the origin of the three-dimensional world coordinate system, although the system could easily be adapted to operate using the left camera. Also, as shown in Figure 1, the origin of the image coordinate system is positioned in the center of the camera image plane.

Let us assume that the detected phenomena is located at point $P(X, Y, Z)$ in the real world, and that point $P'(x_r, y_r)$ represents the projection of the point P onto the image plane of the (right) camera. We propose that the size of the detected phenomena be estimated from its projection onto the plane that has the following characteristics: it and the detected phenomena are equal distance from the (right) camera, and it is parallel to the (right) camera's image plane (plane R in Figure 4 a)).

Before proceeding to the actual estimation of the physical size of the detected phenomena, let us make a remark: each pixel in the image also represents a space in the real world. This space can also be projected onto the described plane R, and therefore, the size of the space visible in the pixel can also be estimated.

Figure 4 b) demonstrates the method used to estimate the size of the physical space (visible inside a single pixel) projected onto the plane R that is parallel to the camera image plane. Let f represent the focal length of the (right) camera and d the distance from the focal point of the camera to the point $P'(x_r, y_r)$.

Let us first denote with ℓ the width of a single pixel (on the camera image plane). As illustrated in Figure 4 b), ℓ also represents the length of a line segment that lies on the line going through both the point $P'(x_r, y_r)$ and the center of the image plane. This shift of the length ℓ towards the edge of the camera image plane has a corresponding shift in the plane R, and let us denote it with L.

Variables f, d, ℓ and L can be expressed either in number of pixels or in standard units of measurement. Therefore, before proceeding, let us make some remarks concerning notation: from now on, all variables indexed with $(\cdot)_p$ will be associated with distance expressed in number of pixels, while the variables indexed with $(\cdot)_m$ will be associated with distance expressed in standard units of measurement (in this case meters). Our goal is to find the value of L_m that we will use to estimate the size of the physical space visible inside a single pixel.

Focal length is a parameter that is often provided by the manufacturers of the used equipment, but can also be retrieved by camera calibration. If given in number of pixels, it can easily be converted to standard units of measurement:

$$f_m = \frac{f_p \cdot ccd_m}{w_p} \tag{12}$$

where ccd_m represents the CCD width expressed in standard units of measurement (often millimeters) and w_p represents the width of the image expressed in number of pixels.

It can be shown that the value of d_m representing the distance from the focal point of the camera to the point $P'(x_r, y_r)$ can be calculated as follows:

$$d_m = \frac{f_m}{f_p} \cdot \sqrt{x_r^2 + y_r^2 + f_p^2}. \tag{13}$$

Please recall, the stereo vision system provides us with the value of D_m, therefore we can calculate the value of the shift L_m as follows:

$$L_m = \frac{f_m}{f_p} \cdot \frac{D_m}{d_m}. \tag{14}$$

Finally, the length of the shift L_m on the plane R that corresponds to the shift of one pixel on the camera image plane, expressed in standard units of measurement, is given by Equation (15).

$$L_m = \frac{D_m}{\sqrt{x_r^2 + y_r^2 + f_p^2}}. \tag{15}$$

Let us make the assumption that the pixels are square, then L_m^2 represents the area of the projected space onto the plane R that is parallel to the camera image plane. In other words, L_m^2 represents the size of the space visible in a single pixel positioned at coordinates (x_r, y_r) in the image plane. From now on, let us denote L_m^2 as a pixel coverage area.

From Equation (15) it can easily be seen that the pixel coverage area and the distance from the camera are directly proportional, meaning that the distant objects will appear smaller in the image. Using pixel coverage areas of all the pixels representing the detected phenomena, we can estimate it's overall physical size. However, stereo vision system does require some non-negligible amount of time to calculate the distance D_m. Hence, for practical reasons, we calculate the pixel coverage area only for one pixel positioned at the center of the detected region. The overall physical size can, therefore, be approximated as the pixel coverage area of the chosen pixel multiplied by the overall number of pixels representing the detected phenomena. In this manner, we do not slow down significantly the actual smoke detection method, since distance estimation using stereo vision is performed only once for each detected region in the image.

Now that we can calculate the pixel coverage area for a pixel in the center of the region, we can estimate the actual area of the detected objects in the real world. The detection method holds the information about the number of pixels in the candidate region, and by using the pixel coverage area of the central pixel we can estimate the area of the entire region. This area refers to the the area in the plane parallel to the image plane intersecting the actual object in physical space as depicted in Figure 4 a).

4.1 Filtering of the Candidate Regions Based on the Estimated Size

The information about the smoke area can now be used in the detection phases described earlier. The candidate regions detected as smoke in the region analysis phase can be further analyzed based on their real-world sizes. The candidate regions that are below the minimum size are eliminated from the detection process. The main reason for doing this is to reduce the number of false alarms. In fact, one of the major causes of noise, and therefore false alarms, is a movement of small objects visible in the scene. One example could be the uncontrolled movement of vegetation (such as grass or tree branches) in the close vicinity to the camera. Although these objects often share some similar characteristics with the smoke, they could be easily dismissed as false alarms if their size is accurately approximated using the method described above.

In our case, we use the minimal size threshold of $5\,\mathrm{m}^2$ based on offline measurements and analysis. We believe that the most of the movement caused by smaller objects in the scene can be eliminated using this threshold, whereas, given the nature of the smoke, actual smoke regions will relatively quickly exceed this threshold.

Nevertheless, one important factor in the estimation of actual smoke size is the possible error due to the different error sources described in Section 3.2. When dealing with large distances, the discretization error becomes predominant due to the quadratic term of Equation (6). Therefore, Figure 5 shows the pixel coverage area for one pixel with regard to the distance from the camera. The figure also shows the pixel coverage area with the maximum positive and negative discretization errors (when the estimated distance is actually larger or smaller than the real distance of the object due to discretization). Recall, the maximum positive discretization error is larger then the maximum negative error as stated in Equation (6) (for the positive error, the second term in the denominator has a negative prefix).

From Figure 5 it is obvious that the area estimation becomes unreliable at larger distances, so it is important to ensure that valid region candidates are not rejected in this process. As already explained, this phase is used to reject candidates that are smaller than the predefined threshold ($5\,\mathrm{m}^2$). Please note that the positive discretization error results in larger pixel coverage area, so in this case the valid candidate regions are not rejected. However, the positive discretization error results in a non-rejection of noise regions that are estimated to have a size above the threshold due to this error. On the other hand, the negative discretization error may result in the rejection of valid candidate regions due to the negative error in size estimation.

We propose reducing the predefined minimal size threshold from initial $5\,\mathrm{m}^2$ to only $1\,\mathrm{m}^2$ if the estimated distance of the candidate region is beyond a certain distance D_{max} after which the area estimation becomes unreliable. In other words, D_{max} is a distance after which we cannot properly estimate the real-world size of the candidate region due to the discretization error. By doing this, we maintain the accuracy of the original smoke detection method at distances larger than D_{max} (for the same level of detection sensitivity), while at shorter distances we achieve the elimination of false alarms induced by the phenomena located in the vicinity of the camera. The value for D_{max} can be arbitrarily set based on the level of precision required from the system. For this purpose, we introduce a user defined limit err_ϑ defined as the maximum allowable percentage error of smoke area inside a single pixel. The percentage error (err) can be calculated as follows:

$$err = \frac{estimated\ area - real\ area}{real\ area} * 100. \tag{16}$$

As an example, for $B = 2\,\mathrm{m}$, $f = 4.655\,\mathrm{mm}$ and $x_{pix} = 0.0022\,\mathrm{mm}$ and a chosen value for the maximum allowable error $err_\vartheta = 21\%$, based on Equations (6),

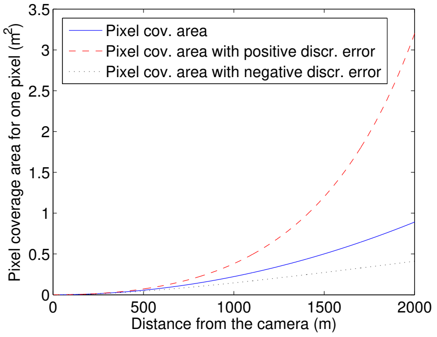


Figure 5. Pixel coverage area (m^2) for one pixel with regard to the distance from the camera (blue solid line), pixel coverage area with maximum positive discretization error for the given distance (red dashed line), and pixel coverage area with maximum negative discretization error for the given distance (black dotted line). The selected pixel has coordinates $(100, 100)$ referent to the center of the image, with $B = 2\,\mathrm{m}$, $f = 4.655\,\mathrm{mm}$ and $x_{pix} = 0.0022\,\mathrm{mm}$.

(15) and (16), it is possible to calculate the value for the maximum distance with allowable error in front of the camera $D_{max} = 385\,\mathrm{m}$.

In this way, the reliability of the pixel coverage estimation could be adjusted based on the needs of the specific detection method or system. If the percentage error increases over the user defined threshold, the minimum size threshold reduces to $1\,\mathrm{m}^2$ in order to compensate for the possible discretization error. It is important to emphasize that the detection method is still capable of detecting smoke beyond the distance specified by D_{max} with the same accuracy as the original version. The reduction of minimum size threshold is to ensure that real smoke plumes are not discarded based on size due to discretization error. This implies that the detection of smoke is sill accurate, but the benefit gained by rejection of potential false alarms based on size reduces with distance.

Another phase where information about pixel coverage area can be useful for reducing false alarm rate and improving the reliability of the detection process is the dynamics analysis phase. The dynamics analysis phase is used to eliminate regions that do not exhibit smoke-like behavior. Smoke behavior is rather difficult to define or simulate since it depends on many different factors as described earlier in this section. However, overall smoke dynamics adhere to certain rules regarding the rate of spread. The measurements published in [30] show that the average smoke area rate of spread (μ) in the image plane is $16.04\,\mathrm{m}^2/\mathrm{s}$ with standard deviation (σ) $76.33\,\mathrm{m}^2/\mathrm{s}$ in the first 3 minutes after the occurrence. It is also stated that by fitting the data with t location-scale distribution it can be calculated that over 99 percent of smoke area change observations fall into the interval defined by $(\mu - 6\sigma, \mu + 6\sigma)$.

Using this information, and the information about the area of the candidate regions calculated using the pixel coverage area, it is now possible to reject those regions with grow or shrink rates outside the allowed range. This consequently results in higher reliability and accuracy of the detection method or system.

As already mentioned, by reducing the false alarms rate of the existing smoke detection methods, it is possible to increase detection sensitivity, allowing the increase of both correct detections rate and the coverage area where the system can detect wildfires. In the following section we will present the evaluation process and methodology as well as the obtained results.

5 EVALUATION

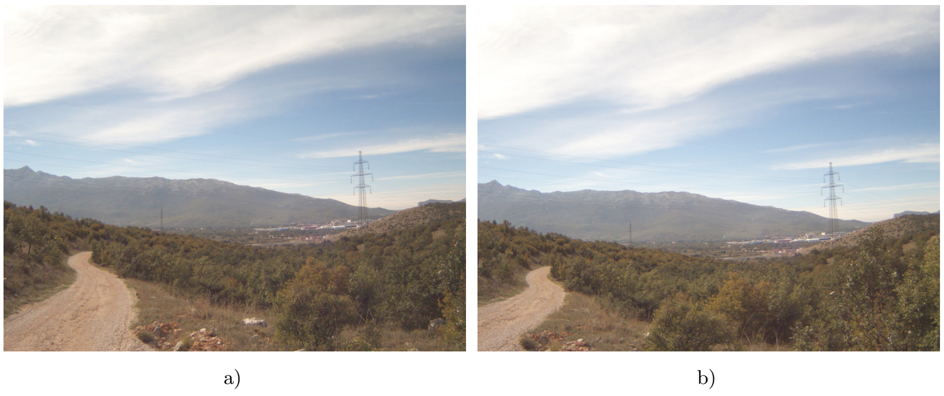


Figure 6. An example of stereo images captured simultaneously by both cameras. Input image taken by: a) the left stereo camera, b) the right stereo camera.

The proposed improvement based on stereo vision was implemented into three existing smoke detection methods. Every method was tested with and without this improvement on a database consisting of 18 856 images. Since all the video sequences were recorded using stereo cameras, the database actually consisted of altogether 37 712 images. These images were extracted from video sequences every 1 s, meaning that the total time span of the footage is approximately 5 hours and 15 minutes. In 9 198 images (approximately 2 hours and 30 minutes) smoke is visible in the image in its various forms. The remaining 9 658 images include various other phenomena that could induce false alarms (e.g., vegetation movement caused by wind, shadowing by clouds, changes in lightning conditions during sunrise or sunset). For a quality evaluation of the proposed improvement based on stereo vision, it is important to use both types of images in order to properly examine all the quality measures of the system. In Figure 6 we show a pair of stereo images that are a part of our database.

Video sequences were recorded on various locations and under different weather conditions to ensure diversity of the scenes. Smoke phenomena captured on video were located at different distances from the cameras to ensure that the evaluation takes into account possible errors of the stereo vision system (discretization and camera alignment errors).

Videos were recorded using two "`Elphel NC353L`" video cameras that were mutually synchronized using external synchronization cable (GPIO). This enabled the synchronization precision up to 1μs and simultaneous capturing of the images on both cameras. The focal length of the left camera was 4.657 mm, while the focal length of the right camera was 4.778 mm. In order to be able to use Equation (3) for our calculations, we first had to compensate this difference in focal lengths (as well as differences in principal point coordinates and skew and distortion parameters) by normalizing the pixel coordinates used as the input for the aforementioned equation. The images were captured in various resolutions. Both cameras were securely mounted on the ends of a wide bar, making the length of the baseline precisely 2 m. The maximum allowable percentage error of smoke area inside a pixel for the stereo system is set to $err_\vartheta = 21\,\%$. In Figure 7 a) we can see a close-up of the built stereo vision system, while in Figure 7 b) we show the system operating in the natural environment with predominant vegetation.

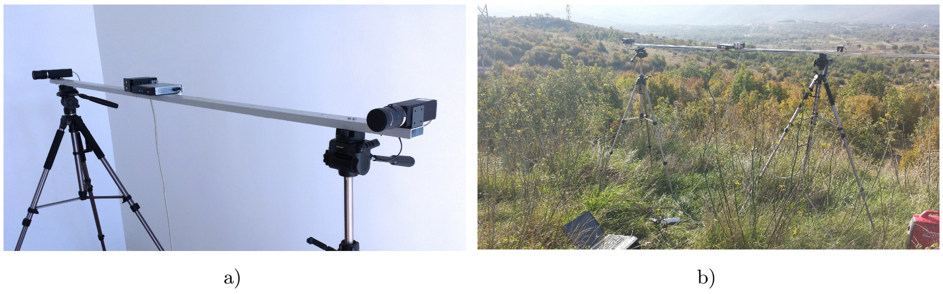


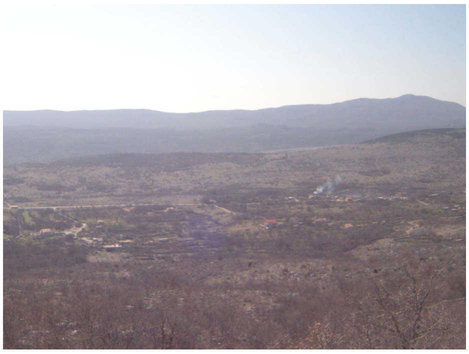
Figure 7. The stereo system used for evaluation, a) in laboratory environment, and b) in natural environment

As mentioned above, three existing smoke detection methods were used for the evaluation. First, each method was tested without any stereo modifications on all 18 856 images taken by the right camera. Second, improved methods were tested on the images of both cameras. In the following, we give a brief explanation of the methods used for evaluation.

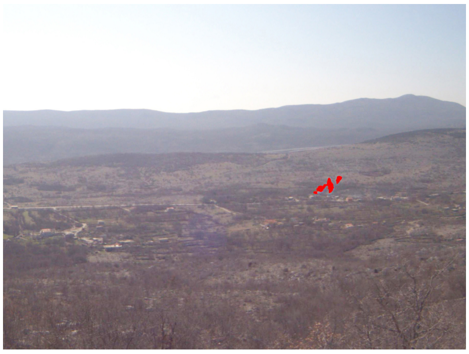
We have implemented *Method 1* presented in [5], which is a wavelet based real-time smoke detection method. The method consists of several detection phases. In the first phase, the moving pixels in the image are detected using a background estimation method. In the next step, the high frequency content of the image is analyzed in order to detect blurring due to the possible occurrence of smoke. Appearance of smoke gradually reduces the sharpness of the edges in the region until the region is completely covered by smoke. The region is analyzed in order to detect a decrease in local wavelet energy which would suggest presence of smoke. In the following phase, the regions are checked for decrease in the U and V channels. The appearance of smoke in the region results in the decrease in the chrominance level when compared to the estimated background. The next phase of the algorithm deals with analysis of the flickering effect that appears on the edges of the smoke contour. The analysis is carried out using temporal wavelet transforms. It is important to emphasize that this effect is noticeable in the short range smoke detection, and is not significant when the smoke is located at larger distances from the camera. The final phase of the algorithm examines the shape of the detected region in order to determine its convexity since general wildfire smoke tends to have a rather convex shape. In the case when the criteria from all the phases are satisfied, the algorithm raises an alarm.

The second method that we have implemented is the method presented in [18] denoted as *Method 2*. This method does not operate on the whole captured image, but the image is rather divided into blocks, or bins, which represent the smallest units for the detection process. The blue channel of each bin is observed over time and compared against the signal range, i.e. the difference between the maximum and the minimum bin value. The blue channel is selected since it exhibits greater sensitivity to smoke appearance then the other channels. In the case the difference between the current bin value and the referent bin value exceeds the percentage threshold based on signal range, the bins are considered as the candidate bins. The algorithm tracks the bins over a certain period of time in order to confirm the potential detection before raising an alarm. To produce the final alarm, there has to be a certain minimal number of candidate regions present over a set time period. Additionally smoke has to make a gradual appearance on the scene, so the regions that exhibit growth between two consecutive captured images that is larger than the maximum permitted growth are dismissed. Another possible filter for rejection of false alarms is the maximum permitted number of 8-connected components in the image. In the case all the conditions are satisfied over a predefined period of time, the system consequently raises an alarm.

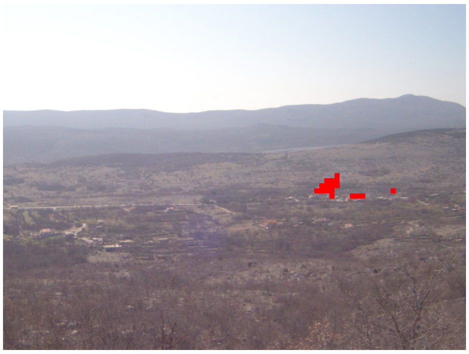
The third method is the method presented in [31] denoted as *Method 3*. This method consists of several different phases or stages of detection. The first phase is the image segmentation and classification phase where the different classes such as water and sky are used for the elimination of possible false alarms. Next, a motion detection phase is used to detect only the moving regions in the image, and thus reducing the amount of data that requires further processing. The following phase performs chromatic analysis where the current chromatic values are compared to the referent smoke-color values. The next step is the texture analysis phase where the regions are analyzed based on wavelet information. In the case of smoke appearance in the scene the region texture should change and there should be a loss in the high frequency range due to the blurring caused by smoke. In the following phase, the dynamic aspect of the candidate regions is examined. The regions that do not exhibit smoke-like behavior over a predefined time period are eliminated from the detection process. Finally, in case the candidate regions are confirmed in each detection phase, they are considered to be smoke, and the alarm is raised by the algorithm. The example detection images for all methods are shown in Figure 8.



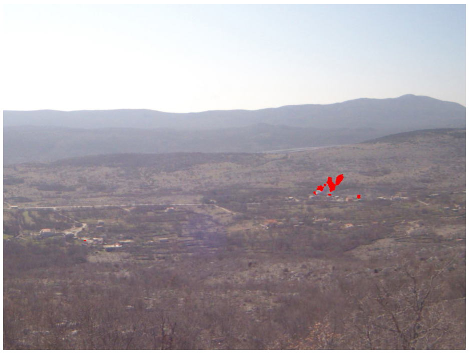
a)



b)



c)



d)

Figure 8. a) Input image with visible smoke, taken by the right stereo camera, b) image with smoke detected by *Method 1*, c) image with smoke detected by *Method 2*, d) image with smoke detected by *Method 3*

For each method, a stereo version was implemented based on the features presented in previous sections (the minimum size and overall dynamics of the candidate regions are examined to verify that they exhibit characteristic features of smoke). Here, we shall discuss the evaluation of each method and compare the results with the improved versions. We will use the evaluation measures for visual smoke detection systems presented in [32]. The evaluation is divided into two main evaluation aspects, global and local evaluation. Global measures evaluate algorithm performance based on the results where the elemental evaluation units are images. The evaluation is based on the algorithm output, where the algorithm decides whether smoke is present in the image. We can use four measures describing different aspect of detection quality: measure correct detections, also known as recall or true positive rate (TPR), specificity or true negative rate (TNR), false positive rate (FPR) and false negative rate (FNR). The measures are defined as:

$$TPR = \frac{TP}{TP + FN}, \tag{17}$$

$$TNR = \frac{TN}{TN + FP}, \tag{18}$$

$$FPR = \frac{FP}{TN + FP}, \tag{19}$$

$$FNR = \frac{FN}{TP + FN} \tag{20}$$

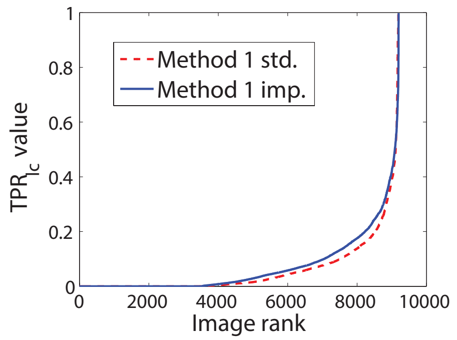
where TP denotes the number of true positive detections, FN represents the number of false negative detections, TN represents the number of true negative detections, and FP represents the number of false positive detections.

The results for global measures for all methods are presented in Table 1.

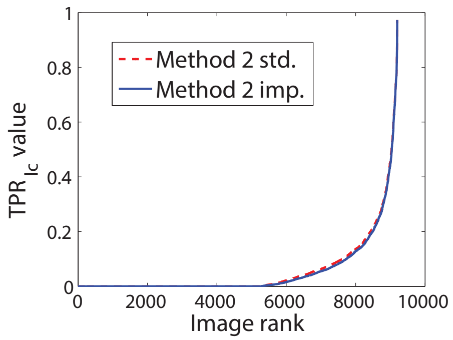
	TPR	TNR	FPR	FNR
Method 1 – standard	0.6601	0.8701	0.1299	0.3399
Method 1 – stereo	0.6677	0.9622	0.0378	0.3323
Method 2 – standard	0.4322	0.9994	0.0006	0.5678
Method 2 – stereo	0.4240	0.9996	0.0004	0.5760
Method 3 – standard	0.6954	0.8884	0.1116	0.3046
Method 3 – stereo	0.7024	0.9358	0.0642	0.2976

Table 1. Results for global measures for all methods

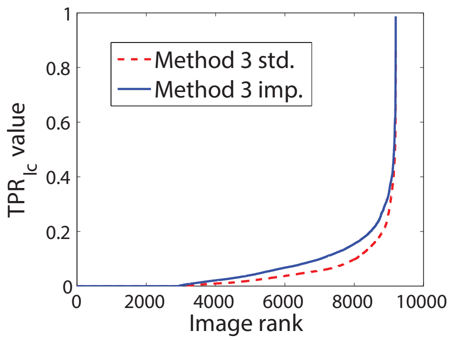
The results are obtained from all evaluation sequences. The stereo versions of the evaluated methods show a general improvement for most evaluation measures, especially a decrease in false alarms with similar or improved correct detections. The decrease in false alarms is a result of region size estimation based on the stereo distance calculation. Majority of false alarm sources are eliminated with this process, such as movements of the vegetation or similar phenomena in the close proximity of the camera. Most of the smoke detection methods have a set of tunable parameters which define the sensitivity of detection with respect to the dynamics in the environment. Increase in the algorithm sensitivity results in a more prompt and precise detection of actual smoke in the scene, however, it also results in a general increase in false alarms. Reducing the sensitivity of the algorithm, on the other hand, decreases false alarm rate, but increases the risk of missed detections. In the case when a tool for elimination of false alarms is introduced, such as stereo distance estimation, it is possible to increase the sensitivity of the algorithm. The possible increase in false alarms is in this case compensated with the false alarms elimination process. That is the reason why some of the results also show the increase in correct detections (recall), as it is the case with *Method 1* and *Method 3*. The exception is a slight decrease in correct detections for *Method 2* due to the fact that the standard version has a very low false alarm rate to begin with. Please note that by increasing the detection sensitivity we also increase the coverage area, i.e., the maximal distance at which the method can detect wildfire.



a)



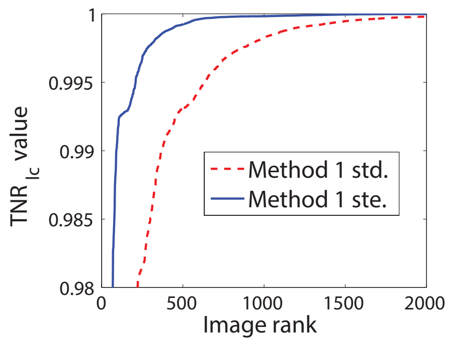
b)



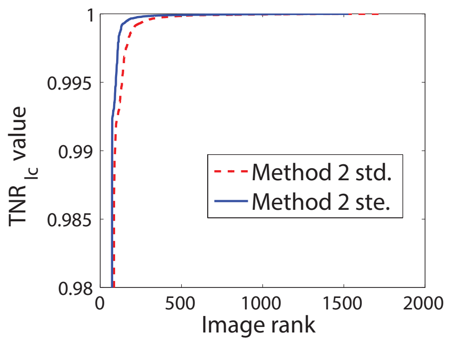
c)

Figure 9. Observer quality graphs for TPR_{lc} measure: a) *Method 1*, b) *Method 2*, and c) *Method 3*

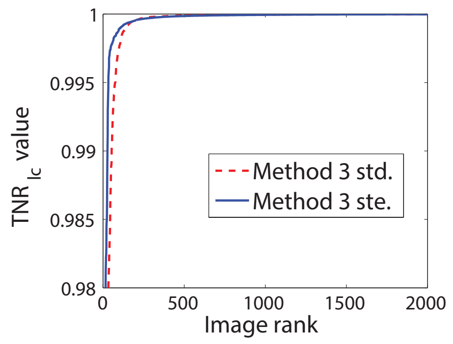
Another type of evaluation is performed using local measures. Local measures are based on the results where the smallest units of detection are individual pixels. Global measures are focused on whether the smoke is detected in the image or not, while the local measures are focused on whether the location of the smoke in the image is correct or not. Generally, when evaluating smoke detection systems, the most important fact is that the alarm is raised with the occurrence of smoke, with as low as possible false alarm rate. The location of the smoke in the image is also important, however, the global performance is the primary criterion. We have performed the evaluation on the local scale for measures described earlier, for all detection methods. The results are presented in the form of observer quality



a)



b)

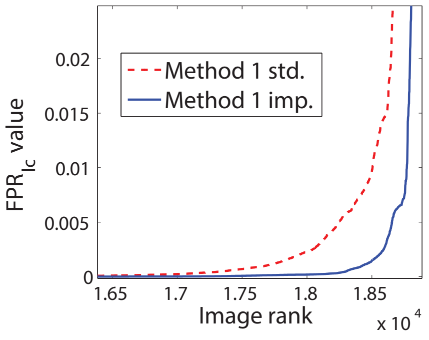


c)

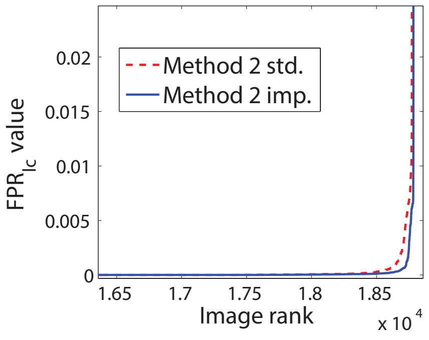
Figure 10. Observer quality graphs for TNR_{lc} measure: a) *Method 1*, b) *Method 2*, and c) *Method 3*

graphs introduced in [32]. The graphs show the value of the specific measure for all the images in the collection, sorted according to the increasing measure values. In the graphs, the y axis represents the value of the specific measure, while the x axis represents the ordinal number of the image in the sorted sequence, or the image rank. Figures 9–12 show the observer quality graphs for local measures of correct detections or true positive rate (TPR_{lc}), true negative rate (TNR_{lc}), false positive rate (FPR_{lc}) and false negative rate (FNR_{lc}), respectively.

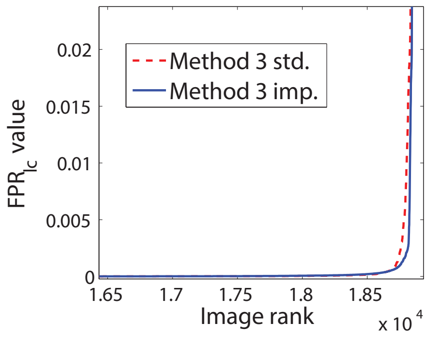
The local correct detections measure (TPR_{lc}) deals with correctly detected smoke pixels within the images in the collection. The stereo versions of the detection methods show same or better local accuracy of correct detections when compared to the standard variants as seen in Figure 9. When comparing the results for local specificity measures (TNR_{lc}), all of the stereo methods exhibit an increase in valid local rejections when compared to the standard versions as shown in Figure 10. This effect is the result of elimination of certain categories of false alarms based



a)



b)



c)

Figure 11. Observer quality graphs for FPR_{lc} measure: a) *Method 1*, b) *Method 2*, and c) *Method 3*

on stereo distance estimation. Similarly, this also affects the results concerning local false alarms where all stereo methods exhibit a drop in local false alarms as shown in Figure 11. Finally, the measure for local false negative rate (FNR_{lc}) for stereo versions remains the same or exhibits a drop in missed detections as shown in Figure 12.

Another measure that can be applied to the evaluation of smoke detection algorithms on the local scale is the Matthews correlation coefficient [33]. The Matthews correlation coefficient (mcc) is defined by:

$$mcc = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{21}$$

and is generally used as a quality measure for binary classifications. This measure takes into account all the detection cases (TP, TN, FP, FN) in a balanced manner

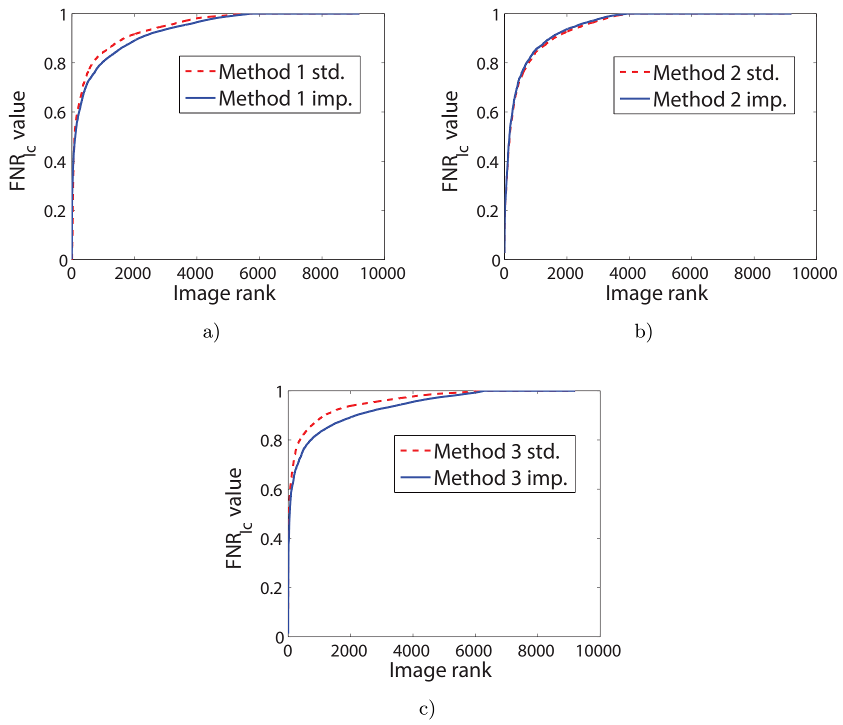


Figure 12. Observer quality graphs for FNR_{lc} measure: a) *Method 1*, b) *Method 2*, and c) *Method 3*

so it can be used even when the classes are of very different sizes. The results for the *mcc* measure for all methods are shown in Figure 13.

The *mcc* measure result values are in the interval $[-1, 1]$. The results for the stereo versions of the detection methods show the same or increased value when compared to the standard versions, which indicates an improvement in the overall local detection quality.

6 CONCLUSIONS

In this paper we proposed a novel approach to smoke detection by introducing stereo vision to the detection process. Standard smoke detection techniques available in literature, can be extended in a way to include stereo vision techniques allowing us to estimate the real world sizes of the detected regions and express them in standard units of measurement.

This provides the detection system with additional information that could be used to improve the reliability of the detection process. Using this information, the

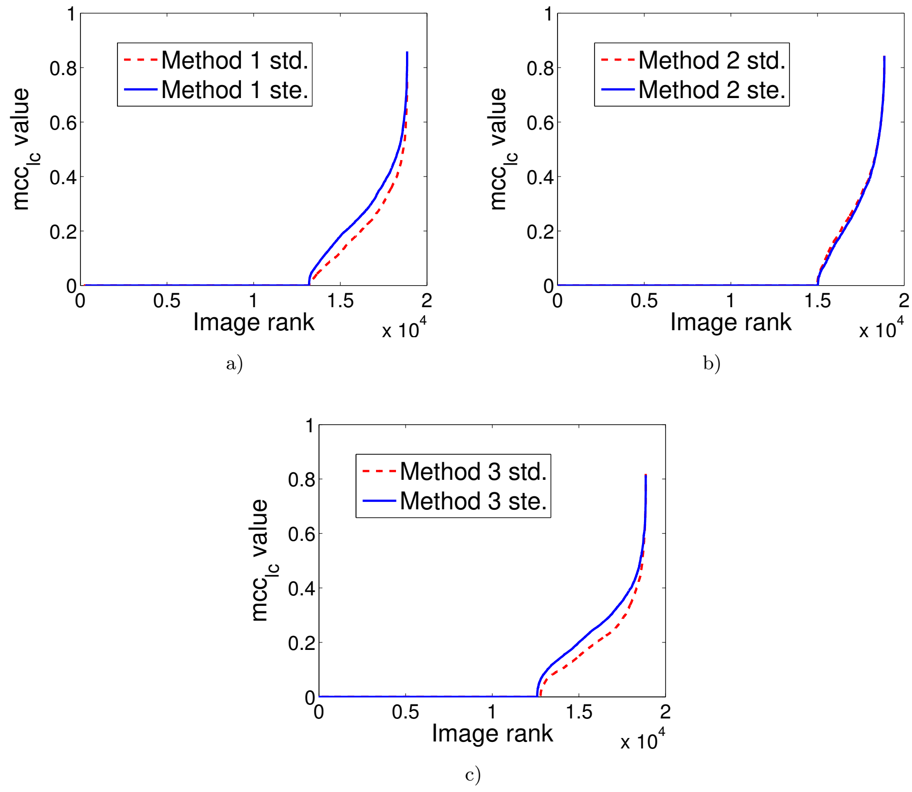


Figure 13. Observer quality graphs for mcc_{lc} measure: a) *Method 1*, b) *Method 2*, and c) *Method 3*

minimum size and overall dynamics of the candidate regions are examined to verify that they exhibit characteristic features of smoke. In this way, it is possible to determine the phenomena that cause false alarms due to their resemblance to smoke and eliminate them from the detection based on size and dynamics constraints. As an example, minimum size constraint is used for the elimination of false alarms induced by movement of small objects in the scene (such as movement of branches and grass in the close vicinity of the camera). Additionally, smoke dynamics are analyzed to verify that regions adhere to certain rules regarding the rate of spread, leading to the elimination of false alarms induced by phenomena that have significantly different dynamics characteristics (such as shadowing by clouds).

Since smoke detection implies estimating relatively large distances, we have proposed a wide-baseline stereo vision system. We have analyzed possible errors that might affect the accuracy of the stereo system as well as their impact on the actual depth estimation. These errors include discretization error, that is a result of an approximation due to the discrete nature of the imaging system, and camera alignment errors, more specifically the errors due to roll, pitch and yaw of one of the stereo cameras.

We have evaluated three existing smoke detection methods and compared them to the newly implemented stereo versions. The results based on global and local evaluation measures have shown improved overall performance, especially in false alarms for all tested methods. Moreover, by increasing the detection sensitivity we not only achieved a lower false alarms rate, but also either maintained or increased the number of correct detections and the coverage area of the existing methods. This shows that this improvement based on stereo vision could be used as a welcome addition to the standard detection methods in terms of system reliability and could also be taken into account when designing future smoke detection systems, especially if intended for areas with a high risk of wildfires.

REFERENCES

[1] Dukuzumuremyi, J. P.—Zou, B.—Hanyurwimfura, D.: A Novel Algorithm for Fire/Smoke Detection Based on Computer-Vision. International Journal of Hybrid Information Technology, Vol. 7, 2014, No. 3, pp. 143–154, doi: 10.14257/ijhit.2014.7.3.15.

[2] Morimitsu, N.: Image Processing Apparatus. US Patent Application US20120133739A1, May 31, 2012.

[3] Bugarić, M.—Jakovčević, T.—Stipaničev, D.: Adaptive Estimation of Visual Smoke Detection Parameters Based on Spatial Data and Fire Risk Index. Computer Vision and Image Understanding, Vol. 118, 2014, pp. 184–196, doi: 10.1016/j.cviu.2013.10.003.

[4] Cappellini, V.—Mattii, L.—Mecocci, A.: An Intelligent System for Automatic Fire Detection in Forests. Third International Conference on Image Processing and Its Applications, July 1989, pp. 563–570, doi: 10.1007/3-540-51815-0_67.

[5] Toreyin, B.—Dedeoglu, Y.—Cetin, A.: Wavelet Based Real-Time Smoke Detection in Video. Proceedings of the 13th European Signal Processing Conference (EUSIPCO 2015), Vol. 20, 2005, pp. 1–4.

[6] Xu, Z.—Xu, J.: Automatic Fire Smoke Detection Based on Image Visual Features. Proceedings of the International Conference on Computational Intelligence and Security Workshops (CISW 2007), 2007, pp. 316–319, doi: 10.1109/CISW.2007.4425500.

[7] Kim, D.—Wang, Y.-F.: Smoke Detection in Video. Proceedings of WRI World Congress on Computer Science and Information Engineering, 2009, pp. 759–763, doi: 10.1109/CSIE.2009.494.

[8] Ochoa-Brito, A.—Millan-Garcia, L.—Sanchez-Perez, G.—Toscano-Medina, K.—Nakano-Miyatake, M.: Improvement of a Video Smoke Detection Based on Accumulative Motion Orientation Model. Proceedings of the Electronics, Robotics and Automotive Mechanics Conference (CERMA 2011), 2011, pp. 126–130.

[9] Ho, C.—Kuo, T.: Real-Time Video-Based Fire Smoke Detection System. Proceedings of the International Conference on Advanced Intelligent Mechatronics, 2009, pp. 1845–1850.

[10] Ma, L.—Wu, K.—Zhu, L.: Fire Smoke Detection in Video Images Using Kalman Filter and Gaussian Mixture Color Model. Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI), Vol. 1, 2010, pp. 484–487, doi: 10.1109/AICI.2010.107.

[11] Piccinini, P.—Calderara, S.—Cucchiara, R.: Reliable Smoke Detection in the Domains of Image Energy and Color. Proceedings of the 15th IEEE International Conference on Image Processing (ICIP 2008), 2008, pp. 1376–1379, doi: 10.1109/ICIP.2008.4712020.

[12] Krstinić, D.—Stipaničev, D.—Jakovčević, T.: Histogram-Based Smoke Segmentation in Forest Fire Detection System. Information and Technology Control, Vol. 38, 2009, No. 3, pp. 237–244.

[13] Anton, M.—Olga, K.: Real-Time Smoke Detection in Video Sequences: Combined Approach. In: Maji, P., Ghosh, A., Murty, M. N., Ghosh, K., Pal, S. K. (Eds.): Pattern Recognition and Machine Intelligence (PReMI 2013). Springer Berlin, Heidelberg, Lecture Notes in Computer Science, Vol. 8251, 2013, pp. 445–450.

[14] Busch, A.—Boles, W. W.: Texture Classification Using Multiple Wavelet Analysis. Proceedings of the Sixth Digital Image Computing – Techniques and Applications Conference (DICTA 2002), 2002, pp. 341–345.

[15] Kara, B.—Watsuji, N.: Using Wavelets for Texture Classification. Proceedings of the International Conference on Signal Processing, 2003, pp. 920–924.

[16] Yu, C.—Zhang, Y.—Fang, J.—Wang, J.: Texture Analysis of Smoke for Real-Time Fire Detection. Proceedings of the Second International Workshop on Computer Science and Engineering (WCSE '09), 2009, pp. 511–515.

[17] Rafiee, A.—Dianat, R.—Jamshidi, M.—Tavakoli, R.—Abbaspour, S.: Fire and Smoke Detection Using Wavelet Analysis and Disorder Characteristics. Proceedings of the 3rd International Conference on Computer Research and Development (ICCRD), 2011, pp. 262–265, doi: 10.1109/ICCRD.2011.5764295.

[18] Fernández-Berni, J.—Carmona-Galán, R.—Carranza-González, L.: A Vision-Based Monitoring System for Very Early Automatic Detection of Forest Fires. Proceedings of the First International Conference on Modelling, Monitoring and Management of Forest Fires (FIVA 2008), 2008, pp. 161–170, doi: 10.2495/FIVA080171.

[19] Chen, J.—You, Y.—Peng, Q.: Dynamic Analysis for Video Based Smoke Detection. International Journal of Computer Science Issues, Vol. 10, 2013, No. 2, pp. 298–304.

[20] Cui, Y.—Dong, H.—Zhou, E.: An Early Fire Detection Method Based on Smoke Texture Analysis and Discrimination. Proceedings of the Congress on Image and Signal Processing (CISP '08), 2008, pp. 95–99, doi: 10.1109/CISP.2008.397.

[21] Ko, B. C.—Kwak, J.-Y.—Nam, J.-Y.: Wildfire Smoke Detection Using Temporospatial Features and Random Forest Classifiers. Optical Engineering, Vol. 51, 2012, Art. No. 017208, doi: 10.1117/1.OE.51.1.017208.

[22] Yang, J.—Chen, F.—Zhang, W.: Visual-Based Smoke Detection Using Support Vector Machine. Proceedings of the Fourth International Conference on Natural Computation (ICNC '08), 2008, pp. 301–305, doi: 10.1109/ICNC.2008.219.

[23] Stipaničev, D.—Šerić, L.—Braović, M.—Krstinić, D.—Jakovčević, T.—Štula, M.—Bugarić, M.—Maras, J.: Vision Based Wildfire and Natural Risk Observers. Proceedings of the 3rd International Conference on Image Processing Theory, Tools and Applications (IPTA), OS1: Special Session on Image Processing for Natural Risks (IPNR), 2012.

[24] Mrovlje, J.—Vrančić, D.: Distance Measuring Based on Stereoscopic Pictures. 9th International Ph.D. Workshop on Systems and Control: Young Generation Viewpoint, 2008.

[25] Gallup, D.—Frahm, J.-M.—Mordohai, P.—Pollefeys, M.: Variable Baseline/Resolution Stereo. 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008, pp. 1–8.

[26] Vrancic, D.—Smith, S. L.: Permanent Synchronization of Camcorders via LANC Protocol. In: Woods, A. J., Bolas, M. T., McDowall, I. E., Dodgson, N. A., Merritt, J. O. (Eds.): Stereoscopic Displays and Virtual Reality Systems XIII. SPIE Proceedings, Vol. 6055, 2006, pp. 165–176.

[27] Lewis, J. P.: Fast Normalized Cross-Correlation. Vision Interface, Vol. 10, 1995, No. 1, pp. 120–123.

[28] Sahabi, H.—Basu, A.: Analysis of Error in Depth Perception with Vergence and Spatially Varying Sensing. Computer Vision and Image Understanding, Vol. 63, 1996, No. 3, pp. 447–461.

[29] Zhao, W.—Nandhakumar, N.: Effects of Camera Alignment Errors on Stereoscopic Depth Estimates. Pattern Recognition, Vol. 29, 1996, No. 12, pp. 2115–2126.

[30] Bugarić, M.—Jakovčević, T.—Stipaničev, D.: Computer Vision Based Measurement of Wildfire Smoke Dynamics. Advances in Electrical and Computer Engineering, Vol. 15, 2015, No. 1, pp. 55–62.

[31] Jakovčević, T.—Stipaničev, D.—Krstinić, D.: Visual Spatial-Context Based Wildfire Smoke Sensor. Machine Vision and Applications, Vol. 24, 2013, No. 4, pp. 707–719, doi: 10.1007/s00138-012-0481-x.

[32] Jakovčević, T.—Šerić, L.—Stipaničev, D.—Krstinić, D.: Wildfire Smoke-Detection Algorithms Evaluation. In: Viegas, D. X. (Ed.): Proceedings of the VI International Conference on Forest Fire Research, 2010, pp. 1–12.

[33] Matthews, B. W.: Comparison of the Predicted and Observed Secondary Structure of T4 Phage Lysozyme. Biochimica et Biophysica Acta (BBA) – Protein Structure, Vol. 405, 1975, No. 2, pp. 442–451, doi: 10.1016/0005-2795(75)90109-9.

Toni Jakovčević graduated in computer science from the University of Split, Croatia, and afterwards received his Ph.D. in 2011. His research interests include artificial intelligence, image analysis and programming languages. Currently, as Assistant Professor, he conducts research, development and teaching in the University of Split.

Marin Bugarić holds the position of a young researcher at the University of Split, Croatia. He received his Ph.D. in 2013. His research interests include early forest fire detection systems based on visual smoke detection, GIS applications, augmented reality systems, forest fire risk indexes, usability, design and analysis of the secure authentication protocols.

Darko Stipaničev is Professor of computer science and automatic control at the University of Split. His research interests include complex systems modeling and control, intelligent systems analyses and design, digital image analyses, advanced Internet technologies, and recently, the application of ICT in environmental protection, particularly wildfires prevention and management, including vision-based wildfires smoke detection.

Computing and Informatics, Vol. 37, 2018, 509–532, doi: 10.4149/cai_2018_2_509

ASHUR: EVALUATION OF THE RELATION SUMMARY-CONTENT WITHOUT HUMAN REFERENCE USING ROUGE

Alan Ramírez-Noriega, Reyes Juárez-Ramírez
Samantha Jiménez, Sergio Inzunza

Universidad Autónoma de Baja California
Facultad de Ciencias Quimicas e Ingeniería
Calzada Universitaria 14418, Parque Industrial Internacional
Tijuana, Baja California, C.P. 22390 México
e-mail: {alan.david.ramirez.noriega, reyesjua, samantha.jimenez, sinzunza}@uabc.edu.mx

Yobani Martínez-Ramírez

Universidad Autónoma de Sinaloa
Facultad de Ingeniería Mochis
Fuente de Poseidon y Angel Flores s/n, Col. Jiquilpan
Los Mochis, Sinaloa, C.P. 81223 México
e-mail: yobani@uas.edu.mx

Abstract. In written documents, the summary is a brief description of important aspects of a text. The degree of similarity between the summary and the content of a document provides reliability about the summary. Some efforts have been done in order to automate the evaluation of a summary. ROUGE metrics can automatically evaluate a summary, but it needs a model summary built by humans. The goal of this study is to find a quantitative relation between an article content and its summary using ROUGE tests without a model summary built by humans. This work proposes a method for automatic text summarization to evaluate a summary (ASHuR) based on extraction of sentences. ASHuR extracts the best sentences of an article based on the frequency of concepts, cue-words, title words, and sentence length. Extracted sentences constitute the essence of the article; these sentences construct the model summary. We performed two experiments to assess the reliability of ASHuR. The first experiment compared ASHuR against similar approaches based on sentences extraction; the experiment placed ASHuR in the first place in each applied test. The second experiment compared ASHuR against human-made summaries, which yielded a Pearson correlation value of 0.86. Assessments made to ASHuR show reliability to evaluate summaries written by users in collaborative sites (e.g. Wikipedia) or to review texts generated by students in online learning systems (e.g. Moodle).

Keywords: Text summarization, summary evaluation, ROUGE, sentences extraction

Mathematics Subject Classification 2010: 68-U15, 68-T50

1 INTRODUCTION

The objective of automatic text summarization is the reduction of an original text to a smaller number of sentences by means of a computer, while keeping the important ideas intact [8]. Many areas use automatic text summarization such as intelligent tutoring systems, telecommunication industry, information extraction, text mining, question answering, news broadcasting, and word processing tools [19, 30].

The information explosion on Internet requires a reduction in the amount of information size and an increase in information efficiency [30]. These activities become easier with automatic summarization because fewer lines may represent the most important information about a document. Thus, users can find the resources more quickly [2, 16].

A summary evaluation shows the high-points of the original text. Manual summary evaluation is the first option because human assessment guarantees achievement of the desired results. However, a text can have many useful summaries; these show the main disadvantages of a manual evaluation approach, as a different evaluator may not agree [20] in determining the correct summary. The manual comparison of peer summaries based on model summaries is an activity that requires much effort and time [25].

Development of evaluation methods for summarization is difficult. Human summaries vary for many reasons such as knowledge, biases, goals, and the intended audience [23]. There are methods to evaluate summaries such as ROUGE [12], BE [9], and Pyramid [23]. They are widely used in summarization to analyze summary content [3]. These methods need human impact to work efficiently, and are considered semi-automatic [16, 18].

Previous methods require a model summary or a set of model summaries to function. The extraction of a model summary is a time-consuming and expensive task [17]. It is necessary to have an ideal summary and the original text to automate this process in these evaluation systems completely.

The purpose of this article is to evaluate a summary without the human model input. Two phases divide the process: The first phase extracts the most representative sentences from the content through an algorithm based on frequencies of concepts, cue-words, title words, and sentence length. Despite being simple and not requiring an in-depth level of knowledge analysis, this technique is suitable for building summaries [16]. The second phase evaluates the original summary based on ROUGE metrics and the built summary in the first phase. The system is called ASHuR (Assessing Summaries without Human reference using ROUGE).

The remainder of the article is structured as follows: Section 2 describes related works. Section 3 explains related topics such as text summarization and tests to evaluate summaries. Section 4 outlines the proposed approach. Sections 5 and 6 describe two experiments together with the results and discussions. The final sections show conclusions and references.

2 RELATED WORK

There are studies related to the evaluation of previous summaries that have dealt with this problem. These studies have faced this issue because of the importance of a summary in the field of education, and its ability to provide a general idea of a lengthy document.

In [11] the authors proposed an integrated method to evaluate summaries using Latent Semantic Analysis (LSA) automatically. This method is based on a regression equation calculated with a corpus of a hundred summaries. It is validated on a different sample of summaries. The equation incorporates two parameters extracted from LSA: semantic similarity and vector length. The aim of this study was to use a simple and innovative LSA-based computational method to evaluate summaries reliably. Despite the efforts made in this article, the authors needed a training set for their algorithms to work. The training set is only for a common topic, which is the limit of this particular idea; a summary of 50 words works in only a few cases. A summary, limited to that number of words excludes many other situations where the evaluation system could be used.

FRESA [29] is a Framework for Evaluating Summaries Automatically, which includes document-based summary evaluation measures based on probabilities distribution. FRESA supports different n-grams and skips n-grams probability distributions. In addition, this environment evaluates summaries in various languages. This framework is an alternative to ROUGE in evaluating summaries based especially on the Jensen-Shanon divergence. FRESA takes the original text as a model, without requiring human intervention, and compares it to the abstract obtained automatically. Their system extracts phrases in evaluating the summary, however, human summaries give bad evaluation results because FRESA considers complete coincidences in sentences. FRESA metrics based on divergence are not perceived clearly and quickly. The conclusion is that values of the metric give a high value of divergence between a text and its summary, this is always applicable to the phrases that are used in this system. Thus, FRESA associates values of great divergence regardless of the strategy used, including random compression. Therefore, there is not an adequate way of evaluating summaries [20].

Louis [18] presented and evaluated a suite of metrics which do not require gold-standard human summaries for evaluation. They proposed three evaluation techniques, two of which are model-free and do not rely on the gold standard for the assessment. The third technique improves standard automatic evaluations by expanding the set of available model summaries with chosen system summaries. SIMetrix is the tool used by these authors. The metrics of this system are based on the Kullback Leibler (KLD) and Jensen Shannon (JSD) divergence, in addition to the Fraction of Topic Words (FoTW). SIMetrix, is a very versatile system, and has a variety of tests to measure the relation between the summary and its content. Although SIMetrix shows good overall results in its tests, it has not excelled in the evaluation of summaries; ROUGE is the standard that is used in reporting automatic summarization evaluation results. However, SIMetrix is used in this investigation to validate summaries.

ROUGE is the evaluation system implemented as the de-facto standard; it is the most commonly used metric of content selection quality used in research papers because it is cheap and fast [21]. ASHuR evaluates a summary based on sentences extraction considering ROUGE as the evaluation system. This is an advantage that the related work does not have.

3 FUNDAMENTALS FOR TEXT SUMMARIZATION

3.1 ROUGE

ROUGE stands for Recall-Oriented Understudy for Gisting Evaluation. ROUGE is a summary evaluation method that includes measures to automatically determine the quality of a summary by comparing it to other (ideal) summaries created by humans [12].

This method has the following tests [13]:

- ROUGE-N: N-gram Co-Occurrence Statistics (versions ROUGE-1, ROUGE-2, ROUGE-3, ROUGE-4).
- ROUGE-L: Longest Common Subsequence.
- ROUGE-W: Weighted Longest Common Subsequence.
- ROUGE-S: Skip-Bigram Co-Occurrence Statistics.
- ROUGE-SU: Extension of ROUGE-S.

Document Understanding Conference (DUC), National Institute of Standards and Technology (NIST), and Text Analysis Conference (TAC) adopted ROUGE package for content-based evaluation [14, 27, 26, 28]. ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU tests have been used in many investigations to evaluate experiments because they have a greater accord with the human evaluation [14, 26, 3, 27].

The typical information retrieval metrics are precision and recall [21], these metrics are used by ROUGE to evaluate summaries [12]. Precision (Equation (1)) is the number of sentences occurring in both the system and ideal summary divided by the number of sentences in the system summary. Recall (Equation (2)) is the number of sentences occurring in both the system and ideal summary divided by the number of sentences in the model summary [27].

$$\text{precision} = \frac{|\{\text{relevantObjects}\} \cap \{\text{retrievedObjects}\}|}{|\text{retrievedObjects}|}, \tag{1}$$

$$\text{recall} = \frac{|\{\text{relevantObjects}\} \cap \{\text{Objects}\}|}{|\text{relevantObjects}|}, \tag{2}$$

$$F_\beta = (1+\beta^2) * \frac{\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) + \text{recall}}. \tag{3}$$

The appeal of precision and recall as an evaluation measure is that after a human defines the gold standard sentence selection, it can be repeatedly used to evaluate automatically produced summaries by a simple comparison of sentence identifiers [21]. F-measure (Equation (3)) is a weighted harmonic mean of recall and precision. Where β is a variable to give preference either recall or precision, when $\beta > 1$ then the preference is given to precision, and when $\beta < 1$ then the preference is given to recall. This study used the F-measure for experiments.

3.2 SIMetrix

SIMetrix tool is a group of metrics to evaluate summaries [18]. Our investigation uses the SIMetrix model without a model summary.

The following SIMetrix metrics validate our proposal [18]:

- KLInputSummary: Kullback Leibler divergence between input and summary
- KLSummaryInput: Kullback Leibler divergence between summary and input. Since KL divergence is not symmetric, the features are computed both ways Input-Summary and Summary-Input. Both features above use smoothing.
- UnsmoothedJSD: Jensen Shannon (JS) divergence between input and summary. No smoothing.
- SmoothedJSD: A version with smoothing.
- CosineAllWords: Cosine similarity between all words in the input and summary.
- PercentTopicTokens: Proportion of tokens in the summary that are topic words of the input.

- FractionTopicWords: The fraction of topic words of the input that appear in the summary.
- TopicWordOverlap: Cosine similarity using all words of the summary but only the topic words from the input.

SIMetrix results showed that the strength of features vary considerably. The best metric is JS divergence, which compares the distribution of terms in the input and summary. According to the SIMetrix documentation, higher divergence scores indicate poor quality summaries. For the other metrics, higher scores indicate better summaries.

4 PROPOSED APPROACH

The proposed approach initially divides the article into its summary and its content. The system constructs the summary model based on the original content. Finally, ROUGE evaluates the model summary and the summary of the original article to obtain the summary assessment. Figure 1 displays this process.

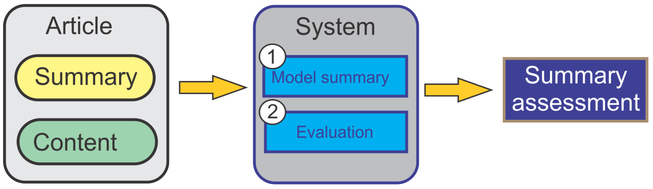


Figure 1. General diagram of the ASHuR evaluation process

4.1 Model Summary Module

This module creates a model summary, the following steps details the process:

1. Identification of raw sentences: This step obtains raw sentences from the content. A raw sentence is one taken from the original text without any special treatment. The ASHuR process begins with this set of sentences; after a process of cleaning, splitting, and scoring of the sentences, our system takes sentences from original raw sentences to produce the summary.
2. Determination of concepts frequencies: ASHuR applies a text cleaning process to raw sentences. Such process involves the following phases:

   - Tokenize sentences: The tokenization breaks down the sentences into a set of words [8] called tokens. The token is the minimal unit to analyze the text in this study.

- Delete stop-words: Stop-words are words that are insignificant in our method. Therefore, ASHuR eliminates stop-words from the original text. The stop-words list includes the most frequently occurring words in a text (e.g. a, the, of, etc.) [5].
- Apply stemming: The stemming technique uses the root form of a word. The primary objective is to assign equal importance to words having the same root. Thus, words expressed in their different forms are considered to be the same [8]. Our proposal uses Porter's algorithm to apply stemming; this is the most common method used in literature [24].

ASHuR gains word frequencies after the cleaning process. This information is useful to assess the impact of the sentence in the document. This phase obtains a processed version of raw sentences.

3. Identification of the article title: Words in the title always represent the main idea of the text. The title plays a particular role in ASHuR because sentences that have title words are more important than other sentences. The title follows the same cleaning process as the rest of the text.
4. Definition of signal words: This phase uses a technique where phrases or words determine the relevance of a sentence, these words are called signal words. There are different kinds of signal words, however ASHuR works with words related to importance such as greatness, conclusion, summary, etc. [16]. These words may be a good indicator of relevant information [4, 27]. This study employs a list of signal words based on [10].
5. Calculation of the sentences score: This phase calculates the score of each sentence based on frequencies and the amount of words. Title words and signal words found in the sentence also proportionally influence the score.
6. Selection of the best sentences: This phase chooses the sentences with the highest score while discarding the sentences which are too short. These sentences are in order according to their score. The total number of words in a sentence must be similar to the number of words of the original summary. ASHuR selects sentences representing the summary of the version of raw sentences.

4.2 Evaluation Module

The first module of the summarization system generates the summary of the original article. ROUGE metrics then compare the generated summary with the model summary. Figure 2 represents the complete process of ASHuR.

For the evaluation part of the process, this study employs the following ROUGE tests: ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU. This study calculates the mean result of ROUGE tests to obtain a single result, however, another option could be to take a ROUGE test to represent the evaluation of the summary. We consider that ROUGE is a useful tool for the tasks assessment and that a new algorithm for this assessment is not necessary.

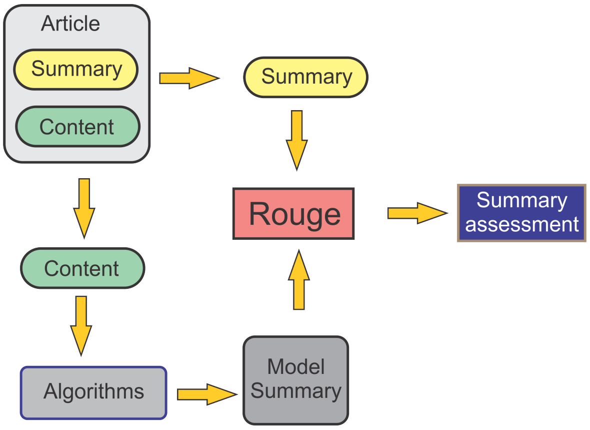


Figure 2. Specific diagram of the ASHuR evaluation process

4.3 Formal Representation of the Model

The process of ASHuR is represented in the following definition and equations:

Definition 1. Let $R_{procesed} = \{T_1, \dots, T_{|R_{procesed}|}\}$ be the set that represents an article R_{raw} with a text cleaning process in sentences. Each element in $R_{procesed}$ has the form $T_j = \{t_1, \dots, t_{|T|}\}$, where elements in T represent words in a sentence.

Definition 2. Let $S = \{c_1, \dots, c_{|S|}\}$ be the set that represents the score of sentences found in a document, where each element in S represents an ordered pair of the form $c_i = (r, d)$, the element r represents the score of the sentence and the element d represents the number of words in a sentence.

After of previous definitions, for each element $T \in R_{procesed}$, then $S_T \leftarrow (r_T, |T|)$, where the score is calculated by the Equation (4) based on Equations (5), (6), and (7). Equation (6) uses the variable a to represent a value for signal words, these words are represented by the set W. Equation (7) uses the variable b to represent a value for title words, these words are represented by the set I. The variable $|T|$ represents the number of elements in T.

$$r_T = f \cdot g \cdot l, \tag{4}$$

$$f = \sum_{t \in T} Freq(t)/|T|, \tag{5}$$

$$g = \begin{cases} a, & \text{if } |W \cap T| > 0, \\ 1, & \text{otherwise,} \end{cases} \tag{6}$$

$$l = \begin{cases} b, & \text{if } |I \cap T| > 0, \\ 1, & \text{otherwise.} \end{cases} \tag{7}$$

Let $S_{sort} = \{x_1, \ldots, x_{|S_{sort}|}\}$ be the set S ordered by the element r of the pair ordered x_k, sentences representing the summary of the article are taken from the set S_{sort}. The Algorithm 1 displays the process to obtain sentences that represent the summary. The variable a represents the sum of words in each x appended to $S_{summary}$, also, the variable max represents the maximum number of words of the summary and min represents then minimum number of words considered by sentence. The generated summary is represented in $S_{summary}$, this is used to evaluate other summaries.

Algorithm 1 Process to obtain the most important sentences

```
1: for all $x \in S_{sort}$ do
2:   if ($a <$ max) and ($x_d >$ min) then
3:     $a \leftarrow a + x_d$
4:     append $x$ to $S_{summary}$
5:   end if
6: end for
```

5 COMPARISON TO SIMILAR APPROACHES USING MODEL SUMMARY

5.1 Experiment

This experiment compares ASHuR to nine Summarization Systems (SS) based on sentence extraction. The aim is to assess the quality of the extracted sentences against similar approaches using model summaries. We selected SS as represented in Table 1 for the experiment because literature references to them and they are freely available.

None of the SS selected have algorithms available to be implemented. Only the applications have been published. Some of the systems are web applications, while others are applications for the Windows operating system. Others are applications for the Linux operating system. This setback complicates the automation of the evaluation process, therefore, the sample size for this iteration is not as extensive as desired.

This experiment uses research articles to perform the comparison between SS because expert researchers review these kind of documents before the publication, so that articles have quality in the abstract (summary) as well as the content. This experiment considers the abstract as the model summary of ROUGE.

Id	**System**
1	ASHuR
2	Autosummarizer [1]
3	Freesummarizer [6]
4	IBM Many Aspects Document Summarization Tool (furthest) [15]
5	IBM Many Aspects Document Summarization Tool (Greedyexp) [15]
6	IBM Many Aspects Document Summarization Tool (K-Median) [15]
7	IBM Many Aspects Document Summarization Tool (SVD) [15]
8	Online summarize tool [22]
9	Open text summarizer [31]
10	Swesum [7]

Table 1. Summarization systems

The test data is contained in 40 articles selected from the special issue "Social Identity and Addictive Behavior" in the Journal of Addictive Behaviors Reports [1], Volumes 1 (June 2015), 2 (December 2015), 3 (June 2016), 4 (December 2016), and 5 (June 2017). We chose this journal because it considers theoretical aspects with few equations that can hinder the work of summarization systems.

The preparation phase of documents deleted the abstract and the references, the rest of the article remained intact. The prepared papers were submitted to each SS to build its summary. The next phase compared generated summaries and the model summaries. Each algorithm made a summary per article which was contrasted with the corresponding original summary.

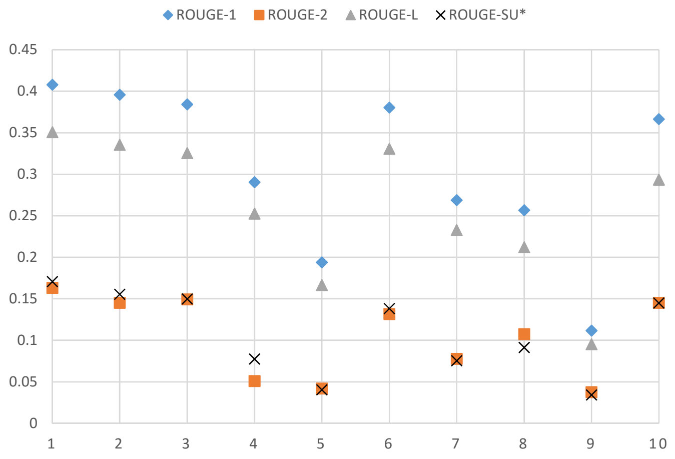
The eight tests of ROUGE evaluated results of SS considering the F-measure. The ROUGE tests result is a value between 0 and 1, the closer to one the better the summary.

5.2 Result

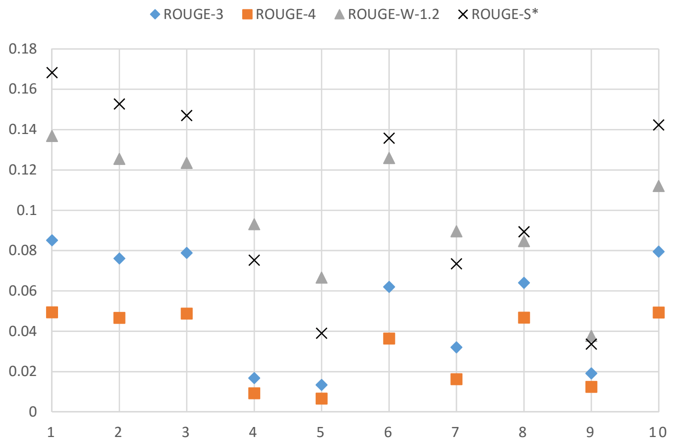
The results of the ROUGE evaluation applied to SS are displayed in Figure 3. Two groups organize the information; group 1 presents the most commonly used tests (see Figure 3 a)), and group 2 presents the rest of tests (see Figure 3 b)). The x-axis deploys Identifiers of SS and the y-axis represents the values reached by the tests. Graphs of results present ROUGE tests by a figure; rhombus, square, triangle, or cross, so, tests can be differentiated.

Means results obtained by SS in ROUGE tests are displayed in Figure 4. This figure shows the values reached in the x-axis and SS in the y-axis. The best-positioned systems are *ASHuR*, *Autosummarizer*, and *Freesummarizer* in that order. The worst positioned are *OpenTextSummarizer* and *IBM_GREEDYEXP*.

[1] `http://www.sciencedirect.com/science/journal/23528532/vsi`



a) Group 1



b) Group 2

Figure 3. Results of summarization systems considering ROUGE

5.3 Discussion

ASHuR obtained higher results in each test than the rest of SS (see Figure 3). This showed that our method achieved sentences more representative of the content of the original text.

Test files contained tables in text format, the systems positioned in the first places dealt with this point correctly. However, other systems such as *OpenTextSummarizer* had problems with the tables, which led to poor evaluation results.

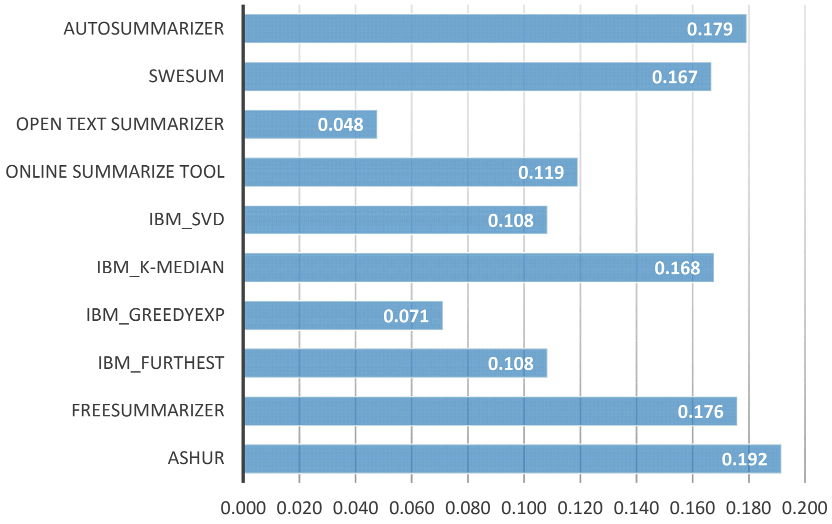


Figure 4. Means of summarization systems in ROUGE tests

It is complicated to achieve values close to one (the ideal value) in some ROUGE tests, but it is simpler in others tests. ROUGE-1 and ROUGE-L are tests that obtain higher scores than the rest of tests. ROUGE-4 is the most complicated test to overcome, this test on average had the lowest results.

The SS position of each ROUGE test varies a few places; accordingly, the ROUGE's tests maintain consistency and regularity in results, even though the score of the systems are similar. This means the summary evaluation of a system will not be in the first positions in a test and the last positions in another.

6 COMPARISON AGAINST HUMAN SUMMARY WITHOUT MODEL SUMMARY

6.1 Experiment

SS did not intervene in this second experiment because the first experiment verified that ASHuR obtains more precise results. The aim is to demonstrate that the ASHuR summary is similar to human summaries. This activity was realized with SIMetrix (Summary Input similarity Metrics) [18]. This tool analyses a text summary through similarity metrics (Section 3.2). SIMetrix is a system that allows, unlike ROUGE, to perform summary evaluations without a summary model. However, important conferences as DUC or TASC do not consider it relevant because they trust to the evaluation of ROUGE.

SIMetrix does not have ROUGE support, however, ROUGE needs a model summary to evaluate other texts. Thus, SIMetrix evaluates summaries in this experiment because it does not require a model summary. The objective of this project is to generate a version of ROUGE to assess abstracts without human intervention in the same way as SIMetrix but with the support of ROUGE.

This study focuses on unstructured documents such as Wikipedia documents. This experiment considers the Wikipedia branch in the category *Main topic classifications* for test data. This category is the main one in the hierarchy of Wikipedia. The rest of the categories is derived from this one. The main category has 10 sub-categories, and these contain other categories (see Table 2). This paper contemplates the direct categories of *Main topic classifications*. The categorization described corresponds to the Wikipedia version of October 1, 2016.

Categories		**Sub-Categories**	**Pages**
Main Topic Classifications		10	14
	Geography	26	75
	Nature	26	15
	Reference works	39	25
	Health	45	13
	History	32	27
	Philosophy	18	51
	Science and technology	9	7
	Humanities	33	49
	Mathematics	21	12
	People	34	2
	Total	286	290

Table 2. Category main topic classifications of Wikipedia

The main category and sub-categories in the Table 2 contain 290 articles. This experiment did not consider articles with the following characteristics:

1. Articles without a summary (e.g. the article Caribmap[2]).
2. Articles that describe a list of other pages (e.g. the article Lost History[3]).
3. Articles that are in two or more of the considered categories (e.g. the article People[4]). Articles that met the desired characteristics were 196.

The comparison process consisted of obtaining a summary of articles for each treatment (ASHuR and human) and comparing it with the content using SIMetrix. Firstly, ASHUR generated its summary, and this was compared with the content to obtain a summary-content relation measure. We then examined the original abstract of the article (human summary) with the content getting another measure of summary-content relation. The hypothesis is that a high positive correlation

[2] `https://en.wikipedia.org/wiki/Caribmap`

[3] `https://en.wikipedia.org/wiki/Lost_history`

[4] `https://en.wikipedia.org/wiki/People`

will be achieved using the Pearson test between both measures of relation. Each summary (ASHuR and Human) was evaluated against its content by 8 SIMetrix tests.

This experiment separated tests in two clusters according to their form of evaluation. Tests that consider that the closer to 0 a result is, the better correlation will exist (B1), and tests that consider that the closer to 1 a result is, the better correlation will exist (B2). The group B1 contemplates the KLInputSummary, KLSummaryInput, UnsmoothedJSD, and SmoothedJSD tests. Group B2 contemplates the CosineAllWords, PercentTopicTokens, FractionTopicWords, and TopicWordOverlap tests.

6.2 Results

The evaluation of ASHuR results and human summaries was contrasted 196 times, one for each article. Figure 5 shows the general results organized by test and treatment (ASHuR and human).

Boxplots represent the data distribution in summarized form in Figure 5, the vertical line inside the rectangle represents the data median. The x-axis represents the applied test and the treatment, ASHuR tests are described at the label end with the letter A, and human tests with the letter H. The y-axis shows the values scale of tests; these vary according to the group of applied tests.

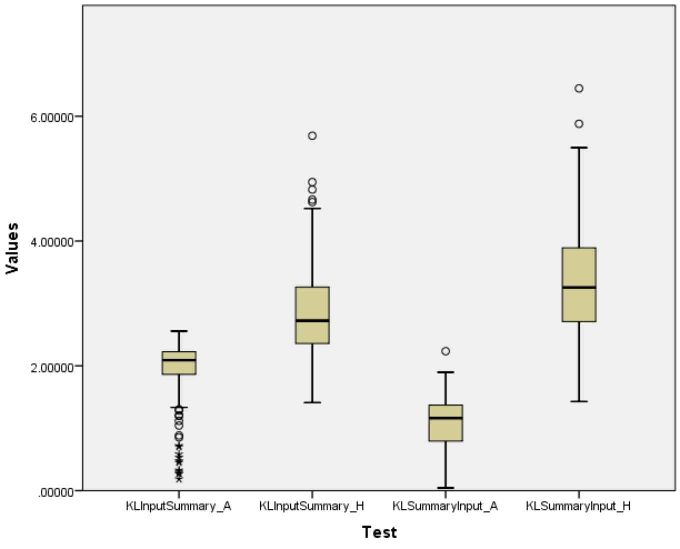
Results of Figure 5 a) represent block B1 tests, the closer to zero the means, the better the summary will be evaluated. Figure 5 b) displays results of block B2 tests ASHuR obtained values closer to zero in each of the tests, however, it also got more outliers.

The closer to 1 the means of block B2 are, the summaries will be better. The best-performing tests are CosineAllWords and TopicWorldOverlap of ASHuR. Tests evaluate summaries based on different aspects; this causes some tests to obtain results closer to zero and others more distant.

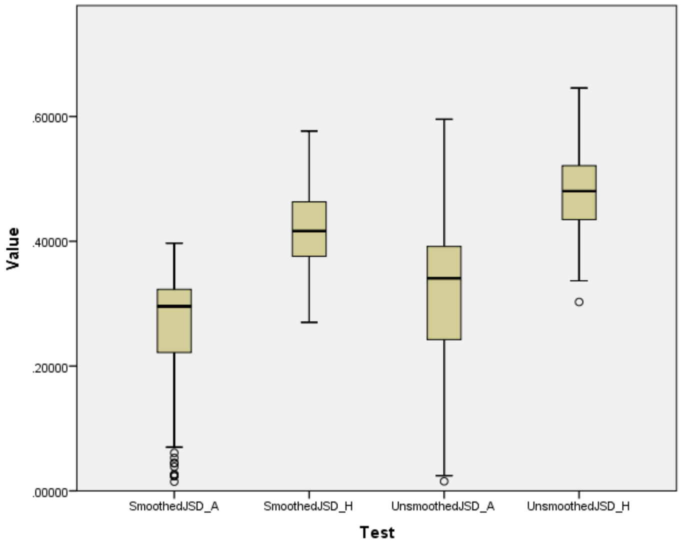
The mean and standard deviation of the tests B1 and B2 are represented in Table 3. This table shows the information according to the test group and the type of treatment (ASHuR and Human).

The Spearman correlation test compared results of ASHuR and the human considering the groups B1 and B2. Figure 6 shows results of 196 evaluations that represent each article, the x-axis represents tests blocks and the y-axis values. Although the data from block B1 are less dispersed than block B2, most of the B2 data are closer to 1, which means that block B2 has most acceptable results than block B1.

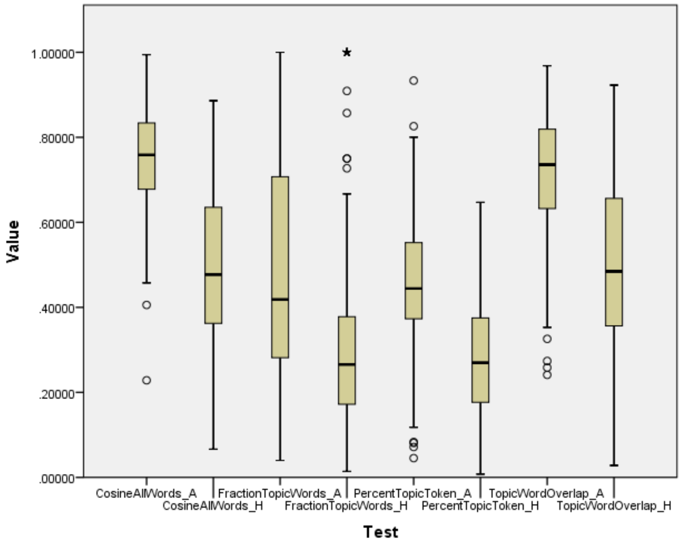
The descriptive data of evaluations are described in the Table 4. Outliers are commonly treated in some way to observe the impact of these on the outcome, for that reason data are analyzed with and without them. The block B1 had no outliers because of the low dispersion of data, however, the block B2 obtained some anomalous values.



a) Tests block B1 (part 1)



b) Tests block B1 (part 2)



c) Tests block B2

Figure 5. Data distribution of the evaluation summary-content organized by test blocks, block B1 – the best result is zero, block B2 – the best result is one

6.3 Discussion

Tests groups B1 and B2 showed a similar behavior in their results (see Figure 5). The ASHuR evaluation gave more desirable results in each test compared to the evaluation of human generated summaries. Results of ASHuR in the group B1 were closer to zero than the results of the human. The results of ASHuR in the block B2 are closer to one than human results.

The results dispersion of block B1, in Figure 5 a), shows more concise data for the evaluation of ASHuR. On the contrary, the human evaluation data are more compact in Figure 5 b), even though ASHuR data achieved a better score. Table 3 shows this information more precisely. Block B2 shows that human data are less spread than ASHuR data in most tests, however, human data do not receive a superior evaluation than ASHuR data.

The evaluation of results indicates that ASHuR generates better summaries than humans. However, these results are provided by automated tests that do not evaluate consistency and congruence of text sentences. Our best results are due to a system based on phrase extraction that is favored by this type of evaluation system. In spite of this, we made tests to put in context real results. If negative results had been obtained at this stage, it would have meant a poor phrase extraction that would have nothing to do with the important aspects of the text.

Test Group	**Test**	**Mean**	**SD**
B1	KLInputSummary_H	2.853	0.678
	KLInputSummary_A	1.939	0.486
	KLSummaryInput_H	3.302	0.878
	KLSummaryInput_A	1.065	0.434
	UnsmoothedJSD_H	0.479	0.057
	UnsmoothedJSD_A	0.313	0.117
	SmoothedJSD_H	0.420	0.060
	SmoothedJSD_A	0.265	0.087
B2	CosineAllWords_H	0.485	0.180
	CosineAllWords_A	0.751	0.124
	PercentTopicTokens_H	0.278	0.141
	PercentTopicTokens_A	0.447	0.161
	FractionTopicWords_H	0.317	0.218
	FractionTopicWords_A	0.504	0.280
	TopicWordOverlap_H	0.490	0.202
	TopicWordOverlap_A	0.712	0.148

Table 3. Mean and standard deviation of SIMetrix test

	B1		**B2**	
Descriptives	**All Values**	**Without Outliers**	**All Values**	**Without Outliers**
Correlations mean	0.812	0.812	0.865	0.901
Trimmed mean (5 %)	0.813	0.813	0.890	0.913
Median	0.822	0.822	0.943	0.951
Standard deviation	0.095	0.095	0.184	0.114
Minimum	0.536	0.536	−0.121	0.545
Maximum	0.999	0.999	0.999	0.999
P-value (mean)	0.188	0.198	0.135	0.099
P-value (trimmed mean)	0.187	0.187	0.110	0.087

Table 4. Descriptive data of the evaluation of correlation tests

The experiment applied the Pearson test to measure the degree of correlation between evaluations of ASHuR and the human. Results (see Table 4) show an average correlation between ASHuR-Human summaries of 0.812 for the block B1, this correlation means that the ASHuR summaries are 81.2 % similar to the human summaries according to the applied tests. However, the p-value obtained of 0.188 was not as good as we would wanted.

The block B2 showed an average correlation of 0.865 and an average trimmed to 95 % of 0.890, this indicates that there are 5 % of anomalous values that are negatively affecting results. When the average correlation is calculated without outliers then an average of 0.901 is obtained and an average trimmed to 95 % of 0.913.

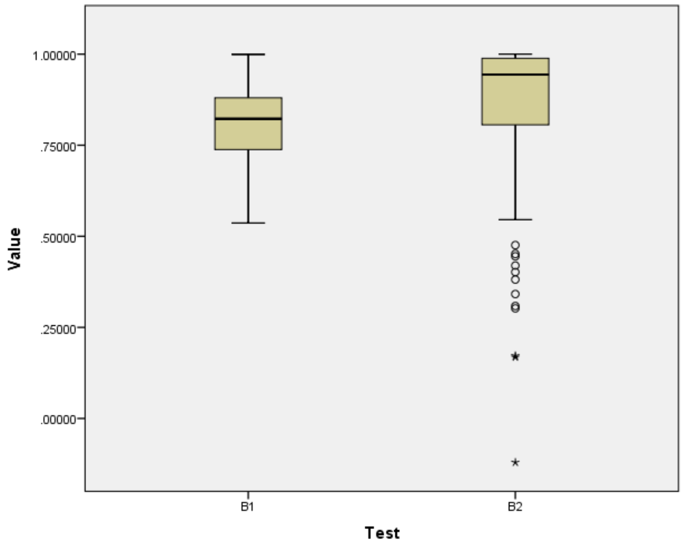


Figure 6. Results concentration of blocks B1 and B2

The block tests B2 shows a higher correlation than block test B1, even though the data of the block B2 are more dispersed than block B1. The block B2 is more prone to generating outliers. Outliers are caused by a significant difference between the results of ASHuR and the human result. These events occurred for the following reasons:

- Different use of words: Although the human summary is correct, it is poorly evaluated because different words are used in the writing of the summary and the content (4 cases).
- Different use to the summary section: The summary section has a different function than summarizing the document content, e.g., describes the use of the article instead of the content (3 cases).
- Inadequate sentence extraction: ASHuR performed an inappropriate phrases extraction due to established design characteristics of the algorithm (3 cases).
- Short summary: The summary is too short, limited to few words, this causes ASHuR only select a sentence that inappropriately represents the content (2 cases).

The proper treatment of these events will give more accurate results to ASHuR in future versions of our algorithm.

7 EVALUATION WITH ASHUR AND ROUGE

This section shows the evaluation of 21 articles of Wikipedia considering the ROUGE evaluation based on ASHuR. These articles are concepts related to Object Oriented Programming (OOP). The procedure consisted of three steps:

1. to obtain the summary using ASHuR,
2. to take the human summary from the Wikipedia article, and
3. to evaluate the human summary with ROUGE considering the ASHuR summary as a model.

ROUGE tests – ROUGE-1, ROUGE-2, ROUGE-L, and ROUGE-SU* – assessed articles summaries. Figure 7 shows the results of the evaluation of each Wikipedia article. Values of the graph represent the F-measure on the y-axis. The x-axis displays articles represented by an identifier. These identifiers are represented in Table 5.

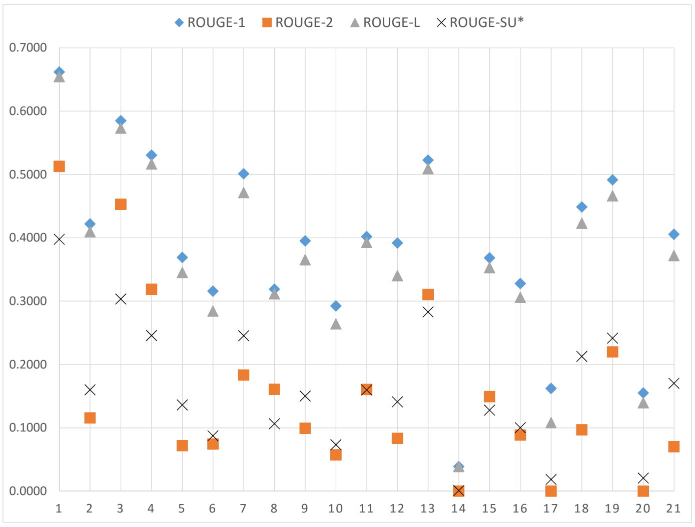


Figure 7. Results of the summaries evaluation with ROUGE and ASHuR

The summary score will depend on the test or tests considered. If a flexible evaluation is necessary, ROUGE-1 is chosen; if a harder evaluation is required, ROUGE-SU* could be used. This study employed an average of four tests, Table 5 shows results. According to the results, some articles present high probabilities to contain an inadequate summary. Articles *overriding*[5], *composition*[6], and *persistence*[7] have the lowest evaluations and they have a probability greater than 80 % to be inadequate or at least they have a lower level than the rest of the articles.

We analyzed these articles in detail to determine why their assessments are so low:

- The article *overriding* uses too many sample code in the content, most of the text is used to explain it. The summary is adequate, but this complement the content instead of functioning as a set of ideas that represent the content.
- The article *composition* has a very short summary based on two statements, this makes the summary evaluation problematic.
- The article *persistence*, although it has an accurate summary, it is relatively short, the content does not utilize words used in the summary.

Id	**Wikipedia Article**	**Mean**
1	Abstract type	0.557
2	Abstraction	0.277
3	Access modifier	0.479
4	Attribute	0.403
5	Class	0.231
6	Concurrency	0.190
7	Constructor	0.350
8	Encapsulation	0.224
9	Overloading	0.252
10	Hiding	0.172
11	Inheritance	0.279
12	Package	0.239
13	Method	0.406
14	Overriding	0.020
15	Modularity	0.250
16	Object	0.206
17	Composition	0.072
18	OOP	0.295
19	Parameters	0.355
20	Persistence	0.079
21	Scope	0.254

Table 5. Average of ROUGE tests for Wikipedia articles

ASHuR can review documents to identify cases where the summary is inadequate to the content by an alert signal.

[5] https://en.wikipedia.org/wiki/Method_overriding

[6] https://en.wikipedia.org/wiki/Object_composition

[7] https://en.wikipedia.org/wiki/Persistence_(computer_science)

8 CONCLUSIONS

This study presents ASHuR, an algorithm to measure the relation summary-content quantitatively without a model summary using ROUGE. According to the classification given in [16], our method works as follows: based on text, works with a single document, extracts information about text with an indicative proposal, considers only one language at the time (mono-lingual), gives an evaluation without ideal summary made by humans. This investigation worked with Wikipedia articles, but ASHuR can be applied to documents with a defined structure by content and summary.

ASHuR consists of two modules. The first module builds a model summary based on content, and the second module evaluates the original summary with the model summary created. ASHuR ranked in the first place among nine SS based on sentences extraction. In another experiment, our method achieved high correlation, based on the Pearson test, between ASHuR summary and human summary.

This study shows that a text can be evaluated without a model summary based on the proposed approach. We realize that the comparison based on human summaries is the best, however, when humans are not available, our proposal could be a good option.

According to evaluations performed in the experiment, the summary assessment implemented with our approach is an approximation with encouraging results. The project gives the possibility of evaluating summaries at the moment; one or multiple model summaries are not needed. Thus, ASHuR can evaluate a summary written by users in collaborative sites (e.g. Wikipedia) or can review texts written by students stored in online repository (e.g. Moodle).

For future work, we propose to solve problems such as synonyms, anaphora, proportion summary – content according to the length and term distribution. These would improve the algorithm and the precision of the sentences. This study considers adding to ASHUR the option of offering recommendations to improve its summary, considering the most common problems encountered in the evaluation.

REFERENCES

[1] AS: Autosummarizer. 2016, `http://autosummarizer.com/`.

[2] Bagalkotkar, A.—Kandelwal, A.—Pandey, S.—Kamath, S. S.: A Novel Technique for Efficient Text Document Summarization as a Service. 2013 Third International Conference on Advances in Computing and Communications (ICACC), 2013, pp. 50–53, doi: 10.1109/ICACC.2013.17.

[3] Dang, H. T.—Owczarzak, K. K.: Overview of the TAC 2008 Update Summarization Task. Text Analysis Conference (TAC 2008), 2008, pp. 1–16.

[4] Ferreira, R.—Freitas, F.—de Souza Cabral, L.—Lins, R. D.—Lima, R.—Franca, G.—Simske, S. J.—Favaro, L.: A Context Based Text Summarization

System. 11th IAPR International Workshop on Document Analysis Systems (DAS), 2014, pp. 66–70, doi: 10.1109/DAS.2014.19.

[5] Fox, C.: A Stop List for General Text. ACM SIGIR Forum, Vol. 24, 1989, No. 1-2, pp. 19–21, doi: 10.1145/378881.378888.

[6] FS: Free Summarizer. 2016, `http://freesummarizer.com/`.

[7] Hassel, M.—Dalianis, H.: SweSum – Automatic Text Summarizer. 2016.

[8] Hingu, D.—Shah, D.—Udmale, S. S.: Automatic Text Summarization of Wikipedia Articles. Proceedings of International Conference on Communication, Information and Computing Technology (ICCICT 2015), 2015, pp. 15–18, doi: 10.1109/ICCICT.2015.7045732.

[9] Hovy, E.—Lin, C.-Y.—Zhou, L.—Fukumoto, J.: Automated Summarization Evaluation with Basic Elements. Proceedings of the 5th International Conference on Language Resources and Evaluation, 2006, pp. 899–902.

[10] Kress, J. E.—Fry, E. B.: The Reading Teacher's Book of Lists. 6th edition, 2015.

[11] León, J. A.—Olmos, R.—Escudero, I.—Jorge-Botana, G.—Perry, D.: Exploring the Assessment of Summaries: Using Latent Semantic Analysis to Grade Summaries Written by Spanish Students. Procedia – Social and Behavioral Sciences, Vol. 83, 2013, pp. 151–155, doi: 10.1016/j.sbspro.2013.06.029.

[12] Lin, C.-Y.: ROUGE: A Package for Automatic Evaluation of Summaries. Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004), Vol. 1, 2004, pp. 25–26.

[13] Lin, C.-Y.: Looking for a Few Good Metrics: Automatic Summarization Evaluation – How Many Samples Are Enough. Proceedings of the NTCIR Workshop-4, 2004, pp. 1765–1776.

[14] Liu, F.—Liu, Y.: Exploring Correlation Between ROUGE and Human Evaluation on Meeting Summaries. IEEE Transactions on Audio, Speech, and Language Processing, Vol. 18, 2010, No. 1, pp. 187–196.

[15] Liu, K.—Terzi, E.—Grandison, T.: ManyAspects: A System for Highlighting Diverse Concepts in Documents. Proceedings of the 34th International Conference on Very Large Data Bases (PVLDB), Vol. 1, 2008, No. 2, pp. 1444–1447, doi: 10.14778/1454159.1454196.

[16] Lloret, E.—Palomar, M.: Text Summarisation in Progress: A Literature Review. Artificial Intelligence Review, Vol. 37, 2012, No. 1, pp. 1–41, doi: 10.1007/s10462-011-9216-z.

[17] Louis, A.—Nenkova, A.: Automatic Summary Evaluation Without Human Models. TAC, 2008.

[18] Louis, A.—Nenkova, A.: Automatically Assessing Machine Summary Content Without a Gold Standard. Computational Linguistics, Vol. 39, 2013, No. 2, pp. 267–300.

[19] Masoumi, S.—Feizi-Derakhshi, M.-R.—Tabatabaei, R.: TabSum – A New Persian Text Summarizer. Journal of Mathematics and Computer Science, Vol. 11, 2014, pp. 330–342.

[20] Molina, A.—Torres-Moreno, J.-M.: El Test de Turing para la Evaluación de Resumen Automático de Texto. Linguamática, Vol. 7, 2015, No. 2, pp. 45–55 (in Spanish).

[21] Nenkova, A.—McKeown, K.: Automatic Summarization. Foundations and Trends in Information Retrieval, Vol. 5, 2011, No. 2-3, pp. 103–233, doi: 10.1561/1500000015.

[22] OST: Online Summarize Tool. 2016, `https://www.tools4noobs.com/summarize/`.

[23] Passonneau, R. J.—Nenkova, A.—McKeown, K.—Sigelman, S.: Applying the Pyramid Method in DUC 2005. Proceedings of the Document Understanding Conference (DUC), Vancouver, BC, Canada, 2005, pp. 1–8.

[24] Porter, M. F.: An Algorithm for Suffix Stripping. Program, Vol. 14, 1980, No. 3, pp. 130–137, doi: 10.1108/eb046814.

[25] Saggion, H.—Torres-Moreno, J.-M.—da Cunha, I.—SanJuan, E.—Velázquez-Morales, P.: Multilingual Summarization Evaluation Without Human Models. Coling, 2010, Poster Volume, pp. 1059–1067.

[26] Sankarasubramaniam, Y.—Ramanathan, K.—Ghosh, S.: Text Summarization Using Wikipedia. Information Processing and Management, Vol. 50, 2014, No. 3, pp. 443–461, doi: 10.1016/j.ipm.2014.02.001.

[27] Steinberger, J.—Ježek, K.: Evaluation Measures for Text Summarization. Computing and Informatics, Vol. 28, 2009, No. 2, pp. 251–275.

[28] Torres-Moreno, J. M.—Saggion, H.—da Cunha, I.—SanJuan, E.—Velázquez-Morales, P.: Summary Evaluation with and Without References. Polibits Research Journal on Computer Science and Computer Engineering and Applications, Vol. 42, 2010, pp. 13–19, doi: 10.17562/PB-42-2.

[29] Torres-Moreno, J. M.—Saggion, H.—da Cunha, I.—Velázquez-Morales, P.—SanJuan, E.: Évaluation Automatique de Résumés Avec et Sans Référence. TALN 2010, Montréal, Canada, 2010, Vol. 1, pp. 19–23 (in French).

[30] Ubul, A.—Atlam, E.-S.—Kitagawa, H.—Fuketa, M.—Morita, K.—Aoe, J.-I.: An Efficient Method of Summarizing Documents Using Impression Measurements. Computing and Informatics, Vol. 32, 2013, No. 2, pp. 371–391.

[31] Yatsko, V. A.—Vishnyakov, T. N.: A Method for Evaluating Modern Systems of Automatic Text Summarization. Automatic Documentation and Mathematical Linguistics, Vol. 41, 2007, No. 3, pp. 93–103, doi: 10.3103/S0005105507030041.

Alan Ramírez-Noriega acquired his Master's degree in applied informatics at Universidad Autónoma de Sinaloa in 2014 and his Ph.D. degree in computer science from the Universidad Autónoma de Baja California in 2017. The main areas of interest are intelligent tutoring systems, knowledge representation, and text mining.

Reyes Juárez-Ramírez received his Master's degree in computer science from the Scientific Research and Higher Education Center in Ensenada in 2000, and his Ph.D. degree in computer science from the Universidad Autónoma de Baja California in 2008. He is currently Professor and Researcher at the Faculty of Chemical Sciences and Engineering, Autonomous University of Baja California. He has two main areas of interest: software engineering and human-computer interaction.

Samantha Jiménez is Ph.D. student at Universidad Autónoma de Baja California, Tijuana, México. She received her Bachelor's degree in 2011 in computer systems and her Master's degree in engineering in 2013 from the University of Colima, Colima, México. Her research interests are in the areas of human computer-interaction, dialogue systems, affective computing, multi-agent systems and evolutionary computing.

Sergio Inzunza received his Master's degree in computer science from the Autonomous University of Baja California in México in 2014. He is currently Ph.D. student in computer engineering, were he focused on creating tools for developers to model user and context information as a way to improve recommender systems.

Yobani Martínez-Ramírez received his Master's degree in computer science from Centro Investigación Científica y de Educación Superior de Ensenada (CICESE) and his Doctorate of Educational Technology from Centro Universitario Mar de Cortés. Since 1997 he has worked with the Universidad Autónoma de Sinaloa (UAS). He is currently Professor and Full-Time Researcher at the Faculty of Engineering of the UAS Mochis with PROMEP profile recognition (teacher improvement program) by the Ministry of Education (SEP). The areas of generation and application of knowledge of interest are implementing innovative systems and educational technology.