Volume 37 Number 3 September 2013 ISSN 0350-5596

Informatica

An International Journal of Computing and Informatics

Editorial Boards, Publishing Council

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor from the Editorial Board can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the list of referees. Each paper bears the name of the editor who appointed the referees. Each editor can propose new members for the Editorial Board or referees. Editors and referees inactive for a longer period can be automatically replaced. Changes in the Editorial Board are confirmed by the Executive Editors.

The coordination necessary is made through the Executive Editors who examine the reviews, sort the accepted articles and maintain appropriate international distribution. The Executive Board is appointed by the Society Informatika. Informatica is partially supported by the Slovenian Ministry of Higher Education, Science and Technology.

Each author is guaranteed to receive the reviews of his article. When accepted, publication in Informatica is guaranteed in less than one year after the Executive Editors receive the corrected version of the article.

Usability and Privacy Aspects of Moodle: Students' and Teachers' Perspective

Mirjana Ivanović, Zoran Putnik, Živana Komlenov
Faculty of Science, University of Novi Sad, Novi Sad, Serbia

Tatjana Welzer, Marko Hölbl, Tina Schweighofer
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia

Keywords: learning management system, privacy, security, experiences, usability, localization

Received: August 3, 2012

Choosing complex sets of tools, usually called learning management systems (LMSs), for creating perfect blends of traditional classroom activities and the most appropriate e-learning course components has become a common practice. Our institutions have opted for open source LMS Moodle. After years of its application in everyday teaching practice we were inspired to analyse the effectiveness of this platform. In this paper, results of the surveys compiled in order to reflect the student and teacher experiences with Moodle are presented. Main focus is providing insights into opinions, expectations and possible reluctance regarding usability and privacy when using its functionalities

Povzetek: V prispevku so predstavljene dolgoletne izkušnje in analize sistema Moodle.

1 Introduction

Contemporary standards in education require usage of different tools in order to supplement teaching and learning processes, as well as efficient assessment. A learning management system is often the foundation of a reliable e-learning platform and complies with standards and best practices recommended by respectable educational and corporate stakeholders (Georgouli, Skalkidis, & Guerreiro, 2008).

At our departments at the Faculty of Science, University of Novi Sad in Serbia and the Faculty of Electrical Engineering and Computer Science, University of Maribor in Slovenia such a solution is used for the design and delivering of courses that are supporting classroom training (Budimac, Putnik, Ivanović, Bothe, & Schützler, 2011).

Several years ago we decided to use and possibly extend an existing e-learning platform for our eCourses instead of developing a new one from scratch. After testing several systems we drew conclusions on the available tools. The system we chose had to be one of the established general purpose LMS solutions, preferably an open source one (Ahmed, 2005). Such a platform, apart from its flexibility and considerable cost savings, would offer possibilities for extensibility and customization according to one's specific needs.

The evaluation of open source LMSs was conducted according to a set of minimum criteria, which included active community, stable development status, good documentation, didactic objective and focus on the presentation of content and communication functionalities.

Our final choice was Moodle (Rice, 2008), for its fine basic features, great extensibility and even some potential adaptability features which were further developed in Novi Sad (Komlenov, Budimac, & Ivanović, 2008). A number of comparative studies and research papers (Al-Ajlan & Zedan, 2008; Di Domenico, Panizzi, Sterbini, & Temperini, 2005; Graf & List, 2005; Munoz & Van Duzer, 2005; Stewart et al., 2007) corroborated our choice.

Moreover, this solution has been accepted by the University of Maribor as the official LMS, and has also been introduced at a significant number of faculties in Novi Sad in the last couple of years, which certainly makes joint studies and reuse of teaching material among our universities more feasible (Bothe, Budimac, Cortazar, Ivanović, & Zedan, 2009).

Moodle is a modular and extensible platform which offers features to support different educational styles. It chiefly follows the established usability conventions (Melton, 2006): it has a simple interface, uses a minimal number of words, features roll-overs providing extra information, etc. Still, usability and privacy concerns must be addressed in detail when using such a solution.

In this paper the results of two surveys are presented in order to reflect the experiences of students and teachers with Moodle, regarding mainly those issues, and consequently the impact of using this LMS in everyday teaching practice on the academic achievements of students. The study was conducted as a part of a bilateral project between our institutions. Participation in the study was voluntary and anonymous for both students and teachers.

The results of our study should be of interest to university administrators, faculty members, and students who plan to offer, teach, or take courses implemented in Moodle. Also it can help many universities that are still deciding the extent of their offerings of online or blended

courses and the most appropriate platforms to use in structuring their offerings.

The rest of the paper is organized as follows. In Section 2 research endeavours somewhat similar to ours are observed, since their results and methodologies applied induced our investigation. However, it is focused on slightly different aspects of the platform in question. Section 3 discusses the survey outline and methods used in the conducted research. The discussion of the results from both students' and teachers' perspectives is presented in Section 4. Conclusions are drawn in Section 5 to foster future research and innovations in online teaching practice.

2 Related work

A significant number of published papers report on students' and/or teachers' perceptions of e-learning and the usability of the employed e-learning tools. This certainly includes Moodle (Kakasevski, Mihajlov, Arsenovski, & Chungurski, 2008; Kennedy, 2005; Kirner, Custódio, & Kirner, 2008; Liyanagunawardena, 2008; Melton, 2006), as one of the LMSs most frequently used at universities worldwide. Their focus groups were, however, usually students participating in one selected study program or even more often a single course, quite rarely complemented with their teachers.

However, some of the studies provided valuable conclusions and provoked further research. While some of them focused on technology-based components of such platforms, others studied the human factor of those systems considering student and instructor satisfaction, the importance of participant interaction in online environments, etc. There were even attempts to develop comprehensive assessment models, incorporating concepts from both information systems and education disciplines (Ozkan & Koseler, 2009).

It was, for instance, found that most information technology majors perceive learning to be more fun and of better quality within a technology-enhanced online learning environment (Parker, 2003). Furthermore, students who take online courses perceive a higher level of quality in their educational endeavours (Hannay & Newvine, 2006). However, lack of interaction, presence, or both may result in students' different observations on how well they may or may not have performed in an online class (Picciano, 2002; Song, Singleton, Hill, & Koh, 2004).

There seems to be a strong positive correlation between the degree of social presence and perceived learning as well as perceived quality of the instructor (Richardson & Swan, 2003). Not surprisingly, it was also revealed that participants of elective online courses tend to rate the modules positively while those in the obligatory courses often rate them more negatively (Smart & Cappel, 2006).

Students that experienced at least one well designed course enriched with resources, timely feedback and interactions with teachers generally report positive experiences (Weaver, Spratt, & Nair, 2008). The instructor's support in learning in fact strongly contributes to learning achievements and course satisfaction.

Besides the instructor's expertise and support, only a few other variables proved to be important for students' perceptions of learning achievements and course satisfaction (Paechter, Maier, & Macher, 2010): the structure and coherence of the teaching material and the course, the stimulation of learning motivation, and the facilitation of collaborative learning.

Teachers may also exhibit differing opinions about online learning and its effectiveness for the student (Bisoux, 2007). It is not rare for teachers to still perceive online learning as having numerous shortcomings, including (Totaro, Tanner, Noser, Fitzgerald, & Birch, 2005): the lack of instructor-student/student-student interaction; no structured classroom environment; students tending to teach themselves the course material; the difficulty of teaching quantitative courses online; the challenges of administering exams online, etc.

The open source learning management system Moodle is widely adopted at many universities and other organizations thanks to its tightly integrated set of tools designed from a social constructivist perspective. The advantages it offers over other (commercially) available LMSs were often analysed during the last couple of years.

The benefits of Moodle over rather popular proprietary LMSs like Blackboard (Kennedy, 2005) can be seen in Moodle's outstanding facilities developed to support communication in various ways, but also in providing better structure for all sorts of courses, i.e. more functional and likeable course organization. Additionally, Moodle's registration system and assignment submission module (Melton, 2006) and other standard modules (Kakasevski et al., 2008) were also assessed to some degree in terms of usability.

Nevertheless, surveys conducted in parallel at more than one university, with comparable groups of students of similar background, as well as their teachers (Tanner, Noser, & Totaro, 2009), are quite rare, especially those that address not only basic but also some important advanced features of the chosen platform.

Accenting privacy issues is also very important since the urge to protect security and privacy of data has lately become significant and extensively studied subject (Eibl, 2009; Klobučar, Jenabi, Kaibel, & Karapidis, 2007; Weippl & Tjoa, 2005).

Therefore we decided on conducting such a twofold survey at our institutions to provide ourselves and our colleagues from other universities, but also other interested parties, with possibly useful students' and teachers' insights in the current usability and privacy aspects of Moodle.

3 Survey outline and methods

The survey was twofold – one part aimed at students and the other at teachers. It was composed of a majority of closed questions, and some specific ones offering the possibility for answering more freely. Most open questions were bond to the closed ones in two ways:

- additional description after choosing an answer (e.g., "Do you distribute you teaching material periodically or at the beginning of the semester? Periodically. Period: ____");
- additional description after answering (e.g., "Do you use the blog functionality within Moodle? YES NO – If NO, please indicate why: ____").

We distributed the survey electronically, using Moodle's Feedback module. The response of both students and teachers was surprisingly fast – we needed only about 10 days to collect all answers in the survey for students and about 3 weeks to conduct the survey for teachers.

However, there were some differences between the aims of the survey used to collect teachers' experiences and opinions and the one prepared for students. The goal of the featured survey for students was to provide insights into their opinions, expectations and reservations regarding the usability of Moodle, the quality of teaching material available, usage of assessment means, communication and collaboration tools, as well as their privacy concerns.

The survey intended for teaching staff was compiled of differently formulated questions. Yet the goal was similar – to provide insights into their experiences, opinions, expectations and cautiousness regarding the effects of using Moodle in their teaching practice. Teachers were required to assess the usability of various Moodle's modules and comment on the ways they employ them in the courses they maintain and teach.

4 Results and discussion

Our institutions, the Faculty of Science, University of Novi Sad in Serbia and the Faculty of Electrical Engineering and Computer Science, University of Maribor in Slovenia, have been implementing e-learning using Moodle for several years now.

The Faculty of Electrical Engineering and Computer Science, University of Maribor employed Moodle for the first time as an obligatory teaching tool in the year 2007 when the execution of the teaching process according the Bologna declaration started. Nevertheless, Moodle had also been used to some extent before that.

The Faculty of Science, University of Novi Sad started using Moodle in 2004. Until now the majority of courses have been implemented in this LMS, especially those taught within Computer Science study programs.

After Moodle was used for several years, we decided to analyse its many specific aspects. A joint project between our institutions gave us the possibility of investigating possible differences in two different study/work environments.

4.1 Students' perspective

The survey was conducted at both institutions, with comparable numbers of students (136 in Slovenia, 130 in Serbia). However, the distribution of participants according to their year of studies was different (Table 1), because the existing undergraduate studies in Slovenia have recently been transformed into a three-year program, which also involved obligatory use of Moodle, so only students from the first two years were available at the time the survey was conducted.

Slovenian results were collected at study programs Computer Science and Informatics (65.44%), and Communication Technologies and Media Communications (34.56%), while Serbian survey participants were all students of Computer Science.

Table 1 depicts the distribution of survey participants according to their year of studies at both universities.

Year of study	Novi Sad	Maribor
1	43.85%	36.76%
2	16.92%	63.24%
3	17.69%	
4	10.00%	
5	8.46%	
PhD	3.08%	

Table 1: Survey participants according to their year of studies.

Gender-wise, there was more than 2/3 of male, and 1/3 of female students (Table 2).

Gender	Novi Sad	Maribor
male	67.69%	77.21%
female	32.31%	22.79%

Table 2: Survey participants according to their gender.

4.1.1 Overall quality of the existing teaching material

The majority of students assessed the quality of the teaching material currently available at our Moodle sites (mainly static content plus some electronic lessons for self-study purposes, enriched by assessment and communication facilities) as very good or good (Figure 1). Interestingly, there were 10 times more students that graded the available material as excellent in Novi Sad, and also none of the survey participants there assessed the available resources as very bad.

It might be possible that students from an EU university have greater expectations than the students in a developing country, but this result certainly shows that it is possible to develop and conduct courses of high quality even without any special funding or much institutional support.

What we were especially curious about were students' suggestions on how to improve the teaching material quality. They included the following: introducing additional exercises with different difficulty levels or examples of previous exams, more tests and assignments for students' self-evaluation, lessons with adaptive elements, video content, links to additional literature, etc.

All this suggests that students actually value course creators' efforts to involve more complex and interactive activities and resources in their courses, which in some cases might require usage of additional, either third-party or own, modules implemented for Moodle.

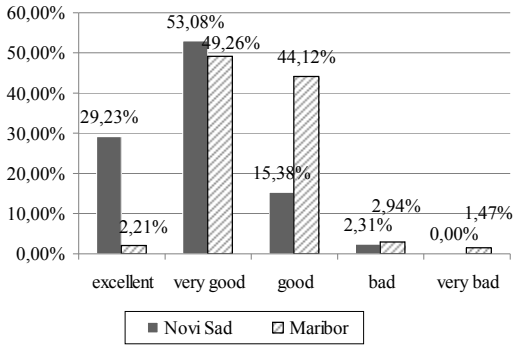


Figure 1: Overall quality of teaching material.

4.1.2 Graded tests

Quality of Moodle features, primarily Quiz module and similar, that enable students to take graded tests was also investigated in this research. 72.31% (Novi Sad) / 42.65% (Maribor) of survey participants had already been examined in such a way. They found it to be very convenient and they especially valued the increased speed of the grading process. The main problems that arouse while solving online tests identified by our students are the time limits and the possibility of hardware failure, which lowers their concentration.

Only 26.15% (Novi Sad) / 35.29% (Maribor) of students think that tests done using computers offer more possibilities to cheat compared to the usual settings when paper tests are used. Nevertheless, the teaching staff keeps constant efforts to reduce this ever existing assessment problem to the minimal possible level by administering tests in controlled environments like supervised computer rooms, limiting the access to certain IP addresses, etc. This practice is understood and supported by 81.54% (Novi Sad) / 44.12% (Maribor) of students. Thus, apart from the fact that online testing is much more employed in grading students in Novi Sad than in Maribor, it is interesting to notice that Serbian students are less resistant to all kinds of cheating restrictions.

4.1.3 Collaborative assignments

Regarding teamwork experiences, 27.94% of students in Maribor and 52.31% of students in Novi Sad (almost twice as much) had already done some collaborative assignments using Moodle's modules suitable for such efforts (Wiki, Workshop, etc.). They generally found these activities both challenging and valuable as learning experiences, and responded very well to the team-building practice promoted through them.

The fact that students are willing to work in small teams in order to solve various assignments, together with their satisfaction with what Moodle's modules intended to foster collaborative activities offer, is backing the already proven hypothesis that students who use opportunities in self-regulated and collaborative learning experience higher learning achievements (Paechter et al., 2010).

4.1.4 Usage of communication tools

Moodle has communication capabilities leaning towards Web 2.0 functionalities, like blog, forums, wiki, or chat. However, students are not very eager to use those features in their studies (Table 3).

Most of them say they still prefer personal communication with professors and teaching assistants or use email communication instead, which to some extent fits global trends noticed in other studies.

Tools	Novi Sad	Maribor
forums and instant messages	18.46%	39.71%
blogs	23.08%	11.03%
chat	26.15%	30.88%

Table 3: Frequent usage of communication tools.

4.1.5 Expressing opinions

Considering online surveys like this one, great majority of students, 91.54% (Novi Sad) / 80.15% (Maribor) of them, had no problem with filling the surveys out if they were to be completed anonymously. In general they value every opportunity to state their opinion on matters that directly influence the quality of the courses they attend. The rest of the surveyed students expressed their lack of belief in the possibility to be completely anonymous while filling out online surveys in environments like Moodle which systematically keep records of all user actions.

However, it is important to notice that 26.92% (Novi Sad) / 62.5% (Maribor) of the students are afraid of the consequences if they express a negative opinion or criticize a teacher within a Moodle course, for instance using a discussion forum. Yet only 6.92% (Novi Sad) / 16.47% (Maribor) of those students claim that their fear is based on some previous negative experience. Additionally, students would assess their teachers and courses they attend more freely if they would be assured anonymity.

4.1.6 Privacy concerns

Most of the survey participants (93.85% in Novi Sad and 91.18% in Maribor) are satisfied with privacy in Moodle. Others that are not so satisfied gave their reasons for that. As the most frequent cause they stated that other students who participate in the same course are able to track their online status and participation in various activities (for example submissions of assignment solutions). So the majority of students stated their wish for privacy,

signalling that at least Moodle's grade book should be more frequently and thoroughly used by their teachers.

On the other hand, some Serbian students in fact wished to be able to see all student grades thinking of that possibility as of a way to improve the transparency of grading. These dilemmas support previous evaluations of student perceptions of various e-learning components that showed that the students' strongest preference was to submit assignments and have the ability to check their grades online (Buzzetto-More, 2008).

A specific view of privacy in Moodle was investigated in detail, namely who should be able to access data from other users' profiles (Figure 2). Considering the question used to explore which pieces of information from user accounts should be hidden, expected answers such as email addresses, phone numbers, student ID numbers, etc. were received.

Some students also mentioned hiding first/last access times and activity logs of course participants. Most of these problems, now that we are aware of their existence and impacts on students' confidence, can be easily solved by changing certain system administration settings and introducing small modifications in course access privileges for users in the student role.

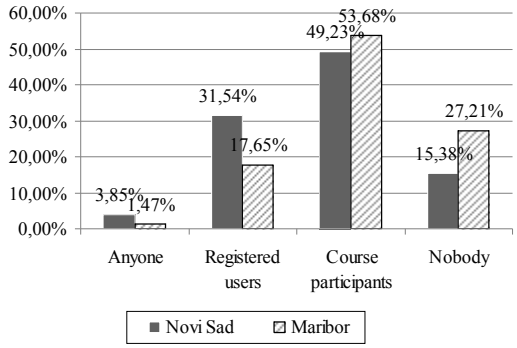


Figure 2: Accessing data from user accounts/system logs.

4.1.7 Technical problems and localization

A number of complaints appeared regarding the stability of the platform – 23.85% (Novi Sad) / 16.18% (Maribor) of students reported some technical issues. They had been usually in fact facing hardware and software limitations of the employed servers. The inconveniences were identified as: connection problems, slow response in case of many users connected to Moodle, difficulties when opening or downloading specific types of files in certain browsers, etc.

In Maribor, practically all of the survey participants believe that the Slovenian localization of Moodle is rather good. Similarly, Serbian language packs are well maintained according to 96.92% of the questioned students.

Students generally consider the localization to be rather important, which corresponds with the findings of other studies claiming that the use of native language in Moodle makes the accomplishment of students' tasks easier (Melton, 2006). Still, a lot of them habitually prefer using the interface in English.

4.2 Teachers' perspective

The other part of the survey was conducted with comparable numbers of teachers and teaching assistants, 25 in Maribor and 18 in Novi Sad, all working with students that participated in the first survey.

4.2.1 Design and implementation of learning resources provided online

Preparation of online learning resources is becoming one of the regular activities of our teaching staff, although it is not strictly required by the management at our faculties. Nevertheless, it requires extra effort and a certain amount of time (Table 4).

However, we expected the teachers to complain even more about the time management problems. Relatively mild feedback could, unfortunately, be credited to the fact that a lot of teachers simply are not motivated to, or do not have enough time, energy, or possibly even skills to produce more than small quantities of very simple online resources (totally opposite from what their students expect them to do).

Answer	Novi Sad	Maribor
more than for traditional resources	38.89%	28%
less than for traditional resources	33.33%	48%
the same as for traditional resources	27.78%	24%

Table 4: Time needed for the preparation of teaching material.

Although teachers from both institutions see the benefits of organizing e-learning efforts by using systems like Moodle, in Novi Sad 33.33% of them still prefer having their own home pages for (at least some of) their courses, saying that it is easier to maintain such pages, that it makes their work more flexible and independent, or that they are simply not significantly motivated to change their habits.

Interestingly, none of the survey participants in Maribor prefers such an option. In fact, 76% of them, as well as 44.44% of teachers in Novi Sad think that using Moodle is much better than maintaining separate course pages.

They point out that the LMS solves administrative issues, keeps all resources in one place making them easily accessible to students, and provides better structure of courses and more features to implement various online activities.

Finally, 22.22% (Novi Sad) / 24% (Maribor) of teachers prefer neither Moodle nor their own course

pages. They use the LMS in some of their courses, but still employ other mechanisms for specific course activities.

Separate tools often have a simpler and more likeable GUI or provide specific development instruments for certain course segments – lessons created and followed in a flexible way, more readable forums, better implemented chat and instant messaging options, freely structured surveys, complex wiki editing and tracking, special types of quizzes, etc.

Regarding the existing standard modules in Moodle, Lesson module seems to be one of the most precious ones equally in Novi Sad and Maribor (Figure 3).

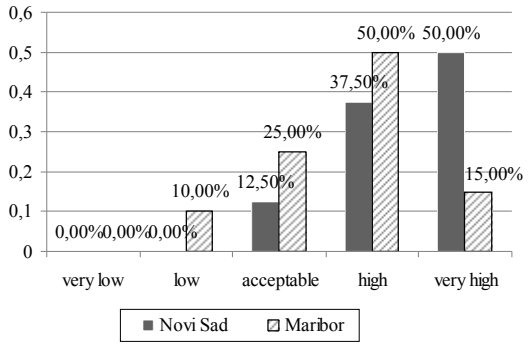


Figure 3: Usability of Moodle's Lesson module.

Teachers that participated in the survey, when using Moodle, apart from providing downloadable resources (lecture slides, assignments used for lab exercises, etc.) or links to external references, often present the teaching material shaped as more or less complex eLessons, built using Lesson module. This module was even extended in order to support creation of (semi-)adaptive eLessons (Komlenov et al., 2008).

Some of the questioned teachers also use modules like Glossary to explain key terms related to the topics they teach, or to provide their students with different kinds of tips or generally offer them easily accessible reference points. Glossary module received relatively good grades as well, especially in Novi Sad (Figure 4).

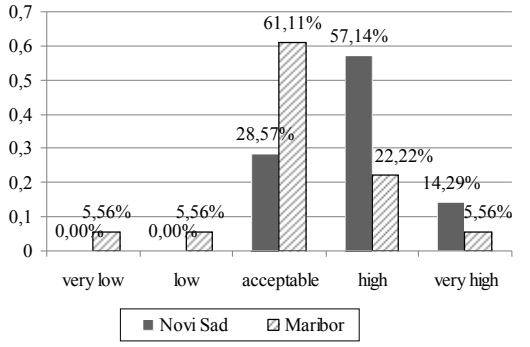


Figure 4: Usability of Moodle's Glossary module.

4.2.2 Graded tests

Only 16.67% (Novi Sad) / 32% (Maribor) of survey participants use online tests to officially assess their students. The rest of them do not use this possibility at all or they just provide tests for students' self-assessment that are always available and can be solved numerous times, but teachers that offer such tests do not analyse the results of their students.

Generally our colleagues still prefer paper tests due to possible organizational problems that can appear when online testing is practiced, issues concerning security and cheating, or they simply do not find online testing serious enough for grading the topics they teach. For some specific subjects there are also no suitable types of questions within the available tools.

Teachers that use online tests for official assessment have rather positive experiences with them. They especially value the implemented grading mechanisms that save them a lot of time so they can invest some more hours in preparation of bigger pools of questions that can be exploited in the following years as well. To prevent cheating they restrict solving tests to:

- certain amounts of time (all such teachers in both Novi Sad and Maribor),
- specific computer labs (all teachers in Novi Sad and 75% of teachers in Maribor),
- only particular IP addresses (all teachers in Novi Sad and 25% of teachers in Maribor).

Generally not many survey participants think that students have more opportunities to cheat when solving tests in Moodle than when doing paper tests (Table 5).

Interestingly, although about the same percentage of students (26.15%) and teachers (27.78%) believe so in Novi Sad, Slovenian teachers should take some more measures of precautions, since only 4% of them believe that it is easier for students to cheat when solving electronic tests instead of paper ones, while 35.29% of their students support that claim.

Assessment	Novi Sad	Maribor
more than when doing paper tests	27.78%	4%
less than when doing paper tests	27.78%	57%
the same as when doing paper tests	44.44%	39%

Table 5: Opportunities for cheating within tests solved in Moodle.

Quality of Moodle testing features, mainly Quiz module, was assessed as well (Figure 5). Some teachers that had not previously used this functionality chose not to grade it, so the assessment would not be influenced by their lack of experience with the options it offers.

4.2.3 Individual and collaborative assignments

The practice to distribute individual assignments to students using Moodle, and afterwards to collect their solutions, is rather common at both institutions.

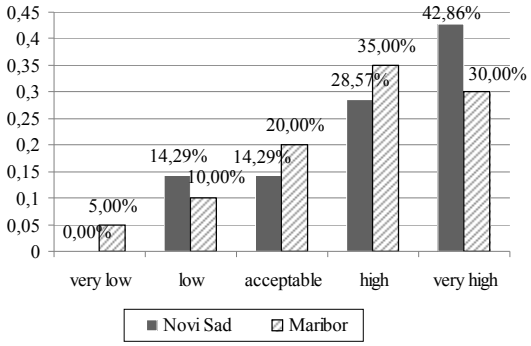


Figure 5: Usability of Moodle's Quiz module.

For this purpose teachers usually apply a variety of options provided in the Assignment module. And they are generally very satisfied with its quality (Figure 6).

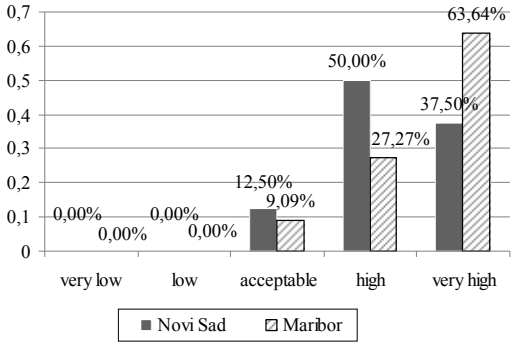


Figure 6: Usability of Moodle's Assignment module.

On the other hand, collaborative activities using appropriate Moodle's modules like Wiki have so far been introduced in only a couple of courses at both institutions.

Hence we received only 5 responses regarding the quality of functionalities of the Wiki module in Novi Sad. In Maribor, however, 20 survey participants assessed this module (Figure 7).

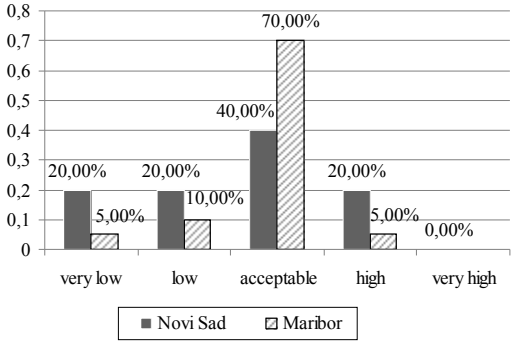


Figure 7: Usability of Moodle's Wiki module.

All in all, it received a lot of negative comments. Nobody addressed its usability as very high. Although this module satisfies the basic needs of students in their efforts to solve various team assignments, it is obvious that, despite its recent restructuring, teachers still think that it is not as functional as separate wiki systems.

4.2.4 Usage of communication tools

Moodle's communication tools are leading mechanisms of informing students about organizational and other issues within our courses according to 72.22% (Novi Sad) / 84% (Maribor) of teachers. Other common communications means are regular message boards (used by 11.11% of teachers in Novi Sad and 16% of teachers in Maribor), electronic message boards (used by 11.11% of teachers in Novi Sad and 68% of teachers in Maribor), personal/course pages, etc.

Teachers were therefore asked to assess the quality of Moodle's communication tools (Figures 8, 9 and 10), particularly having in mind their fitness to the teaching methods they practice and needs/habits of their students.

Discussion forums (Figure 8) seem to be well implemented in Moodle, while Chat module (Figure 9) received significantly lower grades.

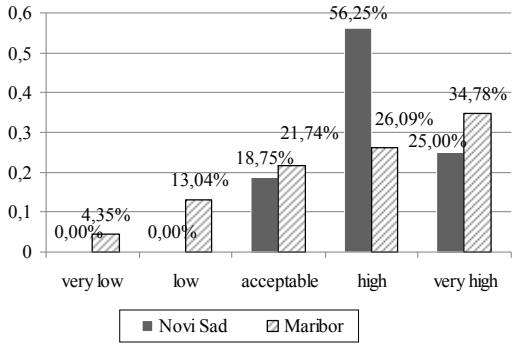


Figure 8: Usability of Moodle's Forum module.

Chat module is in fact implemented with very basic functionalities, so it certainly cannot be an adequate replacement for one of the separate chat products leading on the current software market. One would then expect students to use chat in Moodle much more rarely than discussion forums, but such a conclusion would be quite wrong. Students have obviously found proper uses for chat as well, even with limited functionality and Spartan design of this LMS component.

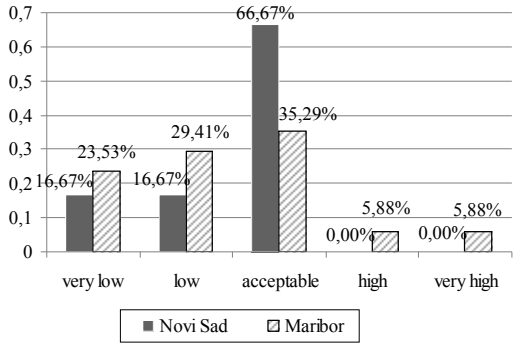


Figure 9: Usability of Moodle's Chat module.

Instant messaging system integrated in Moodle is more commonly applied by our teachers in their communication with students and colleagues, alike among students themselves, possibly because of its possibilities to serve as both synchronous and asynchronous means of communication, but also because of its more user-friendly implementation.

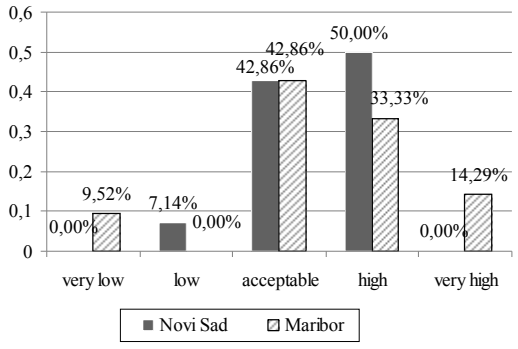


Figure 10: Usability of Moodle's instant messaging functionalities.

4.2.5 Expressing opinions

Majority of teachers, 72.22% (Novi Sad) / 71% (Maribor) of them, do not have a problem with answering this type of surveys. Actually they believe that conducting online surveys is a rather uncomplicated task if Moodle's Feedback module is used. Its usability was assessed as pretty high (Figure 11), thus it does not surprise that this once third-party module became one of the standard Moodle modules.

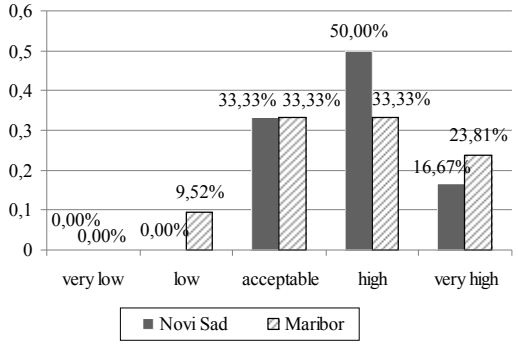


Figure 11: Usability of Moodle's Feedback module.

When it comes to dealing with opinions of students concerning course organization, quality of teaching material, grading issues and other course matters, our teachers find it too challenging.

In fact, only 16.67% (Novi Sad) / 20% (Maribor) of them already received critics within Moodle (usually in discussion forums). In such cases they elaborated their decisions online or in face-to-face meetings with students and/or improved the material in question.

4.2.6 Privacy concerns

All teachers that took part in the survey are generally satisfied with the level of privacy Moodle provides for their students. For example, they are content with the fact that students can only check the data regarding their own marks using the integrated grade book, with the possibility for groups of students to be defined both as separate and visible to each other, etc.

Teacher that took part in the survey have no privacy concerns regarding their own personal data, probably since they publish just some bits of information they really wish to share with their students. They are also protected to a certain extent by the role they have within the system.

4.2.7 Technical problems and localization

On the subject of technical problems, 33.33% (Novi Sad) / 37.5% (Maribor) of teachers said that they had encountered some difficulties while using Moodle. Primarily they were connected to responsiveness of the system while updating content, time required to clear cash/reload material, slow GUI rendering, lack of mass show/hide/move resources, etc.

Some of the issues are obviously the responsibility of the employed server, not Moodle itself, but there is also a certain amount of difficulties caused by Moodle's interface and specific implementation of some of its features that novice teachers have to get familiar with. Of course, some of the problems existed only in previous versions of the platform, not in the latest one.

Regarding localization, 33.33% of teachers in Novi Sad consider it properly done, while others have no opinion on the quality of Serbian language packs since they have never used them. In Maribor, however, 94% of survey participants are satisfied with the quality of Slovenian translation.

While only the teachers in Novi Sad still have the habit to use Moodle's interface in English, all of them, as well as 83% of teachers in Maribor, think that usage of course content in foreign languages is beneficial for their students. That practice promotes mobility of students and internationalization of studies in general, opens more possibilities for students to attain double/joint degrees, and is also valuable for their later professional life.

5 Conclusions

In this paper we presented the analysis of a survey conducted among Serbian and Slovenian students and teachers investigating usability and privacy aspects of

Moodle. Comparison of the results from each group showed that a number of differences in perception exist, possibly due to the heterogeneous points of view and motivations for online learning between teachers and students.

Still, while not always being able to formulate precisely their problems and dilemmas, both students and teaching staff are generally aware of the benefits of e-learning strategies and are very willing to present ideas for potential changes in the application of certain features of the system, as well as initiatives for upgrades of teaching material and techniques.

Students in both Maribor and Novi Sad are generally satisfied with frequently used Moodle's features and currently available teaching material. Teachers find most of the available Moodle modules to be rather functional, but they also commented on the poor functionalities of some of them.

From the teachers' perspective, the major obstacle to even greater application of various online activities in their practice presents a relatively low percentage of students who use instructive and communicative features of Moodle. Forums, chats, blogs, wikis, and other elements characterizing Web 2.0 are fairly unexploited. Online activities can be a good supplement to traditional methods of teaching and learning, but students have to be willing to participate in them and use the offered tools in a proper way.

Mechanisms that we currently employ using Moodle's modules make it easier for teachers to produce clear and easy readable, high quality teaching material and improve communication with their students. Problems that teachers are facing in the application of Moodle's features are mainly connected with the lack of time to learn how to use them and to prepare all the wished resources and activities.

A number of teachers that participated in the survey believe that their efforts would be much more successful if professional instructional designers were hired to help them in the preparation and maintenance of their courses.

Regarding possible privacy issues, the majority of students are satisfied with the privacy level offered by Moodle, though they gave specific remarks and expressed their general opinion that access to their private data should be limited. Teachers, on the other hand, seem to have no privacy concerns whatsoever.

We are aware of the fact that participants of our surveys were highly computer-skilled individuals due to their professional orientation, thus some of the assessments might have been somewhat different if they were made by students and teachers in different fields of study.

Other possible limitations of this investigation could be those that we did not take into consideration neither the possibility that some students might have experienced Moodle only within elective courses, which could have added to their general enthusiasm, nor the students final achievements and grades earned in courses supported by resources and activities developed in Moodle. A wider study with similar goals but varied groups of participants of diverse profiles could additionally prove the correctness of our conclusions.

References

[1] Ahmed, O. (2005). Migrating from Proprietary to Open Source Learning Content Management Systems, Department of Systems and Computer Engineering, Carleton University, Ottawa.

[2] Al-Ajlan, A., & Zedan, H. (2008). Why Moodle, In L. O'Conner (Ed.), *The 12th IEEE International Workshop on Future Trends of Distributed Computing Systems (FTDCS '08)* (pp. 58–64), Los Alamitos: IEEE Press.

[3] Bisoux, T. (2007). The Evolution of E-Learning, *BizEd*, January/February 2007, 22–29.

[4] Bothe, K., Budimac, Z., Cortazar, R., Ivanović, M., & Zedan, H. (2009). Development of a Modern Curriculum in Software Engineering at Master Level across Countries. *Computer Science and Information Systems*, 6 (1), 1–21.

[5] Budimac, Z., Putnik, Z., Ivanović, M., Bothe, K., & Schützler, K. (2011). On the Assessment and Self-Assessment in a Students Teamwork Based Course on Software Engineering. *Computer Applications in Engineering Education*, 19 (1), 1-9.

[6] Buzzetto-More, N. A. (2008). Student Perceptions of Various E-Learning Components. *Interdisciplinary Journal of E-Learning and Learning Objects*, 4, 113–135.

[7] Di Domenico, F., Panizzi, E., Sterbini, A., & Temperini, M. (2005). Analysis of commercial and experimental e-learning systems. *Quality, Interoperability and Standards in e-learning Team*, TISIP Research Foundation, Trondheim.

[8] Eibl, C. J. (2009). Privacy and Confidentiality in E-Learning Systems. In M. Perry et al. (Eds.): *The 4th International Conference on Internet and Web Applications and Services (ICIW 2009)* (pp. 638–642), Los Alamitos: IEEE Press.

[9] Georgouli, K., Skalkidis, I., & Guerreiro, P. (2008). A Framework for Adopting LMS to Introduce e-Learning in a Traditional Course. *Educational Technology & Society* 11 (20), 227–240.

[10] Graf, S., & List, B. (2005). An Evaluation of Open Source E-Learning Platforms Stressing Adaptation Issues. In P. Goodyear et al. (Eds.), *The 5th IEEE International Conference on Advanced Learning Technologies (ICALT'05),* (pp. 163–165), Los Alamitos: IEEE Press.

[11] Hannay, M., & Newvine, T. (2006). Perceptions of Distance-Learning: A Comparison of Online and Traditional Learning. *MERLOT Journal of Online Learning and Teaching*, 2 (1), 1–11.

[12] Kakasevski, G., Mihajlov, M., Arsenovski, S., & Chungurski, S. (2008). Evaluating Usability in Learning Management System Moodle. In V. Luzar - Stiffler, V. Hljuz Dobric, Z. Bekic (Eds.), *The 30th International Conference on Information*

Technology Interfaces (ITI 2008), (pp. 613–618), Los Alamitos: IEEE Press.

[13] Kennedy, D.M. (2005). Challenges in Evaluating Hong Kong Students' Perceptions of Moodle. In Goss, Halima (Eds.), *The Australasian Society for Computers in Learning in Tertiary Education Conference (ASCILITE 2005)*, (pp. 327–336).

[14] Kirner, T.G., Custódio, C. de A., & Kirner, C. (2008). Usability Evaluation Of The Moodle System From The Teachers' Perspective. In M.B. Nunes, M. McPherson (Eds.), *The International Conference e-Learning (IADIS 2008)*, (pp. 371–378).

[15] Klobučar, T., Jenabi, M., Kaibel, A., & Karapidis, A. (2007). Security and Privacy Issues in Technology- Enhanced Learning. In Cunningham P., Cunningham M. (Eds.): *Expanding the Knowledge Economy: Issues, Applications, Case Studies* (pp. 1233–1240). Amsterdam: IOS Press.

[16] Komlenov, Z., & Ivanović, M. (2008). Introducing adaptivity to e-lessons to enhance student learning. In R. Williams, D. Remenyi (Eds.), *The 7th European Conference on e-Learning* (pp. 571–580).

[17] Liyanagunawardena, T.R. (2008). Measuring Student Perception and Actual Usage of Online Learning Management System. *Communications of the IBIMA* 4 (21), 165–168.

[18] Melton, J. (2006). The LMS Moodle: A Usability Evaluation. *Languages Issues* 11/12(1), 1–24.

[19] Munoz, K., & Van Duzer, J. (2005). *Blackboard vs. Moodle: A Comparison of Satisfaction with Online Teaching and Learning Tools*, Humboldt State University.

[20] Ozkan, S., Koseler, R. (2009) Multi-dimensional students' evaluation of e-learning systems in the higher education context: An empirical investigation. *Computers & Education* 53 (4), 1285–1296.

[21] Paechter, M., Maier, B., Macher, D. (2010) Students' expectations of, and experiences in e-learning: Their relation to learning achievements and course satisfaction. *Computers & Education*, 54 (1), 222–229.

[22] Parker, M. (2003). Technology-enhanced e-Learning: Perceptions of First Year Information Systems Students at the Cape Technikon. In J. Eloff et al. (Eds.), *The South African Institute of Computer Scientists and Information Technologists (SAICSIT 2003)* (pp. 316–319).

[23] Picciano, A. (2002). Beyond Student Perceptions: Issues of Interaction, Presence, and Performance in an Online Course. *Journal of Asynchronous Learning Networks*. 6 (1), 21–40.

[24] Rice, H.W. (2008). *Moodle 1.9 E-Learning Course Development – A complete guide to successful learning using Moodle*, Birmingham: Packt Publishing.

[25] Richardson, J.C., & Swan, K. (2003). Examining Social Presence in Online Courses in Relation to Students' Perceived Learning and Satisfaction. *Journal of Asynchronous Learning Networks* 7(1), 68–88.

[26] Smart, K.L., & Cappel, J.J. (2006). Students' Perceptions of Online Learning: A Comparative Study. *Journal of Information Technology Education* 5, 201–219.

[27] Song, L., Singleton, E., Hill, J., & Koh, M. (2004). Improving Online Learning: Student Perceptions and Challenging Characteristics. *Internet and Higher Education* 7, 59–70.

[28] Stewart, B., Briton, D., Gismondi, M., Heller, B., Kennepohl, D., McGreal, R., & Nelson, C. (2007). Choosing MooDLe: An evaluation of Learning Management Systems at Athabasca University. *International Journal of Distance Education Technologies* 5 (3), 1–7.

[29] Tanner, J.R., Noser, T.C., Totaro, M.W. (2009). Business Faculty and Undergraduate Students' Perceptions of Online Learning: A Comparative Study. *Journal of Information Systems Education* 20 (1), 29–40.

[30] Totaro, M.W., Tanner, J.R., Noser, T.C., Fitzgerald, J.F., Birch, R (2005). Faculty Perceptions of Distance Education Courses: A Survey. *Journal of College Teaching & Learning* 2 (7), 13–20.

[31] Weaver, D., Spratt, C., & Nair, C.S. (2008). Academic and student use of a learning management system: Implications for quality. *Australian Journal of Educational Technology* 24 (21), 30–41.

[32] Weippl, E.R., & Tjoa, A.M. (2005). Privacy in e-learning: anonymity, pseudonyms and authenticated usage. *Interactive Technology and Smart Education* 2 (4), 247–256.

Web Phishing Detection Based on Page Spatial Layout Similarity

Weifeng Zhang
School of Computer, Nanjing University of Posts and Telecommunications, China
E-mail: zhangwf@njupt.edu.cn

Hua Lu
Department of Computer Science, Aalborg University, Denmark
E-mail: luhua@cs.aau.dk

Baowen Xu
Department of Computer, Nanjing University, China
E-mail: bwxu@nju.edu.cn

Hongji Yang
Software Technology Research Laboratory, De Montfort University, England
E-mail: hyang@dmu.ac.uk

Keywords: web phishing, page spatial layout similarity

Received: July 8, 2012

Web phishing is becoming an increasingly severe security threat in the web domain. Effective and efficient phishing detection is very important for protecting web users from loss of sensitive private information and even personal properties. One of the keys of phishing detection is to efficiently search the legitimate web page library and to find those page that are the most similar to a suspicious phishing page. Most existing phishing detection methods are focused on text and/or image features and have paid very limited attention to spatial layout characteristics of web pages. In this paper, we propose a novel phishing detection method that makes use of the informative spatial layout characteristics of web pages. In particular, we develop two different options to extract the spatial layout features as rectangle blocks from a given web page. Given two web pages, with their respective spatial layout features, we propose a page similarity definition that takes into account their spatial layout characteristics. Furthermore, we build an R-tree to index all the spatial layout features of a legitimate page library. As a result, phishing detection based on the spatial layout feature similarity is facilitated by relevant spatial queries via the R-tree. A series of simulation experiments are conducted to evaluate our proposals. The results demonstrate that the proposed novel phishing detection method is effective and efficient.

Povzetek: Opisana je detekcija spletnega ribarjenja na osnovi podobnosti strani.

1 Introduction

Along with the wide use of the Internet, a rapidly growing number of people are using various online services such as e-banking, online shopping, etc. These services give users great convenience. Meanwhile, the number of phishing web sites also increases very quickly. According to the statistics of PhishTank, over 1.3 million phishing sites are verified and registered in its database merely in the first two years after its launch. There are about 5919 online phishing sites, an d the number of offline phishing sites are close to 1.3 millon. And everyday a large number of new phishing sites were found, in average 600 sites are submitted and verified daily. Phishing sites cheat users by simulating the interfaces of genuine web sites, and defraud users of their sensitive information like user ID, password, and credit card number. Illegal uses of such stolen information can cause significant loss to users. Plenty of phishing web sites are found every day and phishing fraud is an increasing crime on the Internet. Therefore, it is desirable that phishing sites are detected effectively and users can get alerts to avoid being trapped.

Phishing sites are often sent out in random spam emails. Such emails often target those users who have no experience on network security, and make them believe that the emails come from legitimate organizations. Typically, these emails fake some reasons to require users to update their account information. When a user tries to log in through the interface provided in the email, sensitive information like user name and password will be stolen by the phishing site. Although many users nowadays have more or less experience on security and many email gateways can filter out most spams, a considerable number of users still become victims of phishing sites. There exist various anti-phishing approaches. A detailed review can be found in Section 2. Blacklist based approaches can precisely filter

out any phishing web page included in the blacklist. However, it is very hard to maintain up-to-date blacklists because phishing pages often have short existence and new phishing pages come out at a good pace [1, 2]. Web feature based approaches analyze the feature differences between genuine web page and phishing web page, extract critical features, and construct classifiers to classify subsequent web pages in phishing detection. Such approaches work fast and are able to detect phishing pages that have not been identified before. The difficulty of such approaches lies in determining classification features as phishing pages are always created to be alike to their corresponding genuine pages, which lowers detection accuracy. The third-party tools based phishing detection approach use the third-party tools like search engines to detect phishing pages [3, 4]. We examined this method by experiments, and found lots of phishing pages can be found in the search results, which may be due to SEO(search engine optimization) methods used by creators of phishing sites. If the phishing pages can be searched in search engines, this approach fails basically.

Recently, whitelist approach is often used in phishing page detection. It constructs a feature library from those web pages that are likely to be imitated by phishing pages, and then identifies phishing pages by computing the similarities between a web page and the feature library. As the whitelist approach is based on similarity search rather than exact matching, its detection speed is greatly affected by the feature library size as well as the search strategies. Some filtering methods, e.g., measuring file size, counting image number, and hierarchical clustering, have been used to reduce the number of signature pages to be compared. These methods can also rule out relevant web pages and thus results in errors in phishing detection. Furthermore, hierarchical clustering based filtering usually produces poor clustering results because of the differences among individual web pages, which inevitably impairs the filtering accuracy of feature library.

In order to solve the above problems, we in this paper propose a novel phishing detection approach that exploits the relevant spatial layouts of web pages. Song etal use a vision-based page segmentation algorithm to partition a web page into semantic blocks with a hierarchical structure. Then spatial features (such as position and size) and content features (such as the number of images and links) are extracted to construct a feature vector for each block [5]. A phishing page and its corresponding genuine page are close to each other visually. Consequently, page elements (e.g., text fields, images, buttons, etc.) in the phishing page are probably placed at the same or similar positions and of the same or similar size to their counterparts in the genuine one. Such spatial layout similarities can be used to determine whether a suspect page is close enough to a genuine page to be a phishing page.

Our approach makes use of spatial index on web page spatial layouts to quickly filter out unqualified candidates from the feature library. Also, it takes into account the important spatial layout features in defining and computing the similarity between web pages. On one hand, we improve the filtering performance (filtering speed and filtering accuracy) on feature library by including spatial layout features in the library.On the other hand, we combine spatial layout features with existing popular features to improve the accuracy of phishing detection. By an R-tree indexing pages in the feature library based on spatial features, we are able to fast obtain candidate pages in the library that are visually close to a suspect page. The R-tree can also filter out a considerable number of pages in the library without computing the concrete similarities.

A crucial part in our approach lies in capturing the layout characteristics of a web page to protect, i.e., a page likely to be imitated by a phishing page. After displaying a web page by the rendering engine (also known as layout engine) in a web browser, we develop two segmentation methods to extract its layout features. One method works by analyzing the page's DOM tree. For each tree node, it produces the placement, width, and height of the corresponding page block. Nested blocks are allowed in this method. Based on image segmentation, the other method employs image edge detection techniques to divide the entire image of a web page into non- overlapping blocks.

All those blocks obtained in either way are represented in rectangles and constitute the feature library. We then build an R-tree to index the entire feature library to facilitate search in phishing detection. Subsequently, given a suspect phishing page, its layout features are extracted likewise and used to compare against those pages whose layouts have been captured in the feature library. Through the R-tree, most pages in the library are filtered out based on spatial layout dissimilarity. Further, the concrete similarity is computed between the suspect page and each of the few candidate pages passing the filter. The suspect page will be regarded as phishing page if the similarity is greater than a pre-specified threshold.

We make the following contributions in this paper:

- First, we define the spatial layout features for web pages, and develop two feature extraction methods that make use of relevant functionality of modern web browsers.
- Second, we present an effective web page similarity definition that takes into account the proposed page spatial layout features.
- Third, we design an R-tree index for legitimate web page library that is organized based on spatial layout features, and propose library search algorithms for phishing detection.
- Fourth, we conduct an extensive experimental study to evaluate our proposals. The results suggest the proposed approach is effective and efficient.

The rest of this paper is organized as follows. Section 2 briefly reviews the related work on phishing page detection.

Section 3 presents the layout features of web pages and our extraction methods. Section 4 details how to build R-tree to index the web page layout features in the feature library. Section 5 reports on an extensive experimental study. Finally, Section 6 concludes the paper and discusses several directions for future work.

2 Related work

The email based approach filters out phishing links in an email by anti-spam technologies. Fette et al. proposed machine learning based PILFER [6] that extracts 9 features from the links in an email and trains a random tree as a classifier for subsequent pages. Bergholz et al. [7] generates extra email features through a trained Markov chain and a new model of potential class topics. These extra email features, together with basic ones, are used to train a classifier that is able to reduce over two thirds false positives than PILFER. Moreover, Bergholz et al. [8] proposed a detection method that simulates hidden interference information by OCR technology. It however incurs long running time and shows poor classification performance because the used OCR technology is not good at recognizing texts in disturbed images.

The blacklist based method constructs a blacklist of collected phishing site URLs. When a user visits a URL, the URL will be checked up in the blacklist. If the URL is in the blacklist, the access to it will be blocked. Popular web browsers such as Microsoft Internet Explorer and Google Chrome have built-in anti-phishing blacklists [9]. Some other software, e.g., NetCraft [2], SiteAdvisor [10], can filter out phishing sites in the their blacklists by browser toolbars. However, it is a big challenge to maintain the blacklist up-to-date in the presence of the dynamics of phishing sites.

The third-party tools based anti-phishing mainly depends on the ranking of search engines. Such methods make use of the fact that new web sites and sites with few clicks are usually ranked lowest by search engines. Given a web page, the TF-IDF scores of each term in it are calculated, and the top 5 terms are selected to generate a lexical signature of the page. Subsequently, the lexical signature is sent to a search engine (e.g., Google) to evoke a search. The web page is regarded as a phishing page if its domain name is out of the top N in the search result [1, 9]. Orthogonally, Moore et al. [11] proposed to use search engines to find potentially vulnerable hosts. In particular, analyzing search logs discloses the search terms commonly used by attackers, which indicate vulnerable web sites.

There also exists similarity based anti-phishing. Such an approach decides a given web page to be a phishing page if its similarity to a genuine page is greater than a prespecified threshold. So far two kinds of similarities have been used: structural and visual. Liu et al. [12] proposed to detect phishing web pages by computing the structural similarity between corresponding DOM trees. Angelo et al. [13] proposed to compute the similarity by comparing the HTML tags of two web pages. In contrast, visual similarities are computed based on the image features of two web pages [1, 14, 15]. It is noteworthy that our approach in this paper not only extracts new visual features from web pages, i.e., spatial layout, but also combines visual similarity with structural similarity when comparing web pages. To the best of our knowledge, this is the first work that is characterized by such comprehensiveness in phishing detection.

3 Spatial layout features of web page

After downloading a requested web page, a web browser analyzes the html document, extracts the embedded web links, executes the scripts in the html document, and then decides whether to issue other URL requests. When the resources (e.g., texts, pictures, etc.) contained in the page are returned, the web browser will render these resources according to their properties. The user then can see in the browser the web page with both texts and visual resources. As phishing pages are always intended to make people believe they are the genuine pages, they look very similar or even identical to their target genuine pages. As a result, the rendering features of a phishing page and the counterparts in the target genuine page are very similar visually. Accordingly, the basic phishing detection process consists of the following three steps (see Figure 1).

Step 1: Access the web page (denoted as url) and its embedded resources. *Step 2*: Extract the features of the rendered web page. All the extracted features form a signature that is denoted as S (url). *Step 3*: Compute the similarity between S(url) and those signatures in the feature library. If there exists one signature S_i in the library such that similarity between S (url) and S_i is greater than a pre-specified threshold, the web page url is judged as a phishing page, and an alert is issued.

From the above steps it can be seen that two points are very important for good results of phishing detection. It is vital to extract critical and suitable features from rendered web pages. It is also very helpful to improve the accuracy of feature similarity computation. Conventional features of rendered web pages include text features, image features, overall image features, etc. They are covered in detail elsewhere [16]. In this paper, we focus on the spatial layout features of rendered web pages, which have not been considered in the literature, and utilize them in effective phishing detection.

3.1 Extraction of spatial layout features

After a web page is completely downloaded by a browser, it is resolved into a HTML DOM tree and all its embedded elements are rendered in the browser through a rendering engine. When a page is being rendered, it is easy to get the rectangle region that each page element occupies in the browser. Such rectangle regions can be obtained in

Figure 1: Feature library based phishing detection

an alternative way as follows. A rendered web page can be divided into image segments by some edge detection algorithm which can return the rectangle region each image segments occupies in the browser.

Definition 1. *(**Spatial Feature**) A spatial feature of a web page element is a rectangle denoted as* $rect = \langle left, top, width, height \rangle$*, where* $(top, left)$ *represents the top-left corner coordinate of the rectangle in the browser,* $width$ *represents the rectangle's width, and* $height$ *represents the rectangle's height.*

The spatial features of all elements in a web page form the spatial layout feature of that web page. As mentioned above, the spatial layout feature can be obtained by either combining a browser rendering engine with the page DOM tree or applying image detection algorithms after page rendering. Next, we present these two options in Sections 3.1.2 and 3.1.2 respectively.

3.1.1 DOM tree based spatial layout feature extraction

DOM tree based extraction of spatial layout features needs to call the browser layout engine and analyze a DOM tree using some tools. The browser layout engine integrates texts, images, css files, java scripts, and other related resources altogether to render a web page, by referring to the page's DOM tree. Assume the pixel in the upper-left corner of the web browser display region is origin (0, 0), the X-axis is from left to right, and Y-axis is from top to down, both measured in units of pixel. The browser engine positions each DOM tree node according to its four numerical attributes: X coordinate, Y coordinate, width and height. These four attributes form the layout feature of a page element corresponding to a DOM tree node.

The layout feature of a web page element is obtained as follows. We first call the parsing engine of a chosen web browser to parse the web page and get a corresponding DOM tree. After that, we traverse the DOM tree, and get the corresponding display region of each node. If a node's display region area is larger than a pre-specified threshold (we set it 50 in this paper), the layout of this node is generated. Here we ignore DOM tree nodes with small display regions because small regions are insignificant visually and thus unimportant in phishing detection.

Definition 1 defines the general spatial features of web page elements. Applying it to the DOM tree based feature extraction, the spatial layout feature of a web page is a set of relevant rectangles, as defined in the following.

Definition 2. *(**DOM Tree based Spatial Feature**) Given a web page* p*, and its DOM tree* $DT(p)$*, its DOM tree based spatial feature* $SFD(p)$ *is a set of sufficiently large rectangles, each corresponding to a node in the DOM tree. Formally,* $SFD(p) = \{rect_i \mid \text{Area}(rect_i) > T_{area}, \exists node_i \in DP(p) \text{ s.t. } node_i\text{'s extent is } rect_i\}$.

In the definition T_{area} denotes a pre-specified threshold that helps filter out small rectangles.

Figure 2: DOM tree based spatial layout features

Figure 2 shows an example of DOM tree based spatial layout features. Each block represents the position and size of a rendered page element, which corresponds to a DOM tree node. Note that we filter out small rectangles with area smaller than 50 measured in pixels. It can also be seen from the figure that different topology relationships exist among these rectangles. A rectangle recti contains another one rectj if the former's corresponding DOM tree node ni is an ancestor of the latter's node nj. The adjacent relationship between two rectangles also reflects their corresponding nodes are adjacent in the DOM tree. In contrast, the overlap relationship between rectangles results from the relayout of corresponding DOM tree nodes, which is done by the rendering engine according to relevant CSS rules and/or java script codes.

DOM tree based layout feature extraction needs to call web browser APIs to get the layout information. However, web browsers (e.g., the Microsoft Internet Explorer) only allow us to get the width/height values of every block and the Top/Left values relative to its parent block. In order to get the X and Y coordinates of each block, we need to recursively add the Top values and the Left values of the related blocks. Figure 3 gives an example of calculation of the block coordinates.

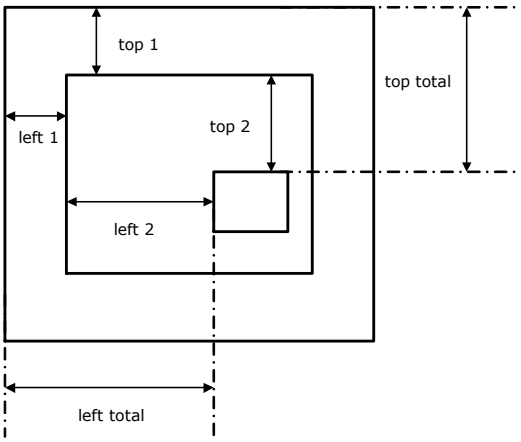


Figure 3: Calculation of block coordinates

In this example, we need to calculate the X and Y coordinates of the innermost block. As the Microsoft Internet Explorer does not provide methods to directly get these two parameters, we calculate them through the parameters of other blocks. First, we get the Top value and the Left value of block A relative to its parent block. Second, we obtain the Top value and Left value of the parent block relative to the upper block, and so on. Finally, if we find that the upper block has the label 'Body', the iteration stops. As a result, the sum of all the Top values is the coordinate Y, and the sum of all the Left values is the coordinate X. In Figure 3, we compute the left total and top total as follows:

left total = left 1 + left 2, and top total = top 1 + top 2.

Here, the left total is the value of the block relative to the left value of the browser window, i.e., the X coordinate of the block's top-left corner. Likewise, top total is the Y coordinate of its top-left corner.

3.1.2 Image segmentation based spatial layout feature extraction

The image segmentation based extraction produces image blocks in a rectangle. Each rectangle is also represented by four numerical attributes: the X coordinate, the Y coordinate, the width, and the height. Such rectangles are extracted as follows. We first parse the web page, render it through the browser's rendering engine, and get the entire page in an image. After that, we divide that image into a collection of smaller image blocks based on the gaps between them, using some image edge detection algorithm. Finally, we calculate the layout feature, i.e., the top-left corner coordinates and size, of each image block thus obtained.

Definition 3. ***(Image Segmentation based Spatial Feature)*** *Given a web page p, its image segmentation based spatial feature $SFI(p)$ is a set of sufficiently large rectangles, each corresponding to an image block in the rendered page. Formally, $SFI(p) = \{rect_i \mid \text{Area}(recti) > T_{area}, rect_i \in Blocks_{IS}(p)$[1]$\}$.*

Rectangles in Figure 4 are obtained from a rendered web page through image segmentation, where each rectangle reflects the position and size of a displayed element. Note less blocks are generated by image segmentation than by DOM tree based extraction. Therefore, we set the area threshold T_{area} to a smaller value 20 in this example. As a result, those blocks whose areas are smaller than 20 are excluded. Compared to the DOM tree based layout features in Figure 2, the image segmentation based blocks in Figure 4 are more visible to human visual discrimination.

Figure 4: Image segmentation based spatial layout features

After the definitions and extractions of web page spatial layout features, we are ready to introduce the concept of corresponding blocks between two web pages.

[1]We use $Blocks_{IS}(p)$ to denote all the blocks that are obtained by an image segmentation method applied to page p.

3.2 Corresponding blocks matching

The spatial similarity between web page blocks can be calculated based on a number of ways, including distance, shape, size, and topological relations. Zhang gives a fuzzy topological surface matching method [17]. Tong put forward a theory of probability matching model [18]. And Masuyama calculate the possibility of matching based on overlap area ratio of two blocks [19].

Our web page similarity definition is based on a concept called corresponding blocks. A pair of corresponding blocks A_i and B_i come from pages A and B respectively, and they are visually close to each other. Intuitively, if each A_i in a suspicious page A has a corresponding block B_i in a genuine page B, A is a phishing page that imitates B. We propose two rigorous definitions for corresponding blocks.

Definition 4. ***(OA based Corresponding Blocks)*** *Given blocks A_i and B_i that are extracted by a same method from web pages A and B respectively, if the size difference (both width and height) between A_i and B_i is less than a threshold T_{size}, and the ratio between the their overlap area and max(Area(A_i), Area(B_i)) is larger than a threshold $T_{overlap}$, A_i and B_i are considered as corresponding blocks.*

The idea behind OA based corresponding blocks is that if the overlap occupies a very high portion of either block, these two blocks are regard as a pair matching in phishing detection. An example is shown in Figure 5. The size difference between the left two blocks does not exceed a pre-specified threshold, and the ratio between their overlap area and the larger block area exceeds a pre-specified threshold. Therefore, the two blocks are determined to be corresponding blocks. In contrast, the two blocks in the right have a small portion as overlap, and they have a large difference in height. As a result, they are not corresponding blocks.

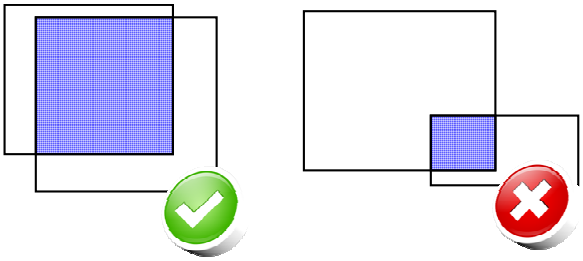


Figure 5: OA based block matching

Definition 5. ***(CDA based Corresponding Blocks)*** *Given blocks A_i and B_i that are extracted by a same method from web pages A and B respectively, if the Euclidean distance between their centers is less than a threshold T_{dist}, and their size difference (both width and height) is less than a threshold T_{size}, A_i and B_i are considered as Center Distance and Area (CDA) based corresponding blocks.*

On the other hand, the CDA method is based on the center distance and area. If the center distance between the query block and a block in the feature library is less than the predetermined threshold and their size difference is sufficiently small, they are matched as corresponding blocks. For example, the left two blocks are corresponding blocks in Figure 6. In contrast, although the center distance of the right two blocks does not exceed the threshold, their height difference is too large, and therefore they are not corresponding blocks.

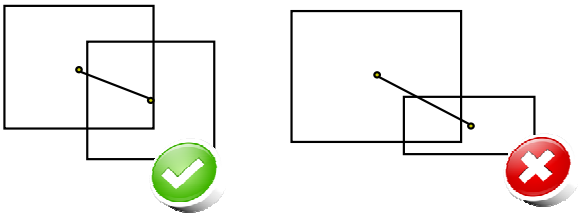


Figure 6: CDA based block matching

Given a query page and a page in the feature library, we can find all pairs of corresponding blocks between them according to either Definition 4 or Definition 5. In Section 4 we will present how to obtain all such corresponding blocks by making use of R-tree on the feature library. After obtaining all corresponding block between two pages, we are ready to compute the similarity between them.

3.3 Computing similarity based on spatial layout features

In this section, we use an example shown in Figure 7 to illustrate the similarity computation. Figure 7 shows the layouts of web pages A and B. Here A has 5 blocks and B has 6 blocks. We normalize both pages into the same coordinate system. With appropriate thresholds, we can find that A-1 and B-1 are corresponding blocks, A-2 and B-2 are corresponding blocks, A-5 and B-3 are corresponding blocks, and so are A-4 and B-4. In total, there are four pairs of corresponding blocks between pages A and B.

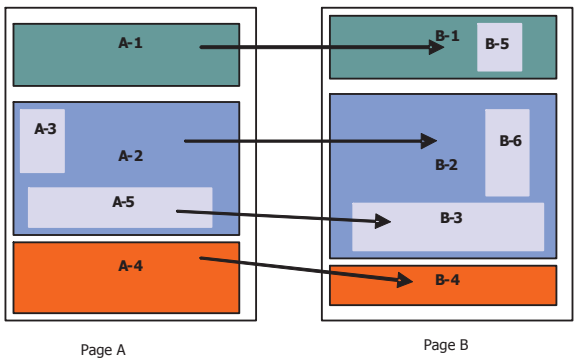


Figure 7: Corresponding blocks between two web pages

With the concept of corresponding blocks, we define the layout similarity between two web pages as follows in For-

mula 1. Here, n_A is the number of blocks in web page A, n_B is the number of blocks in web page B, while n_{cor} is the number of corresponding blocks between A and B.

$$Sim(A,B) = (1 - \frac{|n_A - n_B|}{max(n_A, n_B)}) \cdot \frac{n_{cor}^2}{n_A \cdot n_B} \quad (1)$$

Formula 1 calculates the ratio of the corresponding blocks in either page, normalizes the product of the two ratios with respect to n_A as well as n_B, and uses the result as the layout similarity between two pages A and B.

Alternatively, we can directly use the number of corresponding blocks n_{cor} as the similarity. We call this page similarity *Common block Number* (CN). Or, we can use the ratio of the corresponding blocks n_{cor}/n_A as the similarity when we decide whether page A is a phishing page. We call this page similarity *Common block Number Ratio* (CNR). In Section 5, both CN and CNR page similarity definitions will be implemented and compared with the one defined in Formula 1.

On the basis of extracting the page rendering features, the similarity of page signatures can be computed as described above. Further, page signatures can mix up text signatures, image signatures, overall picture signatures, layout signatures, and so on. The structure and search of the feature library have direct impact on the phishing detection speed. To speed up layout based similarity search in the detection, we create an R-tree to index all web page spatial layout features in the library.

4 Spatial layout feature based phishing detection

4.1 Motivation

Given the signatures of two web pages, their similarities can be calculated by simple matching [13], N largest match [20], EMD method [3, 21], Bipartite graph based matching method [16, 22], etc. However, for web phishing detection, a suspicious web page should be compared to the feature library whose size has a significant impact on the phishing detection speed. In practice, the feature library is often large and it needs to be frequently updated. This demands an efficient filtering method that quickly filters out irrelevant web sites from the feature library and enables to compare the suspicious page with a limited number of candidates from the feature library.

The bipartite graph based matching method uses some features to pruning web pages in the feature library, e.g., the number of DOM tree nodes, and the number of image nodes in web pages. Although this method improves the speed of phishing detection to some extent, it needs to traverse the entire feature library with time complexity. Therefore, this method does not apply to the case of large feature libraries.

Motivated as such, we propose to build a spatial index for the layout features of web pages in order to speed up the search of the feature library.

4.2 Indexing spatial layout features of web pages

R-tree is a tree based indexing data structure similar to B tree. It is widely used in spatial databases for indexing spatial data, which supports efficient processing of various spatial queries. For example, we can easily use R-tree to search for all gas stations within two kilometers to the current location. In R-trees, each spatial object is approximated by a minimum bounding rectangle (MBR), and MBRs are grouped to larger but fewer MBRs according to specific rules. The grouping of MBR is conducted recursively until a final single MBR is reached which is corresponds to the root of the R-tree.

Each node of an R tree has a certain number of entries. Each non-leaf node entry stores two fields: the address (pointer) of the corresponding child node, and the MBR of the corresponding child node. Each leaf-node entry stores the address (identifier) of the corresponding object and the object's MBR. It is noteworthy that each non-leaf node's MBR, stored in its corresponding entry in its parent node, is the MBR of all its child nodes' MBR. Based on the data partitioning described above as well as the consequent MBR containment property, R-tree can prune unpromising nodes in query processing and thus shorten the query time significantly.

In this paper, we use an R-tree to index the spatial features of all the web pages in the feature library. In particular, we create an R-tree and insert to it each spatial feature captured in an MBR for all web pages in the feature library. Each block is represented as $\langle pageID, blockID, MBR, pointer\rangle$, where $pageID$ is the identifier of the page that contains the block, $block$ is the identifier of the block within its page, MBR is the block's MBR, and $pointer$ points to other relevant information of that block. The overall indexing scheme is shown in Figure 8.

In particular, the Page Hash Table maps a given page identifier to the address of a Block Hash Table for that page. There are multiple Block Hash Tables, each for a specific page. Each such a Block Hash Table maps a block identifier to the R-tree leaf node where the block's index entry is contained.

4.3 Searching spatial layout feature library

Given a suspicious web page s that may be a phishing page, we need to find those legitimate pages that s is similar to. Here we employ the space layout based similarity to measure how similar two web pages are. All legitimate web pages in the feature library are organized based on their spatial layout features and indexed as described in Section 4.2. In other words, the phishing detection is reduced to a similarity search of the feature library using the spatial layout based page similarity metric.

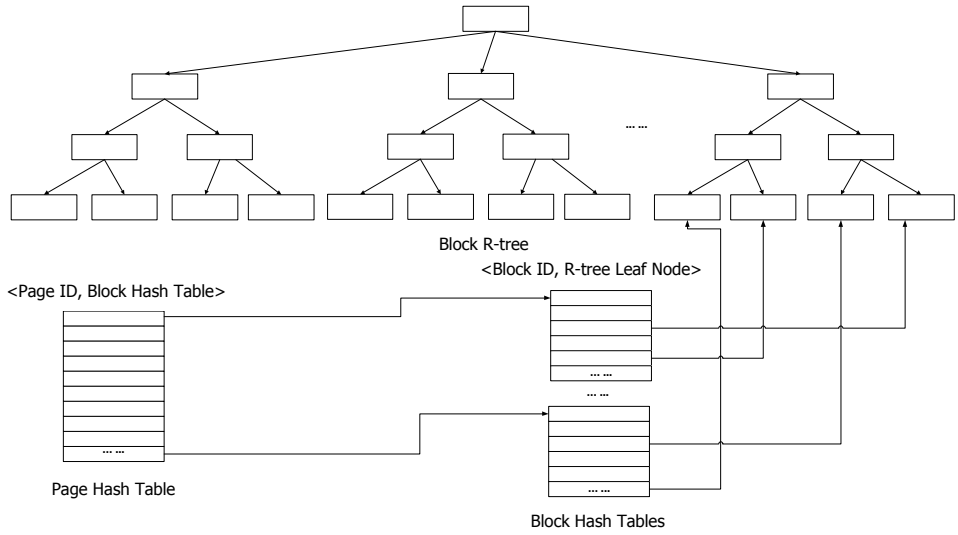


Figure 8: Index of the Library of Page Spatial Layout Features

The overall page similarity search is encapsulated in a function **pageSearch**. Its pseudo code is shown in Algorithm 1. It takes as input a suspicious web page s, the legitimate page feature library FL, and two integer numbers, N and K, for controlling the sizes of corresponding block matches and similar page matches respectively.

Algorithm 1 pageSearch(Suspicious web page s, legitimate web page feature library FL, number of blocks to match N, number of similar pages to return K)

1: $bs \leftarrow$ obtain all blocks from s ▷ Either DOM tree or image segmentation is employed here, depending on how the feature library is constituted.
2: $mbs \leftarrow$ initialize an array of $|bs|$ elements
3: $pages \leftarrow$ initialize a hash table that maps a page ID to a list of block IDs
4: **for** each block b_i in bs **do**
5: $mbs[i] \leftarrow$ topMatch(FL's R-tree, b, N)
6: **for** each $\langle PageID, BlockID\rangle$ in $mbs[i]$ **do**
7: add $\langle PageID, BlockID\rangle$ to $pages$
8: $H \leftarrow$ initialize a max-heap
9: **for** each page p that appears in $pages$ **do**
10: $score \leftarrow$ SIM(p, s) ▷ According to Equation 1
11: push $\langle p, score\rangle$ to H
12: **return** top-K pages in H

In particular, the algorithm first obtains the blocks of a given suspicious page s, and put them in bs (line 1). Here, either s's DOM tree or an image segmentation method is applied to generate all the blocks, depending on how the spatial layout features in the legitimate library are generated. Subsequently, for each block b in bs, it calls function topMatch that searches through the block R-tree to get the top-N corresponding block matches for b (line 5). Note OA based corresponding block matching is shown in Algorithm 2, and CDA based matching is shown in Algorithm 4.

Referring to Algorithm 2, OA based matching employs a max-heap H for matching blocks and a recursive search function **queryOA** via the library's block R-tree (line 3). The recursive function is shown in Algorithm 3. It conducts a depth-first traversal on the R-tree, and only considers nodes that overlap with the current block b (lines 3, 14, and 18). The overlapping blocks are kept in the max-heap H. Finally, the top-N ones are returned by Algorithm 2.

Algorithm 2 topMatchOA(legitimate web page feature library RF's R-tree R_{RF}, block b, integer value N)

1: $H \leftarrow$ initialize a max-heap
2: $OA_N \leftarrow 0$
3: queryOA(R_{RF}, b, OA_N, H)
4: **return** the top-N elements of H

The CDA based block match works as shown in Algorithm 4. Taking the mindist [23] metric between an R-tree node and the current block b as the key, it visits all block R-tree nodes in a best-first manner [24]. During the process, only those blocks with similar size are considered (line 6). Once the first N closest and most similar blocks are found, indicated by the result size, the algorithms returns the top-N matches (lines 10–11).

Refer to the page search (Algorithm 1) again. After the top-N corresponding block matches are obtained, it adds each result record $\langle PageID, BlockID\rangle$ to a list $pages$ (lines 6–7). After all blocks in the suspicious page s are processed likewise, the page search algorithm calculates the similarity between s and each page that appears in $pages$, and returns the top-K pages with the highest similarity scores (lines 8–12). The returned K pages will be used for further check to decide whether s is a phishing page or not. For that purpose, other page features like texts, images can be used to take a closer comparison between each returned page and the suspicious page s.

As a remark, we use the page spatial layout based simi-

Algorithm 3 queryOA(R-tree node n, block b, the current Nth overlap area OA_N, the current matched block heap H)

```
1: if n is a leaf node then
2:     for each block nb indexed by n do
3:         if nb does not overlap with b then continue
4:         if |H| < N then
5:             push ⟨nb.pageID, nb.blockID, OA(nb, b)⟩ to H
6:             if |H| = N then
7:                 update OA_N is necessary
8:         else if OA(nb, b) > OA_N then
9:             push ⟨nb.pageID, nb.blockID, OA(nb, b)⟩ to H
10:            update OA_N is necessary
11: else
12:    if |H| < N then
13:        for each child node cn of n do
14:            if cn overlaps with b then
15:                queryOA(cn, b, OA_N, H)
16:    else
17:        for each child node cn of n do
18:            if cn overlaps with b and OA(cn, b) > OA_N then
19:                queryOA(cn, b, OA_N, H)
```

Algorithm 4 topMatchCDA(legitimate web page feature library RF's R-tree R_{RF}, block b, integer value N)

```
1: result ← initialize an empty set
2: H ← initialize a min-heap
3: push ⟨R_RF.root, 0⟩ to H
4: while H is not empty do
5:     pop H's top element to ⟨n, dist⟩
6:     if node n is a block and n has comparable size length as b then add ⟨n.pageID, nb.blockID⟩ to result
7:     else
8:         for each child node cn of n do
9:             push ⟨cn, mindist(cn.center, b.center)⟩ to H
10:    if |result| = N then break
11: return result
```

larity SIM (line 10) in Algorithm 1. This similarity definition can be replaced by the other two page similarities CN and CNR that are defined in Section 3.3.

4.4 Overall phishing detection implementation

We use the above algorithms presented in Section 4.3 to compute the similarities between a suspicious web page and relevant pages in the legitimate library. Note that we do not compute such a similarity with respect to every page in the library. Instead, we prune irrelevant pages in the library through the R-tree.

The overall phishing detection procedure is shown in Figure 9. A specifically-designed browser plug-in records the protected web URLs that require username and password, extracts the spatial layout features (and other signatures like text and/or image) of that page, and stores the information in the feature library. The spatial layout feature R-tree is updated accordingly when the new piece of information is inserted to the feature library.

When a user accesses a current web page that also requires username and password but has a different URL from the stored one, the phishing detection will be invoked. The current page is used as the suspicious one.

The spatial layout features (and signatures) of the current web page are then extracted by the plug-in. Next, a set of K candidate web pages are returned by the page search algorithm (Algorithm 1 in Section 4.3). Further, the similarity of signatures between the suspicious page and the feature library is computed. When the similarity is greater than the given threshold, an alarm is raised to alert the user that she/he may be accessing a phishing web site.

To ease the phishing detection we can apply the threshold strategy directly to the spatial layout feature based similarity in deciding whether a suspicious page is a phishing page or not. In particular, if one of the computed similarities (in Algorithm 1) is greater than a pre-specified threshold, the suspicious web page will be considered as a phishing web page directly without involving further signatures.

Note that the feature library can be stored either locally or on a remote server. In the latter scenario, the browser plug-in will send the current URL to the server when a user visits a web page that requires a password. The server will conduct the phishing detection for the URL and return the detection result to the client browser.

5 Experimental study

In this section, we evaluate the spatial layout feature based phishing detection through a series of experiments. We investigate the filtering performance of the spatial layout feature based library filtering, and the overall effectiveness of phishing detection based on spatial layout similarity.

5.1 Performance metrics

Throughout our experiments, we have the following variables relevant to the numbers of different web pages involved in phishing detection.

- A is the number of *phishing* web pages that are detected as *phishing* web pages.
- B is the number of *normal* web pages that are detected as *phishing* web pages.
- C is the number of *phishing* web pages that are detected as *normal* web pages.
- D is the number of *normal* web pages that are detected as *normal* web pages.

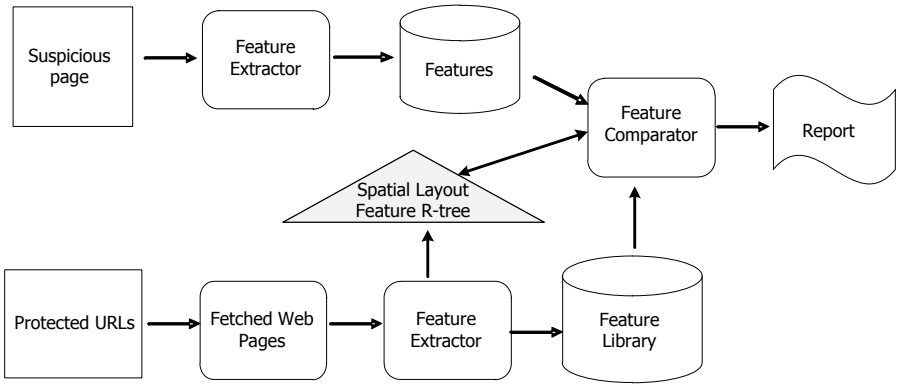


Figure 9: Spatial layout similarity based phishing detection

Accordingly, we consider the following performance metrics in our experimental study.

- Precision = A/(A+B)
- Recall = A/(A+C)
- True positive rate (TPR) = A/(A+C)
- False positive rate (FPR) = B/(B+D)

Precision is the ratio of correct reports in all phishing page reports. Recall describes the detected proportion of all phishing pages. These two evaluations are mutually exclusive. A high precision means a low recall, while a low precision means a high recall. For phishing detection, recall is more important and small numbers of incorrect reports of phishing pages are acceptable because the security is the major concern.

In the experiments, phishing web pages are referred as positive instances, and the normal web pages are referred as negative instances. TPR represents the probability that true phishing web pages are reported ass phishing web pages; FPR represents the probability that normal web pages are incorrectly reported as phishing web pages. Different thresholds can be used in obtaining the corresponding TPR and FPR, as well as the corresponding ROC curve. Accordingly, the corresponding AUC (area under the curve) can be calculated [4].

If the legitimate web page targeted by a phishing page is reserved after the feature library filtering, i.e., it is not pruned out, we call it a hit. Accordingly, we consider another performance metric *hit rate*. It is the number of hits divided by the total number of phishing web pages.

5.2 Experimental settings

We compare the design options listed in Table 1. All experiments are implemented in javascript and Java. They are run on a Pentium 5 desktop PC with double 2.6GHz CPUs, 2GB main memory. The PC runs on Windows XP SP3, and has a Mozilla Firefox 3.0 web browser.

We makes use of the data collected from web site PhishTank [25]. As an open and free anti-phishing web site, it allows users to easily submit the suspicioused web sites which will be confirmed by experts. We do not directly retrieve phishing web sites because most phishing web pages do not exist for a long period of time.

Specifically, we retrieve 100 positive pairs of English phishing pages and their corresponding real-world legitimate web pages. We also collect 100 negative legitimate English web pages from banks, credit unions and online services according to Yahoo! Directory [26]. In the experiments, the 100 target web pages of phishing web pages are stored in the feature library; the 100 corresponding phishing web pages (positive examples) and the 100 general web pages (negative examples) are used as suspicious web pages.

For each legitimate or suspicious web page, its DOM tree based spatial layout features are obtained by calling a Firefox plug-in. Whereas the image segmentation based spatial layout features are obtained using the nested Earth Mover's Distance based image segmentation [21].

5.3 Results

5.3.1 Hit rates

We first investigate the hit rates of different design options. Specifically, we compare 12 different methods that are obtained by selecting one option for each of the three phases in Table 1. The results are shown in Figures 10 to 13. As K increases, all methods get higher hit rates.

Figure 10 reports the results of those methods that use OA block matching following DOM tree based block generation. For small (less than 5) and large (larger than 18) K values, OA-SIM achieves the highest hit rates. Whereas all three methods perform very closely. This shows that the spatial layout feature based similarity (SIM) is able to improve the hit rates.

Figure 11 reports the results of those methods that use CDA block matching following DOM tree based block generation. For four particular K values (8, 9, 10, 11), all three methods have the same hit rate. Apart from that, CDA-SIM clearly outperforms the other two methods for

Table 1: Design Options

Phase	Options
Block generation	DOM tree based (DOM), Image segmentation based (IMG)
Corresponding Blocks Match	Overlap Area (OA), Center Distance and Area (CDA)
Page similarity	CN, CNR, SIM (Equ. 1)

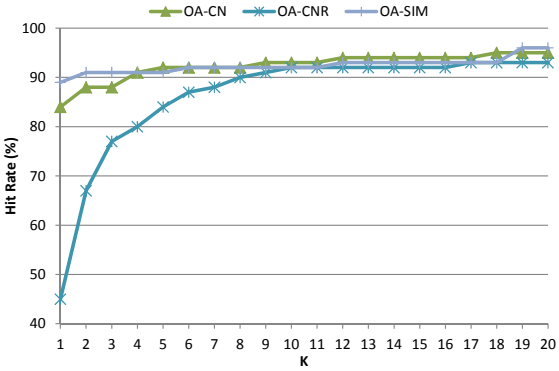


Figure 10: OA block matching for DOM tree segmentation

all K values. This suggests that SIM is a very good similarity definition that is able to get high hit rates.

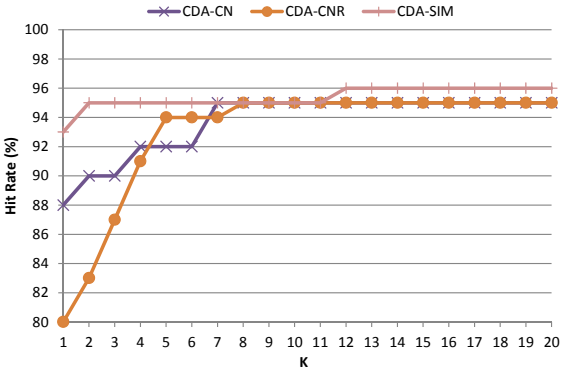


Figure 11: CDA block matching for DOM tree segmentation

In addition, comparing Figure 10 and 11 indicates that CDA is a better corresponding block matching option as it leads to higher hit rates that OA. Given to blocks, CDA takes into account not only their size difference but also the distance between their centers. Therefore, CDA captures corresponding blocks better than OA that considers overlapping areas only. A large overlapping area does not necessary mean two blocks are visually similar and close to each other.

Figures 12 and 13 report the results of those methods that use image segmentation based block generation, followed by OA and CDA matching respectively.

According to the results shown in Figure 12, SIM works very well as a novel similarity definition for methods using OA block matching. The method OA-SIM obtains the highest hit rates for all K values except 19 and 20 where OA-CN gets better with very slight differences.

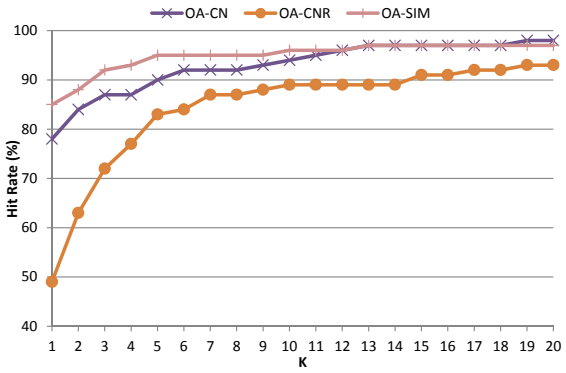


Figure 12: OA block matching for image segmentation

SIM still performs well for methods using CDA block matching, according to the results reported in Figure 13. All methods here are less steady, which indicates that CDA is more sensitive to the image segmentation based block generation.

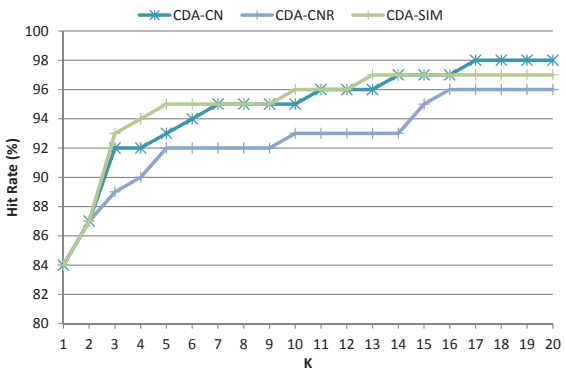


Figure 13: CDA block matching for image segmentation

To summarize, image segmentation based block generation yields better performance than DOM tree based block generation in terms of hit rates. This is because the segmentation captures the visual effects and spatial layout features better for web pages rendered in a browser. Also, CDA captures corresponding blocks better than does OA.

5.3.2 TPR and FPR

We investigate both TPR and FPR for different methods. They are two important indicators of the phishing detection accuracy. In particular, we study the receiver operating characteristic by plotting both indicators in ROC curves.

The results are reported in Figures 14 and 15, for DOM tree based block generation and image segmentation based block generation respectively.

Both ROC curves are convex compared to their corresponding main diagonals. This shows all three methods in comparison (OA-CNR, OA-SIM and CDA-SIM) are effective in phishing detection. Moreover, the curves of CDA-SIM have larger AUC (area under the curve) in both figures. This indicates that CDA-SIM is the best among all the three in consideration.

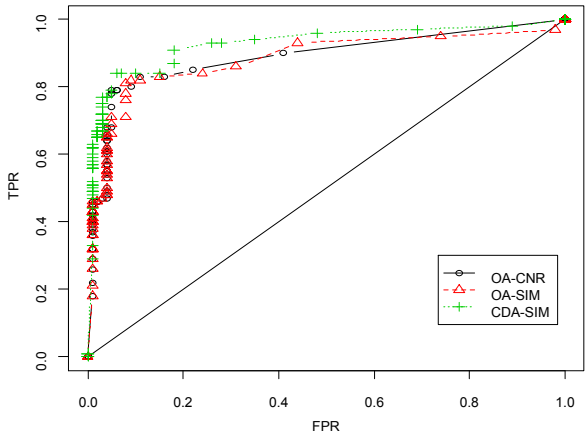


Figure 14: ROC curve of methods using DOM tree segmentation

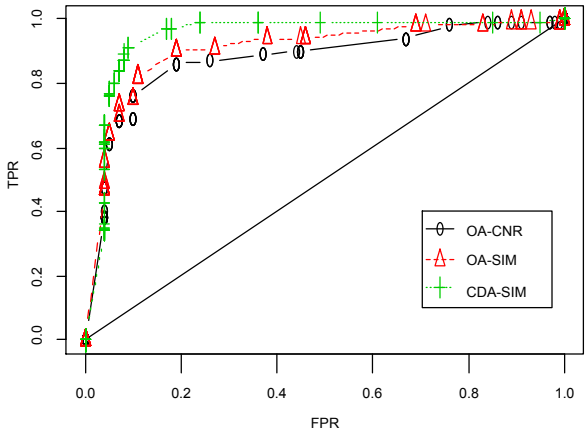


Figure 15: ROC curve of methods using image segmentation

5.3.3 Precision and recall

Next, we study another two performance metrics, namely precision and recall, for methods that use SIM as the page similarity. We omit CN and CNR because the results from previous experiments demonstrate that SIM has the best overall performance. We compare DOM tree based block generation and image based block generation, followed by OA or CDA block matching. The results are listed in Table 2.

The highest precision (0.933) is achieved by DOM-CDA-SIM. While the highest recall (0.919) is achieved by IMG-CDA-SIM. This demonstrates that CDA-SIM is a very good combination for phishing detection because it is able to achieve both high precision and high recall.

We also calculate the F1 score for each method using the following formula:

$$\text{F1 score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

The results are also reported in Table 2. Among all the four methods in comparison, IMG-CDA-SIM is overall the best method for phishing detection as it achieves the highest F1 score.

5.3.4 Library search time

Finally, we study the execution time of the proposed spatial layout based phishing detection. In particular, we measure the execution time that is spent on search the legitimate web page library. We compare a sequential scan method with our R-tree facilitated library search. A sequential scan compares a given suspicious web page with each legitimate page in the legitimate library. We vary the R-tree fanout from 5 to 15 in the index for spatial layout features. We vary the legitimate library size from 1 to 100 to see its effect on the library search efficiency. The results on library search time are reported in Figure 16.

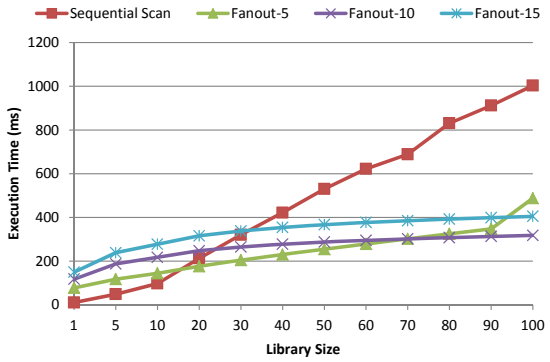


Figure 16: Legitimate library search time comparison

The sequential scan works the best for very a small legitimate library with no more than 10 pages. As the library size increases, the propose spatial layout feature based filtering catches up and clearly outperforms the sequential scan when the library contains more than 30 pages. A legitimate library usually is large with tens or even hundreds of legitimate pages, it is therefore beneficial to employ the proposed spatial layout feature based search to filter out irrelevant legitimate pages quickly. This can speed up the overall phishing detection.

Table 2: Results of Precision and Recall

Method	Precision	Recall	F1 Score
DOM-OA-SIM	0.911	0.837	0.872
DOM-CDA-SIM	0.933	0.857	0.894
IMG-OA-SIM	0.826	0.909	0.865
IMG-CDA-SIM	0.910	0.919	0.915

6 Conclusion and future work directions

In this paper, we propose a spatial layout similarity based approach for phishing web detection. This novel approach takes into account important spatial layouts of web pages. For this approach, we first invent two meaningful methods that extract the spatial layout features from web pages. After obtaining such spatial layout features, we define a similarity function to quantize how visually similar two web pages are. Such a similarity measurement indicates how a suspicious page is a phishing one in relation to a legitimate web page. In order to speed up searching the legitimate feature library, we further design an R-tree index for all spatial layout features in the library and develop search algorithms accordingly. Finally, we evaluate the proposed approach through a series of experiments. The results demonstrate the effectiveness and efficiency of our proposal.

Several directions exist for future work. First, it is of interest to combine the spatial layout features proposed in this paper with other types of features available for web pages. For example, text features of web pages can be extracted together with spatial layout features. As a result, phishing detection can make use of a mix of different types of features.

Second, it is relevant to integrate DOM tree and image segmentation in web page block generation. The former is easy to implement with accessible web browser APIs; while the latter captures the visual and spatial layouts of web pages more closely to the way humans do. Combining these two methods may generate blocks that are more decisive in phishing detection.

Third, the library search algorithms (Section 4) in this paper can be further optimized by processing relevant spatial queries in a collective way. Such optimization is more relevant when the legitimate page library is large.

Acknowledgments

Weifeng Zhang's work was supported by the National Natural Science Foundation of China under Grant No.61272080 and No.61100135, Opening Foundation of Guangxi Key Laboratory of Trustworthy Software, and Opening Foundation of Jiangsu Key Laboratory of Computer Information Processing Technology in Soochow University (Grant No.KJS0714).

References

[1] Yue Zhang, Jason Hong, and Lorrie Cranor (2007). Cantina: a content-based approach to detecting phishing web sites. In *WWW*.

[2] (2011). Netcraft Anti-phishing Toolbar. http://toolbar.netcraft.com.

[3] Anthony Y. Fu, WenYin Liu, and XiaoTie Deng (2006). Detecting phishing web pages with visual similarity assessment based on earth mover's distance (EMD). *IEEE Trans. Dependable Sec. Comput.*, 3(4):301–311.

[4] Yue Jiang, Bojan Cukic, and Yan Ma (2008). Techniques for evaluating fault prediction models. *Empirical Software Engineering*, 13(5):561–595.

[5] R. Song, H. Liu, J.R. Wen, and W.Y. Ma (2004). Learning blockimportance models for webpages. *In Proceedings of the WWW*.

[6] I. N. Fette, Sadeh, and A. Tomasic (2006). Learning to detect phishing emails. ISRI Technical report, CMU-ISRI-06-112.

[7] André Bergholz, Jeong Ho Chang, Gerhard Paass, Frank Reichartz, and Siehyun Strobel (2008). Improved phishing detection using model-based features. In *CEAS*.

[8] André Bergholz, Gerhard Paass, Frank Reichartz, Siehyun Strobel, Marie-Francine Moens, and Brian Witten (2008). Detecting known and new salting tricks in unwanted emails. In *CEAS*.

[9] Ziv Bar-Yossef and Maxim Gurevich (2009). Estimating the impressionrank of web pages. In *WWW*, pages 41–50.

[10] (2011). McAfee SiteAdvisor. http://www.siteadvisor.com.

[11] Tyler Moore and Richard Clayton (2009). Evil searching: Compromise and recompromise of internet hosts for phishing. In *Financial Cryptography*, pages 256–272.

[12] WenYin Liu, GuangLin Huang, XiaoYue Liu, Min Zhang, and XiaoTie Deng (2005). Detection of phishing webpages based on visual similarity. In *WWW*

(Special interest tracks and posters), pages 1060–1061.

[13] Angelo P. E. Rosiello, Engin Kirda, Christopher Kruegel, Fabrizio Ferr, and Politecnico Di Milano (2007). A layout-similarity-based approach for detecting phishing pages. In *SecureComm*, pages 454–463.

[14] Ponnurangam Kumaraguru, Ro Acquisti, Lorrie Faith Cranor, and Jason Hong (2010). Teaching johnny not to fall for phish. *ACM Trans. Internet Techn.*, 10(2).

[15] WenYin Liu, XiaoTie Deng, GuangLin Huang, and Anthony Y. Fu (2006). An antiphishing strategy based on visual similarity assessment. *IEEE Internet Computing*, 10(2):58–65.

[16] WeiFeng Zhang, YuMing Zhou, Lei Xu, and BaoWen Xu (2010). A method of detecting phishing web pages based on hungarian matching algorithm. *Chinese Journal of Computers*, 33(10):1963–1975.

[17] QiaoPing Zhang, DeRen Li, and JianYa Gong (2004). Surface entity matching techniques of city map database. *International Journal of Remote Sensing*, 8(2):107–112.

[18] XiaoHua Tong, SuSu Deng, and WenZhong Shi (2007). Probability based map entity matching method. *International Journal of Surveying and Mapping*, 36(2):210–217.

[19] A Masuyama (2006). Methods for detecting apparent differences between spatial tessellations at different time points. *International Journal of Geographical Information Science*, 20(6):633–648.

[20] E. Medvet, E. Kirda, and C. Kruegel (2008). Visual-similarity-based phishing detection. In *SecureComm*, pages 1–6.

[21] JiuXin Cao, Bo Mao, JunZhou Luo, and Bo Liu (2009). A phishing web pages detection algorithm based on nested structure of earth mover's distance (Nested-EMD). *Chinese Journal of Computers*, 32(5):922–929.

[22] Harold W. Kuhn (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

[23] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent (1995). Nearest neighbor queries. In *SIGMOD Conference*, pages 71–79.

[24] Gísli R. Hjaltason and Hanan Samet (1999). Distance browsing in spatial databases. *ACM Trans. Database Syst.*, 24(2):265–318.

[25] (2011). PhishTank. http://www.phishtank.com.

[26] (2011). Yahoo! Directory. http://dir.yahoo.com/.

Influence of CNF Encodings of AtMost-1 Constraints on UNSAT-based PMSAT Solvers

Mohamed El Bachir Menai
Department of Computer Science, College of Computer and Information Sciences
King Saud University, P.O.Box 51178, Riyadh 11543, Saudi Arabia
E-mail: menai@ksu.edu.sa
http://faculty.ksu.edu.sa/menai

Tasniem Nasser Al-Yahya
Department of Computer Science, College of Computer and Information Sciences
King Saud University, P.O.Box 51178, Riyadh 11543, Saudi Arabia
E-mail: talyahya@ccis.imamu.edu.sa

Keywords: artificial intelligence, satisfiability problems, constraint satisfaction, boolean cardinality constraint, CNF encoding, AtMost-1 constraints

Received: July 7, 2012

Partial maximum Boolean satisfiability (Partial MaxSAT or PMSAT) is an optimization variant of Boolean Satisability (SAT). It asks to find a variable assignment that satisfies all hard clauses and the maximum number of soft clauses in a Boolean formula. Several exact PMSAT solvers have been developed since the introduction of the MaxSAT evaluations in 2006, based mainly on the Davis- Putnam-Logemann-Loveland (DPLL) procedure and branch and bound (B&B) algorithms. One recent approach that provides an alternative to B&B algorithms is based on unsatisfiable (UNSAT) core identification. All PMSAT algorithms based on UNSAT identification are dependent on two essential external components: (1) a cardinality constraint encoder for encoding AtMost-1 constraints into conjunctive normal form (CNF); and (2) a SAT solver. Ensuring the effectiveness of both components directly affects the performance of the PMSAT solver. Whereas great advances have been made in PMSAT algorithms based on UNSAT core identification, only a few research work has been conducted to understand the influence of CNF encoding methods on the performance of PMSAT solvers. In this paper, we investigate the influence of three CNF encoding methods for AtMost-1 constraints on an UNSAT-based PMSAT solver. We implement the solver using the pairwise, parallel, and sequential encodings, and evaluate its performance on industrial instances. The experimental results show the impact of the CNF encoding method on the performance of the PMSAT solver. Overall, the best results were obtained with the sequential encoding.

Povzetek: Predstavljena je nova metoda PMSAT, t.j. optimirane variante izpolnjivosti Boolovih enačb (SAT).

1 Introduction

The Boolean satisfiability (SAT) problem is the core of computationally intractable NP-complete problems [7]. Informally, SAT asks if a Boolean expression can be made $True$ by assigning Boolean values to its variables. The maximum satisfiability (MaxSAT) [12] problem is an optimization version of SAT, which consists in finding an assignment that maximizes the number of clauses that are $True$ in a CNF formula. In recent years there has been an increasing interest in designing and implementing MaxSAT solvers. Indeed, state-of-the-art MaxSAT solvers are able to solve large number of instances that were beyond the reach of solvers developed just few years ago.

The partial maximum satisfiability (PMSAT) [5, 22] is a problem between SAT and MaxSAT that is better suited for modeling and solving over constrained problems such as packing, planning, and scheduling. PMSAT instances can be solved using either an exact or stochastic local search method. Exact methods are complete, but are limited to small problem instances. Many exact methods for PMSAT have been introduced in the 2007-2011 SAT conferences, e.g. [10, 14, 15]. Stochastic local search methods for PMSAT can handle very large problem instances, but do not guarantee that an optimal solution will be provided, especially as the instance grows in size [5, 21].

In 2007, the MaxSAT evaluation devoted a special track for exact PMSAT solvers. Most PMSAT solvers submitted to past MaxSAT evaluations are based on B&B methods, e.g. IncWMaxSatz [14, 15] and WMaxSatz+ [13], or DPLL methods, e.g. QMaxSAT and PM2 [1]. The 2007-2011 MaxSAT evaluations show that PMSAT solvers based on B&B methods are superior in the random category, while PMSAT solvers based on DPLL are the best in the

industrial category. DPLL-based solvers for PMSAT are built on top of powerful DPLL-based SAT solvers. Many take advantage of the improvements to DPLL-based SAT solvers, including unsatisfiable (UNSAT) core generation. All PMSAT algorithms based on UNSAT core generation are dependent upon two essential external components: (1) a CNF encoder for Boolean cardinality constraints, which expresses constraints in CNF representation; and (2) a SAT solver, which checks the satisfiability state of the SAT instance and extracts UNSAT cores.

This paper studies the influence of three CNF encoding methods for AtMost-1 constraints on UNSAT-based PMSAT solvers and gives experimental evidence of their impact on the performance of the PMSAT solvers. The rest of the paper is structured as follows. Brief overview of the topics addressed in the paper, namely the CNF encoding problem and PMSAT algorithms using UNSAT core generation are given in Section 2. Related work to CNF encoding methods for AtMost-k constraints on SAT solvers is presented in Section 3. The proposed method and the design of a PMSAT solver with three different encodings for the AtMost-1 constraint are explained in Section 4. Experimental results are presented and analyzed in Section 5. The paper is concluded in Section 6.

2 Background

Boolean cardinality constraints state that at most one Boolean variable (AtMost-1) is allowed to be $True$). Many problems are expressed by CNF clauses and AtMost-1 constraints, such as mixed Horn formulas, planning, and PMSAT [2]. Because it was generally believed that solving such problems through pure CNF encoding is inefficient, many authors have proposed specialized algorithms: Pseudo-Boolean solvers. However, it has been shown [3] that an appropriate CNF encoding method and a robust SAT solver can provide a competitive approach, thus allowing modern SAT techniques, e.g. clause learning, restarts, etc., to be fully functional without the necessity of adapting PMSAT solvers within a mixed ad-hoc solver.

PMSAT was first defined in 1996 by Miyazaki et al. [22], in the context of the optimization of database queries. A PMSAT instance is a CNF formula in which some clauses are soft and the rest are hard. Solving a PMSAT instance consists in finding an assignment that satisfies all the hard clauses and the maximum number of soft clauses.

The following sub-sections discuss the properties of a CNF encoding method and give a quick overview on UNSAT-based PMSAT methods.

2.1 CNF encoding problem

The goal of the CNF encoding problem is the correct and efficient translation of a Boolean cardinality constraint over a set of Boolean variables into a CNF formula. Although there is no general definition of a good encoding technique, such encodings are generally judged in terms of their correctness, consistency enforcing level, and encoding size.

2.1.1 Correctness

Correctness is an essential property that must be preserved in CNF encoding methods. It is formally presented in Definition 1.

Definition 1. *Given a CNF formula F over $X = \{x_1, x_2, ..., x_n\}$, F is said to encode a Boolean cardinality constraint, e.g. $(x_1, x_2, ..., x_n) \leq k$, correctly if and only if, for any complete truth assignment I on X, I satisfies F if and only if I satisfies $(x_1, x_2, ..., x_n) \leq k$.*

2.1.2 Consistency enforcing level

Consistency enforcing methods appeared in the context of constraint satisfaction problems (CSPs) as inference methods to assist searching. It has been shown that applying consistency enforcing methods to CSP instances defined with constraints only in extension, i.e., Boolean cardinality constraints, is equivalent to applying unit propagation to a polynomial SAT encoding of the constraints [11]. Consistency enforcing methods that infer constraints based on pairs of variables are referred to as arc consistency (AC) algorithms. AC ensures that any legal value in the domain of a single variable has a legal match in the domain of any other selected variable. The strength of consistency achieved by the unit propagation rule is reflected in the DPLL performance in many problems. However, achieving stronger consistency is not always beneficial, because of the additional time and space overheads. Thus, there is a trade-off between the effort spent on consistency and that spent on subsequent DPLL search. To date, according to the present research, AC and AC$^+$ are the only consistency enforcing methods incorporated into CNF encoding methods. The AC property is defined below:

Definition 2. *Given a CNF formula F over $X = \{x_1, x_2, ..., x_n\}$, F is said to preserve the AC property when encoding a Boolean cardinality constraint, e.g. $(x_1, x_2, ..., x_n) \leq k$ if and only if, for any partial truth assignment I on X, unit propagation restores AC for $(x_1, x_2, ..., x_n) \leq k$, specifically:*

1. *Unit propagation produces an empty clause when more than k variables in X are assigned to be $True$.*
2. *Unit propagation assigns to $False$ all other variables in X when k variables in X are assigned to be $True$.*
3. *Unit propagation assigns to $True$ all other variables in X when $n - k$ variables in X are assigned to be $False$.*

The definition of the AC$^+$ property is given below:

Definition 3. *Given a CNF formula F over $X = \{x_1, x_2, ..., x_n\}$, F is said to preserve the AC$^+$ property when encoding a Boolean cardinality constraint, e.g.*

$(x_1, x_2, ..., x_n) \leq k$, if and only if, for any complete truth assignment I on X, applying unit propagation on F assigns the same variables of X as restoring AC$^+$ for $(x_1, x_2, ..., x_n) \leq k$, specifically:

1. *Whenever an empty clause is not generated, and*
2. *All the variables of X are assigned.*

Table 1: Summary of CNF encoding methods. The table entry n indicates: Number of variables to be encoded. The columns "Auxiliary variables" and "Auxiliary clauses" indicate the number of new variables and clauses introduced by the encoding, respectively. The "AC/AC$^+$" column shows whether unit propagation for the encoding enforces AC, AC$^+$, both, or neither.

CNF encoding method	Auxiliary variables	Auxiliary clauses	AC /AC$^+$
Pairwise	none	$\frac{n(n-1)}{2}$	Both
Sequential	$n-1$	$2n+n-4$	Both
Parallel	$2n-1$	$\leq 7n - 3\lfloor \log n \rfloor - 6$	none

2.1.3 Encoding size

In practice, reducing the size, i.e., the number of literals or variables, of the resultant CNF formula in an encoding does not guarantee enhanced performance, regardless of how this is measured. Nevertheless, small encoding size is worth aiming for; because computational results can be unpredictable, all else being equal, a smaller encoding is preferable.

The three CNF encoding methods of AtMost-1 cardinality constraints considered in this paper are: pairwise, parallel counter, and sequential counter encoding methods [24]. Table 1 summarizes these methods and their properties. The encodings are presented in the order in which they were introduced.

2.2 UNSAT-based PMSAT solvers

UNSAT-based methods consist in using iteratively SAT solvers to identify and relax UNSAT formulas in PMSAT instances. Any UNSAT subset of clauses in an UNSAT formula is called an UNSAT core. Fu and Malik [10] proposed the first UNSAT-based solver for PMSAT, that consists of the following steps: (1) Identification of UNSAT sub-formulas; (2) Relaxation of each clause in each UNSAT sub-formula by adding a relaxation variable to each clause in an UNSAT sub-formula; and (3) Addition of new Equals-1 constraint indicating that exactly one of these relaxation variables can be assigned the value $True$.

The 2008-2011 MaxSAT evaluations have seen many competitive PMSAT solvers based on the Fu&Malik method, including MSU1.2 [18, 19], MSU4.0 [18], MSUnCore [16, 18-20], PM2 [1], and WPM1 [1].

3 Related work

This section provides some insights into works that compare the performance of various CNF encoding methods for AtMost-k constraints, for different MaxSAT instances.

In [18, 19], sequential counters [24], sorter networks, binary decision diagram (BDD), and bitwise encodings [9] are proposed as alternative encodings to the pairwise encoding used in the Fu&Malik algorithm. The experiments conducted on some selected MaxSAT instances show that the best results were obtained with BDD and bitwise encodings. According to the study in [18], one may conclude that the bitwise encoding is the most appropriate for AtMost-1 constraints.

Marques-Silva and Planes [20], compare BDD and sorting network encoding, in terms of CPU time. Their results show that sorting network encoding is less time consuming than BDD encoding, with a few outliers.

The pairwise, bitwise [9], and sequential counters [24] encodings are tested on a wide range of industrial MaxSAT instances [23]. The results indicate that bitwise encoding is the best one, as already suggested in [18], whereas the performance of the sequential counters is considered as quite good.

4 Proposed method

All UNSAT-based PMSAT solvers are built on top of powerful SAT solvers. Thus, the studies in [6, 8, 17] have motivated the investigation of CNF encoding methods of Boolean cardinality constraints on the performance of UNSAT-based PMSAT solvers. The most appropriate combination of: (1) Boolean cardinality constraint type; (2) CNF encoding method; and (3) SAT solver; are based on the arguments below.

Marques-Silva and Planes [19] have shown that clauses introduced by Equals-1 constraint used in the original Fu&Malik algorithm [10] can be reduced by replacing it with an AtMost-1 constraint, while maintaining the correctness of the algorithm. In order to quantify the impact of CNF encoding methods on the performance of UNSAT-based PMSAT solvers, we implement and test a PMSAT solver based on the following CNF encoding methods for AtMost-1 constraint: pairwise, parallel, and sequential encodings [24]. In the literature only few competitive SAT solvers generating UNSAT cores are described. Research and discussions with some developers of SAT solvers led to the selection of PicoSAT-936 [4] as the SAT/UNSAT solver that will be used to extract UNSAT cores for our PMSAT solver. PicoSAT-936 [4] is a state-of-the-art conflict-driven clause learning (CDCL) SAT solver that comes with the PicoMUS utility to compute minimal UNSAT cores.

Fu&Malik(α)
Input : α is PMSAT instance
Output:
  if α *satisfiable* **then**
    Number of soft clauses falsified in α
  else
    ∞
$Optimal = 0$
while $True$ **do**
  (α_{SAT}) = PMSATtoSAT(α) (call PMSATtoSAT to convert PMSAT instance α to SAT instance α_{SAT})
  $State$ = PicoSAT-936(α_{SAT}) (call PicoSAT-936 solver & return $True$ if α satisfiable, otherwise $False$)
  if $State == True$ **then**
    return $Optimal$
  α_{UNSAT} = PicoMUS(α_{SAT}) (call PicoMUS utility & return minimal UNSAT cores α_{UNSAT})
  $\beta = \emptyset$
  foreach $soft\_clause \in \alpha_{UNSAT}$ **do**
    $\beta = \beta \cup a_{new}$ (a_{new} is a new auxiliary variable)
    $\alpha = (\alpha \backslash soft\_clause) \cup (soft\_clause \vee a_{new})$
  if $\beta == \emptyset$ **then**
    return ∞
  $CNF = Encode\_AtMost - 1(\sum_{a \in \beta} a \leq 1)$ (encode AtMost-1 cardinality constraint to CNF clauses using: Pairwise, parallel, or sequentail encoding)
  $\alpha = \alpha \cup CNF$ (add AtMost-1 cardinality constraint)
  $Optimal = Optimal + 1$

Figure 1: General procedure for proposed method

Algorithm 1 shows the general steps of the proposed UNSAT-based PMSAT solver. The algorithm consists in iteratively calling PicoSAT-936 on a converted PMSAT to SAT instance. PicoSAT-936 determines whether the formula is satisfiable or not, and in case the instance is unsatisfiable, it gives an unsatisfiable core by calling the PicoMUS utility. At this point, the algorithm produces new variables, relaxation variables, one for each soft clause in the unsatisfiable core. The new working instance consists in adding the new variables to the soft clause in the unsatisfiable core, adding a cardinality constraint saying that at most one of the new variables should be $True$ using: pairwise, parallel, or sequential encoding [24]. This procedure is applied until PicoSAT-936 returns satisfiable.

5 Empirical evaluation

This section presents an experimental evaluation of the performance of the UNSAT-based PMSAT solver using the three encodings. The main goal is to assess the effect of the CNF encoding methods, i.e., pairwise, parallel, and sequential. Three versions of our UNSAT-based PMSAT solver, related to CNF encoding methods, were tested: PMSAT_UNSAT/PW (pairwise encoding), PMSAT_UNSAT/PC (parallel encoding), and PMSAT_UNSAT/SC (sequential encoding).

Moreover, the performance of the solver under the three encodings is compared with that of another well-known UNSAT-based solver for PMSAT described in the literature. The solver selected for the comparison is MSUnCore [16, 18-20], introduced in the 2009 MaxSAT evaluation and ranked in $5th$ place. The evaluation is based on CPU time, i.e., the time spent executing user code and system functions. Experiments were performed on an Intel core 2/Linux machine running at 2.50GHz with 4GB of memory. The time limit for the experiments with each instance of each data set was set to 30 minutes. Benchmarks drawn from the 2010 MaxSAT evaluation were chosen. Challenging industrial instance sets were carefully selected (see Table 2).

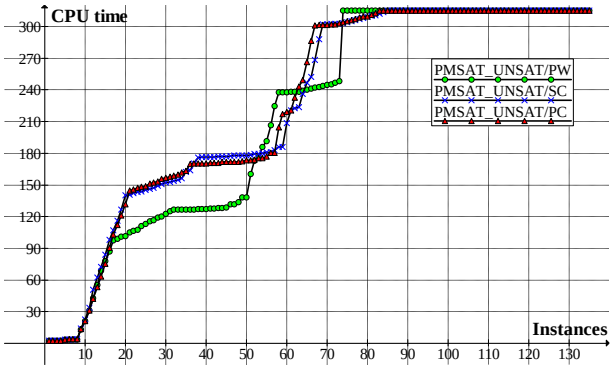


Figure 2: Run time results for selected 2010 industrial instances. The y axis represents the CPU time in seconds.

Figure 1 plots cumulative run time distributions for the solver under the three encodings, for industrial benchmark category. It can be seen that the performance difference of the three encodings considered differs significantly. Figure 1 also shows that PMSAT_UNSAT/PW exhibits a sharp transition in its behavior. Up to a certain point, PMSAT_UNSAT/PW is competitive with the other two encodings. However, as the instances become more complex, the number of clauses added by the pairwise encoding increases quadratically. This produces larger UNSAT cores and increases the computational cost of the pro-

Table 2: Summary of selected industrial instance sets for 2010 benchmarks.

Set Name	No. of instances	Min. nb. of variables - Max. nb. of variables	Min. nb. of clauses - Max. nb. of clauses
Haplotype-Assembly/	6	11642-19918	37730-59200
bcp-fir/	59	54-203287	112-583176
bcp-hipp-yRa1/SU/	38	5352-8334	81318-209643
bcp-hipp-yRa1/simp/	17	166-4477	431-51775
pbo-routing/	15	996-4028	2755-11763

cess. As a result, the performance of PMSAT_UNSAT/PW declines and the computation usually aborts due to excessive memory requirements. As shown in Figure 1, PMSAT_UNSAT/PC and PMSAT_UNSAT/SC exhibit similar behavior, i.e. both have the same transition shape. It can be seen that with the exception of a few outliers, PMSAT_UNSAT/SC outperforms PMSAT_UNSAT/PC, however the differences are essentially negligible. This is explained by the fact that although PMSAT_UNSAT/SC makes use of a sequential, unary counter that requires more clauses for encoding than is the case with PMSAT_UNSAT/PC, PMSAT_UNSAT/SC enforces AC/AC$^+$ properties. Thus, in contrast to PMSAT_UNSAT/PC, a solution can be found faster, and no search is required to check whether or not the constraint is fulfilled.

A summary of the number of solved instances is presented in Table 3. For the industrial category, PMSAT_UNSAT/SC solved two more instances than PMSAT_UNSAT/PC (84 as opposed to 82) and, as expected, both solved significantly more instances than PMSAT_UNSAT/PW. The previous results demonstrate that UNSAT-based solvers for PMSAT are effective in solving problem instances obtained from industrial settings.

Figures 2, 3, and 4 compare MSUnCore with the solver under the three encodings, for the industrial category. The close up of Figure 2 shows that PMSAT_UNSAT/PW outperformed MSUnCore for some instances. But overall, MSUnCore performance was better. Figure 3 shows that MSUnCore outperformed PMSAT_UNSAT/PC for almost all problem instances. Figure 4 shows that the performances of MSUnCore and PMSAT_UNSAT/SC are comparable, although MSUnCore is an improved variant of the Fu&Malik [10] procedure that is implemented with the latest techniques for handling hardware requirements. Finally, these results provide experimental evidence of the influence of CNF encodings of AtMost-1 constraints on UNSAT-based PMSAT solvers.

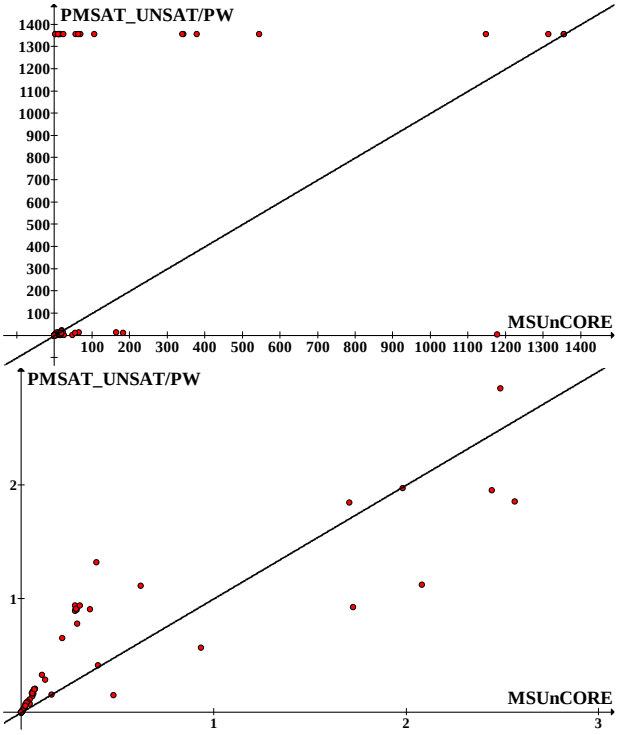


Figure 3: Scatter plots for selected industrial instances: PMSAT_UNSAT/PW versus MSUnCore. x and y axis represent the CPU time in seconds.

6 Conclusions and future work

This paper has presented an empirical study of the influence of the three CNF encoding methods for AtMost-1 constraints on UNSAT-based PMSAT solver: pairwise, parallel, and sequential, with regard to the original Fu&Malik [10] algorithm. An PMSAT solver based on these CNF encoding methods has been implemented and compared to MSUnCore. Overall, empirical results on selected PMSAT instances drawn from the 2010 MaxSAT evaluation have shown that the sequential encoding is the most competitive encoding method among the three, while it was shown to present an unstable behavior on SAT instances [6, 8, 17].

In a future work, we intend to include more CNF encoding methods for AtMost-1 constraints, such as bitwise encoding, binary decision diagram, commander, and sorting network methods. Moreover, we plan to study the effect of different Boolean cardinality constraint types, such as AtLeast-1, Equals-1, and AtMost-k, on UNSAT-based solvers for PMSAT.

Acknowledgement

We thank the anonymous referees for their helpful comments on the manuscript.

Table 3: Performance results on 2010 industrial benchmark. The table's entry of the form $x/y(z)$ corresponds to:x :Number of solved instances, y :Total number of instances, and z : Total CPU time in seconds to solve x instances. Some instances could not be solved for memory issues: 6 instances(*), 4 instances(**), and 1 instance(***).

Set Name	PMSAT_ UNSAT/PW	PMSAT_ UNSAT/PC	PMSAT_ UNSAT/SC	MSUnCore
bcp-hipp-yRa1/simp/	8/17 (3.77)	8/17 (4.09)	8/17 (3.97)	8/17 (5.39)
bcp-hipp-yRa1/SU/	9/38 (93.54)	13/38 (140.68)	12/38 (136.01)	12/38 (3190.4)
bcp-fir/	36/59 (79.0)*	41/59 (87.9)**	44/59 (96.3)***	51/59 (4470.9)
Haplotype-Assembly/	5/6 (61.1)	5/6 (68.4)	5/6 (65.6)	5/6 (72.8)
pbo-routing/	15/15 (10.8)	15/15 (11.4)	15/15 (11.4)	15/15 (4.6)
Total	73/135 (248.21)	82/135 (312.47)	84/135 (318.68)	91/135 (7744.09)

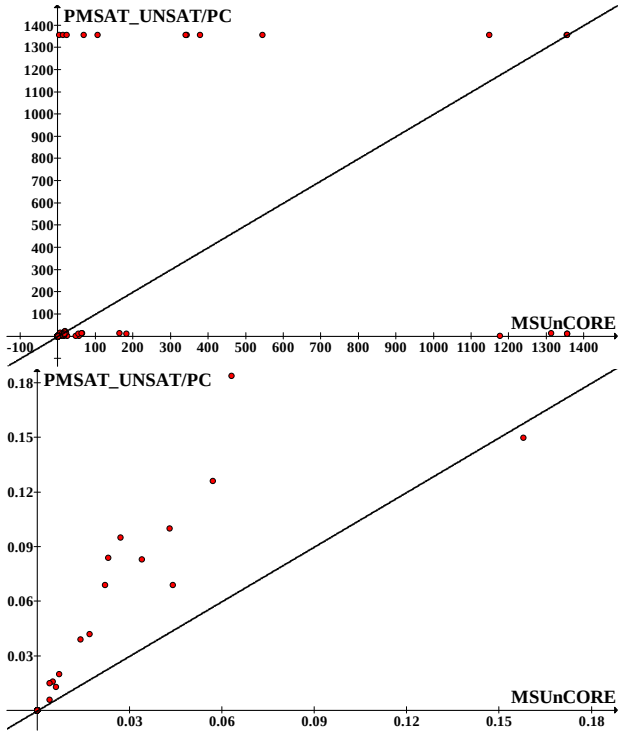


Figure 4: Scatter plots for selected industrial instances: PMSAT_UNSAT/PC versus MSUnCore. x and y axis represent the CPU time in seconds.

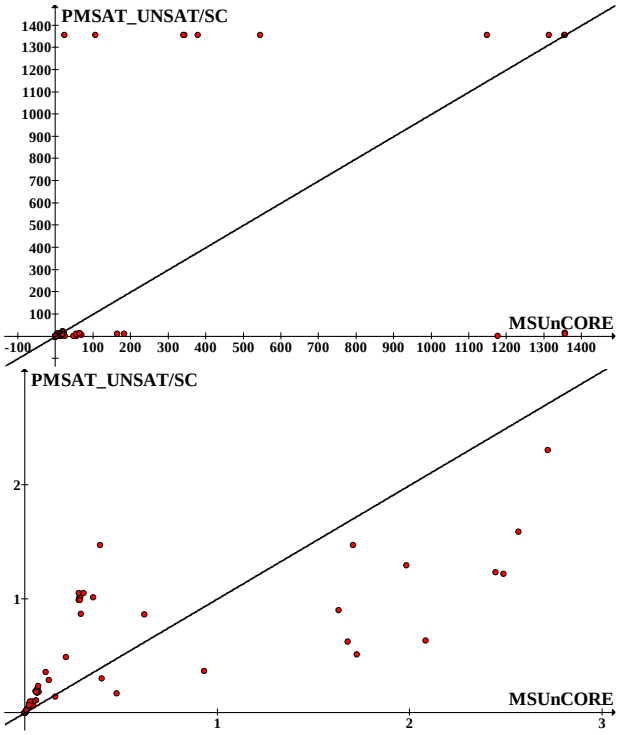


Figure 5: Scatter plots for selected industrial instances: PMSAT_UNSAT/SC versus MSUnCore. x and y axis represent the CPU time in seconds.

References

[1] C. Ansótegui, M. L. Bonet, and J. Levy (2009). Solving (Weighted) Partial MaxSAT through satisfiability testing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 427–440.

[2] J. Argelich, A. Cabiscol, I. Lynce, and F. Manyà (2009). Sequential encodings from Max-CSP into Partial Max-SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 161–166.

[3] O. Bailleux and Y. Boufkhad (2003). Efficient CNF encoding of boolean cardinality constraints. In Francesca Rossi, editor, *Principles and Practice of Constraint Programming - CP 2003*, LNCS 2833, pages 108–122.

[4] A. Biere (2008). PicoSAT essentials. *Journal on Satisfiability, Boolean Modeling, and Computation*, 4(2-4):75–97.

[5] B. Cha, K. Iwama, Y. Kambayashi, and S. Miyazaki (1997). Local search algorithms for Partial MAXSAT. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Conference on Innovative applications of Artificial Intelligence, AAAI'97/IAAI'97*, pages 263–268.

[6] J.-C. Chen (2010). A new SAT encoding of the At-Most-One constraint. In *Proceedings of the 10th*

Workshop of Constraint Modelling and Reformulation.

[7] S. A. Cook (1971). The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM symposium on Theory of computing, STOC'71*, pages 151–158, New York, NY, USA.

[8] A. Frisch and P. Giannaros (2010). SAT encoding of boolean cardinality, some old, some new, some fast, some slow. *Manuscript*, pages 195–201.

[9] A. Frisch, T. Peugniez, A. Doggett, and P. Nightingale (2005). Solving non-boolean satisfiability problems with stochastic local search: A study of encodings. *Journal of Automated Reasoning*, 35:143–179.

[10] Z. Fu and S. Malik (2006). On solving the Partial MAX-SAT problem. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing, SAT'06*, LNCS 4121, pages 252–265.

[11] I. P. Gent. Arc consistency in SAT (2002). In Frank van Harmelen, editor, *Proceedings of the 15th Eureopean Conference on Artificial Intelligence*, Eureopean Conference on Artificial Intelligence, pages 121–125, Lyon, France.

[12] D. S. Johnson (1974). Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278.

[13] C.-M. Li, F. Manyà, N. Mohamedou, and J. Planes (2009). Exploiting cycle structures in Max-SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 467–480.

[14] H. Lin and K. Su (2007). Exploiting inference rules to compute lower bounds for MAX-SAT solving. In *Proceedings of the 20th International joint Conference on Artificial Intelligence*, pages 2334–2339, San Francisco, CA, USA.

[15] H. Lin, K. Su, and C.-M. Li (2008). Within-problem learning for efficient lower bound computation in Max-SAT solving. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, volume 1, pages 351–356.

[16] V. Manquinho, J. Marques-Silva, and J. Planes (2009). Algorithms for weighted boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing, SAT'09*, LNCS 5584, pages 495–508.

[17] J. Marques-Silva and I. Lynce (2007). Towards robust CNF encodings of cardinality constraints. In *International Conference on Principles and Practice of Constraint Programming, CP 2007*, LNCS 4741, pages 483–497.

[18] J. Marques-Silva and V. Manquinho (2008). Towards more effective unsatisfiability-based maximum satisfiability algorithms. In *Proceedings of the 11th international conference on Theory and applications of satisfiability testing, SAT'08*, LNCS 4996, pages 225–230.

[19] J. Marques-Silva and J. Planes (2007). On using unsatisfiability for solving maximum satisfiability. *Computing Research Repository*, abs/0712.1097.

[20] J. Marques-Silva and J. Planes (2008). Algorithms for maximum satisfiability using unsatisfiable cores. In *Proceedings of the Conference on Design, Automation and Test in Europe, DATE'08*, pages 408–413, New York, NY, USA.

[21] M. B. Menai (2005). A two-phase backbone-based search heuristic for Partial MAX-SAT: An initial investigation. In M. Ali and F. Esposito, editors, *Innovations in Applied Artificial Intelligence, 18th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2005*, Bari, Italy, June 22-24, 2005, LNAI 3533, pages 681–684.

[22] S. Miyazaki, K. Iwama, and Y. Kambayashi (1996). Database queries as combinatorial optimization problems. In *Proceeding of 1st International Symposium on Cooperative Database Systems for Advanced Applications*, pages 477–483.

[23] F. Morgado and J. Marques-Silva (2011). The MSUnCore MAXSAT solver. Pragmatics of SAT 2011, Workshop of the SAT'11 conference, Ann Arbor, USA.

[24] C. Sinz (2005). Towards an optimal CNF encoding of boolean cardinality constraints. In Peter van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, LNCS 3709, pages 827–831.

Pruning the Computation of Distributed Shortest Paths in Power-law Networks

Gianlorenzo D'Angelo
Department of Mathematics and Informatics, University of Perugia,
Via Vanvitelli, 1, I-06123, Perugia, Italy
E-mail: gianlorenzo.dangelo@dmi.unipg.it

Mattia D'Emidio and Daniele Frigioni
Department of Computer Science, Information Engineering and Mathematics,
University of L'Aquila, Via G. Gronchi, 18, I-67100, L'Aquila, Italy
E-mail: mattia.demidio@univaq.it, daniele.frigioni@univaq.it

Keywords: communication networks, distributed algorithms, distance-vector algorithms, shortest paths, experimental analysis.

Received: October 24, 2012

We propose a general, simple and practical technique, named Distributed Leafs Pruning *(*DLP*), which can be combined with every distance vector routing algorithm based on shortest paths, allowing to reduce the total number of messages sent by that algorithm. We combine the new technique with three algorithms known in the literature:* DUAL*, which is loop-free and is part of CISCO's widely used EIGRP protocol;* DUST*, which has been shown to be effective on networks with power law node degree distribution, although it suffers of looping;* LFR*, which has been very recently introduced, is loop-free and has been shown to be very effective on real networks. We give experimental evidence that these combinations lead to an important gain in terms of the number of messages sent by* DUAL*,* DUST *and* LFR*, on networks having a power-law node degree distribution. We also notice that, in many cases the use of* DLP *determines a gain in terms of the maximum and the average space occupancy per node.*

Povzetek: Članek predlaga novo metodo DLP, ki jo je moč integrirati v poljuben algoritem za iskanje najkrajših poti v omrežjih.

1 Introduction

Shortest paths is surely one of the most important and studied combinatorial problems in the literature. In particular, the problem of computing and updating efficiently all-pairs shortest paths in a distributed network whose topology dynamically changes over the time is considered crucial in today's communication networks. The solutions found in the literature for this problem can be classified as *distance-vector* and *link-state* algorithms.

In a distance-vector algorithm, a node needs to know and possibly to store the distances from each of its neighbors to every other node in the network. This information is used to compute the distance and the next node in the shortest path to each destination, which are stored in a data structure usually called routing table. Most of the distance-vector solutions for the distributed shortest paths problem proposed in the literature (e.g., see [5, 8, 13, 14, 16, 21, 22, 24]) rely on the classical Distributed Bellman-Ford method (DBF from now on), originally introduced in the Arpanet [17], which is still used in real networks and implemented in the RIP protocol. DBF has been shown to converge to the correct distances if the link weights stabilize and all cycles have positive lengths [3]. However, the convergence can be very slow (possibly infinity) due to the well-known *looping* and *count-to-infinity* phenomena. A loop is a path induced by the routing table entries, such that the path visits the same node more than once before reaching the destination. A node "counts to infinity" when it increments its distance to a destination until it reaches a predefined maximum distance value.

In a link-state algorithm, as for example the *OSPF* protocol widely used in the Internet (e.g., see [18]), a node must know the entire network topology to compute its distance to any network destination (usually running the centralized Dijkstra's algorithm for shortest paths). Link-state algorithms are free of the looping and count-to-infinity problems. However, if a network change occurs, each node needs to receive up-to-date information on the entire network topology. This is achieved by broadcasting each change of the network topology to all nodes [18] and by using a centralized dynamic algorithm for shortest paths, as for example that in [12].

In the last years, there has been a renewed interest in devising new light-weight distributed shortest paths solutions for large-scale Ethernet networks (see, e.g., [9, 11, 19, 23,

25, 26]), where distance-vector algorithms seem to be an attractive alternative to link-state solutions when scalability and reliability are key issues or when the memory power of the nodes of the network is limited.

1.1 Related work

Notwithstanding this increasing interest, the most important distance vector algorithm is still DUAL (Diffuse Update ALgorithm) [13], which is free of the looping and count-to-infinity phenomena, thus resulting an effective practical solution (it is in fact part of CISCO's widely used EIGRP protocol). Another distance vector algorithm is DUST (Distributed Update of Shortest paThs), which has been first introduced in [6] and successively developed in [7]. Compared with DUAL, DUST suffers of the looping and count-to-infinity phenomena, even though it has been designed to heuristically reduce the cases where these phenomena occur. However, DUST uses an amount of data structures per node which is much smaller than those of both DBF and DUAL. In [6, 7] the practical performance of DBF, DUAL, and DUST have been measured in terms of both number of messages, and space occupancy per node on both realistic and artificial instances of the problem. Another distance vector algorithm, named LFR (Loop Free Routing), has been recently proposed in [10]. Compared with DUAL, LFR has the same theoretical message complexity but it uses an amount of data structures per node which is much smaller than that of DUAL, and slightly higher than that of DUST. Moreover, in [10] LFR has been experimentally shown to be a good compromise between the number of messages sent and the memory requirements per node with respect to DUAL on both realistic and artificial instances of the problem.

1.2 Results of the paper

In this paper, we provide a new general, simple, and practical technique, named *Distributed Leafs Pruning* (DLP), which can be combined with every distance-vector algorithm for shortest paths with the aim of overcoming some of their main limitations in large scale networks (high number of messages sent, high space occupancy per node, low scalability, poor convergence). This technique has been devised to be effective mainly in networks having a power-law node degree distribution, which are able to model many real-world networks such as the Internet, the World Wide Web, citation graphs, and some social networks [1]. The main idea of DLP relies on the following observations: a network with power-law node degree distribution with n nodes typically has a very small average node degree and a high number of nodes with unitary degree; any shortest path from a node with unitary degree v to any other node of the network has necessarily to pass through the unique neighbor of v in the network, and hence v does not provide any useful information for the distributed computation of shortest paths.

In order to check the effectiveness of DLP, we combined it with DUAL, DUST, and LFR, by obtaining three new algorithms named DUAL-DLP, DUST-DLP, and LFR-DLP, respectively. Then, we implemented the six algorithms in the OMNeT++ simulation environment [20], a network simulator which is widely used in the literature. As input to the algorithms, we considered the same instances of networks with a power-law node degree distribution used in [6, 7, 10], that is the Internet topologies of the *CAIDA IPv4 topology dataset* [15] (CAIDA - Cooperative Association for Internet Data Analysis is an association which provides data and tools for the analysis of the Internet infrastructure), and the random topologies generated by the *Barabási-Albert* algorithm [2]. The results of our experimental study can be summarized as follows: the application of DLP to DUAL, DUST and LFR provides a significant improvement in the global number of sent messages with respect to the original algorithms, and in many cases an improvement also in the maximum and in the average space occupancy per node. In particular, the ratio between the number of messages sent by DUAL-DLP and DUAL is within 0.12 and 0.48; the ratio between the number of messages sent by DUST-DLP and DUST is within 0.04 and 0.81; the ratio between the number of messages sent by LFR-DLP and LFR is within 0.36 and 0.51. Concerning the space occupancy, we have observed a gain in the maximum case for DUAL and LFR, and in the average case for DUAL.

1.3 Structure of the paper

The paper is organized as follows. In Section 2 we introduce some useful notation and definitions used in the paper. In Section 3 we describe the Distributed Leafs Pruning technique. In Section 4 we describe the combination of DLP with DUAL, DUST, and LFR. In Section 5 we give experimental evidence of the effectiveness of DLP. Finally, in Section 6 we give some concluding remarks and outline future research directions.

2 Background

We consider a network made of processors linked through communication channels that exchange data using a message passing model, in which: each processor can send messages only to its neighbors; messages are delivered to their destination within a finite delay but they might be delivered out of order; there is no shared memory among the nodes of the network; the system is asynchronous, that is, a sender of a message does not wait for the receiver to be ready to receive the message.

2.1 Graph notation

We represent the network by an undirected weighted graph $G = (V, E, w)$, where V is a finite set of nodes, one for each processor, E is a finite set of edges, one for

each communication channel, and w is a weight function $w : E \rightarrow \mathbb{R}^+ \cup \{\infty\}$ that assigns to each edge a real value representing the optimization parameter associated to the corresponding channel. We assume that the graph is connected. An edge in E that links nodes $u, v \in V$ is denoted as $\{u, v\}$. Given $v \in V$, $N(v)$ denotes the set of neighbors of v. The maximum degree of the nodes in G is denoted by *maxdeg*. A path P in G between nodes u and v is denoted as $P = (u, ..., v)$. The *weight* of P is the sum of the weights of the edges in P. A *shortest path* between nodes u and v is a path from u to v with the minimum weight. The *distance* $d(u, v)$ from u to v is the weight of a shortest path from u to v. Given two nodes $u, v \in V$, the *via* from u to v is the set of neighbors of u that belong to a shortest path from u to v. Formally: $via(u, v) \equiv \{z \in N(u) \mid d(u, v) = w(u, z) + d(z, v)\}$. Given a time t, we denote as $w^t()$, $d^t()$, and $via^t()$ the edge weight, the distance, and the via at time t, respectively. We denote a sequence of k update operations on the edges of G by $\mathcal{C} = (c_1, c_2, ..., c_k)$. Assuming $G_0 \equiv G$, we denote as G_i, $0 \leq i \leq k$, the graph obtained by applying the operation c_i to G_{i-1}. The operation c_i either inserts a new edge in G_i, or deletes an edge of G_i, or modifies (either increases or decreases) the weight of an existing edge in G_i. We consider the case in which $\mathcal{C}$ is a sequence of *weight increase* and *weight decrease* operations, that is operation c_i either increases or decreases the weight of edge $\{x_i, y_i\}$ by a quantity $\epsilon_i > 0$. The extension to *delete* and *insert* operations is straightforward, in fact deleting an edge $\{x, y\}$ is equivalent to increase $w(x, y)$ to $+\infty$, and inserting an edge $\{x, y\}$ with weight α is equivalent to decrease $w(x, y)$ from $+\infty$ to α.

2.2 Distance-vector algorithms

We consider the generic routing problem between all the nodes of a network, in which each node needs to find a shortest path to each other node. This problem can be tackled in different ways. The most reliable, robust and used approach is that based on distributed all-pairs shortest paths. We are interested in the practical case of a dynamic network in which an edge weight change (increase/decrease) can occur while one or more other edge weight changes are under processing. A processor v of the network might be affected by a subset of these changes. As a consequence, v could be involved in the concurrent executions related to such changes.

Distance-vector routing algorithms based on shortest-paths usually share a set of common features. In detail, given a graph $G = (V, E, w)$, a generic node v of G:

- knows the identity of every other node of G, the identity of all its neighbors and the weights of the edges incident to it;
- maintains and updates its own routing table that has one entry for each $s \in V$, which consists of at least two fields: $\mathtt{D}^t[v, s]$, the estimated distance between v and s at time t, and $\mathtt{VIA}^t[v, s]$, the neighbor used to forward data from v to s at time t;
- handles edge weight increases and decreases either by a single procedure (see, e.g., [13]), which we denote as WEIGHTCHANGE, or separately (see, e.g., [7]) by two procedures, which we denote as WEIGHTINCREASE and WEIGHTDECREASE;
- requests information to its neighbors and receives replies by them through a specific exchange of messages (see, e.g., message $query$ in [13] or message *get.feasible.dist* in [10]) and propagates a variation to the estimated routing information as follows: (i) if v is performing WEIGHTCHANGE, then it sends to its neighbors a message, from now on denoted as $update$; a node that receives this kind of message executes procedure UPDATE. (ii) if v is performing WEIGHTINCREASE or WEIGHTDECREASE, then it sends to its neighbors message $increase$ or $decrease$, respectively; a node that receives $increase$/$decrease$ executes procedure INCREASE/DECREASE, respectively.

2.3 Power-Law networks

Networks having a power-law node degree distribution, from now on referred as "power-law networks", are very important in practice and includes many of the currently implemented communication infrastructures, like the Internet, the World Wide Web, some social networks, and so on [1]. Practical examples of power-law networks are the Internet topologies of the *CAIDA IPv4 topology dataset* [15], and the artificial instances generated by the *Barabási-Albert* algorithm [1].

The CAIDA dataset is collected by a globally distributed set of monitors. The monitors collect data by sending probe messages continuously to destination IP addresses. Destinations are selected randomly from each routed IPv4/24 prefix on the Internet such that a random address in each prefix is probed approximately every 48 hours. The current prefix list includes approximately 7.4 million prefixes. For each destination selected, the path from the source monitor to the destination is collected, in particular, data collected for each path probed includes the set of IP addresses of the hops which form the path and the Round Trip Times (RTT) of both intermediate hops and the destination.

A Barabási–Albert topology is generated by iteratively adding one node at a time, starting from a given connected graph with at least two nodes. A newly added node is connected to any other existing nodes with a probability that is proportional to the degree of the existing nodes. In Figure 1 we show the power-law node degree distribution of a *CAIDA* network (a) and of a *Barabási-Albert* network (b).

3 The new technique

The main goal of Distributed Leafs Pruning (DLP) is to reduce the number of messages sent by a generic distance-

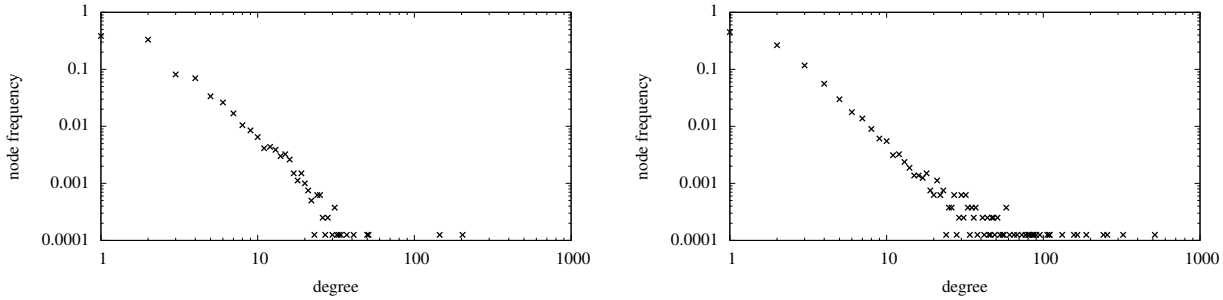


Figure 1: Power-law node degree distribution of: a *CAIDA* graph with 8000 nodes and 11141 edges (a); a *Barabási-Albert* graph with 8000 nodes and 12335 edges (b).

vector algorithm. DLP has been designed to be efficient mainly in power-law networks. The idea underlying DLP is very simple and it is based on the following observations:

- a power-law network with n nodes typically has average node degree much smaller than n and a number of nodes with unitary degree which is generally high. For example, the graphs of the *CAIDA IPv4 topology dataset* have average node degree approximately equal to $n/2000$, and a number of nodes with unitary degree approximately equal to $n/2$;
- nodes with unitary degree do not provide any useful information for the distributed computation of shortest paths. In fact, any shortest path from a node with degree one v to any other node of the network has necessarily to pass through the unique neighbor of v in the network.

To describe the technique we need to introduce some preliminary definitions. Given an undirected weighted graph $G = (V, E, w)$, the *core* of G is the graph $G_c = (V_c, E_c, w_c)$ which represents the maximal connected subgraph of G having all nodes of degree greater than one. A node $v \in V$ is a *central node* if $v \in V_c$, otherwise v is a *peripheral node*. An edge of G that links two central nodes is a *central edge*, an edge that links a central node with a peripheral node is a *peripheral edge*. For each peripheral node u, the unique central node v adjacent to u is called the *owner* of u.

3.1 Data structures

Given a generic distance-vector algorithm **A**, DLP requires that a generic node of G stores some additional information with respect to those required by **A**. In particular, a node v needs to store and update information about central and peripheral nodes and edges of G. To this aim, v maintains a data structure called *Classification Table*, denoted as CT_v, which is an array containing one entry $CT_v[s]$, for each $s \in V$, representing the list of the peripheral neighbors of s. A central node is not present in any list of CT_v. A peripheral node is present in $CT_v[s]$, for exactly one $s \in V$, and s is its owner. Each list contains at most $maxdeg$ entries and the sum of the sizes of all the lists is always smaller than n. Hence the space overhead per node due to CT_v is $O(n)$.

3.2 Properties

The main purpose of DLP is to force distributed computation to be carried out only by the central nodes. The peripheral nodes receive updates about routing information passively from the respective owners, without starting any kind of distributed computation. Then, the larger is the set of the peripheral nodes of the network, the bigger is the improvement in the global number of messages sent by the algorithm. The following lemma introduces some basic relationships between the paths that link central and peripheral nodes.

Lemma 3.1. *Given an undirected weighted graph $G = (V, E, w)$, and its core $G_c = (V_c, E_c, w_c)$, let $\{p, c\}$ be a peripheral edge such that $c \in V_c$ at time t. The following relations hold:*

- $d^t(x, p) = d^t(x, c) + w^t(c, p),\ \forall\ x \in V \setminus \{p\}$;
- $via^t(x, p) = via^t(x, c),\ \forall\ x \in V \setminus \{p\}$.

Proof. By contradiction, let us assume that $d^t(x, p) \neq d^t(x, c) + w^t(c, p)$ for a certain $x \in V \setminus \{p\}$. Then, two cases can occur: if $d^t(x, p) < d^t(x, c) + w^t(c, p)$, it follows that there exists, at time t, another path from node x to node p that does not include node c, which is a contradiction, as p is a peripheral node and has a unique adjacent node at time t; on the other hand, if $d^t(x, p) > d^t(x, c) + w^t(c, p)$ it follows that $d^t(x, p)$ is not the weight of a shortest path, which is again a contradiction. □

Some useful additional relationships can be derived from Lemma 3.1. In particular, if between the time instants t_i and t_{i+1} the weight of the edge $\{p, c\}$ between a peripheral node p and his corresponding owner c changes, that is $w^{t_i}(p, c) \neq w^{t_{i+1}}(p, c)$, then p can update its own routing

table towards each node of the network $x \in V$ simply by computing:

$$d^{t_{i+1}}(p,x) = d^{t_i}(p,x) + w^{t_{i+1}}(p,c) - w^{t_i}(p,c), \quad (1)$$

$$via^{t_{i+1}}(p,x) = \{c\}. \quad (2)$$

In a similar way, if a generic node of the network $x \in V$, between the time instants t_i and t_{i+1}, receives an update about a weight change in the path towards a generic central node c (that is, $d^{t_{i+1}}(x,c) \neq d^{t_i}(x,c)$), then the nodes involved in the change are x, c and the peripheral neighbors of c, if they exists. By Lemma 3.1, these nodes can update their estimated routing tables by computing, for all peripheral nodes p with owner c:

$$d^{t_{i+1}}(x,p) = d^{t_i}(x,p) + d^{t_{i+1}}(x,c) - d^{t_i}(x,c), \quad (3)$$

$$via^{t_{i+1}}(x,p) = via^{t_{i+1}}(x,c). \quad (4)$$

3.3 Distributed Leafs Pruning

The application of DLP to a distance vector algorithm **A** induces a new algorithm denoted as **A**-DLP. The global behavior of **A**-DLP can be summarized as follows. While in a classic routing algorithm every node performs the same code thus having the same behavior, in **A**-DLP central and peripheral nodes have different behaviors. In particular, central nodes detect changes concerning both central and peripheral edges while peripheral nodes detect changes concerning only peripheral edges. If the weight of a central edge $\{u,v\}$ changes, then node u (v, respectively) performs the procedure provided by **A** for that change only with respect to central nodes for the distributed computation of the shortest paths between all the pairs of central nodes. During this computation, if u (v, respectively) needs information by its neighbors, it asks only to neighbors in the core (see Figure 3(a)). Once u (v, respectively) has updated its own routing information, it propagates the variation to all its neighbors through the $update$, $increase$ or $decrease$ messages of **A** (Figure 3(b)). When a generic node x receives an $update$, $increase$ or $decrease$ message concerning node s, it stores the current value of $\mathtt{D}[x,s]$ in a temporary variable $\mathtt{D}_{old}[x,s]$. Now, if x is a central node, then it handles the change and updates its routing information by using the proper procedure of **A** (UPDATE, INCREASE, or DECREASE) and propagates the new information to its neighbors (see Figure 3(c)). Otherwise, x handles the change and updates its routing information towards s by using Lemma 3.1 and the data received from its owner. At the end, x calls the procedure UPDATEPERIPHERALS reported in Figure 2 using s and $\mathtt{D}_{old}[x,s]$ as parameters. If the routing table entry of s is changed (line 1), then the information about the peripheral neighbors of s, if they exist, is updated by using Equations 3 and 4 (lines 3–4).

If a weight change occurs in a peripheral edge $\{u,p\}$, then the central node u sends message $p\_change(p, w(u,p), u)$ to each of its neighbors (Figure 4(a)), while p sends a $p\_change(p, w(u,p), u)$ message to its owner u. When a generic node x receives message p_change from a node y, it simply performs procedure PERIPHERALCHANGE of Figure 5, which is a modified flooding algorithm to forward the message over the network (Figure 4(b)). Procedure PERIPHERALCHANGE first verifies at line 1 whether the message was already received by applying Lemma 3.1. If the message does not provide updated information, it is discarded. Otherwise, the procedure needs to update the data structures at x. We distinguish two cases: if x coincides with p, then the procedure updates the routing table for all the nodes $s \in V$, by using Equations 1 and 2 (see Lines 2-6). Otherwise ($x \neq p$), the procedure simply updates the routing table entry concerning p by using Lemma 3.1 (Lines 8-10). At this point, the procedure propagates the information about the change, forwarding message p_change to all the neighbors, except to nodes u and possibly p (Lines 11–12).

In what follows we provide the correctness proof of **A**-DLP which clearly depends on the correctness of **A** that is assumed. In particular, we proof the correctness of **A**-DLP for a single weight change operation. The extension to the case of multiple weight changes is straightforward.

Theorem 3.2. *Given a graph $G = (V, E, w)$, let c be a weight change operation on G, occurring at a certain time t_c. For each pair of nodes $v, s \in V$ that change their*

Event: node v invokes procedure
UPDATEPERIPHERALS$(s, \mathtt{D}_{old}[v,s])$
Procedure: UPDATEPERIPHERALS$(s, \mathtt{D}_{old}[v,s])$
1 **if** $\mathtt{D}[v,s] \neq \mathtt{D}_{old}[v,s]$ **then**
2 **foreach** $k \in CT_v[s]$ **do**
3 $\mathtt{D}[v,k] := \mathtt{D}[v,k] + \mathtt{D}[v,s] - \mathtt{D}_{old}[v,s]$
4 $\mathtt{VIA}[v,k] := \mathtt{VIA}[v,s]$
5 update any auxiliary data structures of **A**

Figure 2: Procedure UPDATEPERIPHERALS.

Event: node x receives the message
p_change$(p, w(u,p), u)$ from y
Procedure: PERIPHERALCHANGE$(p, w(u,p), u)$
1 **if** $w(u,p) \neq (\mathtt{D}[x,p] - \mathtt{D}[x,u])$ **then**
2 **if** $x \equiv p$ **then**
3 **foreach** $s \in V \setminus \{x\}$ **do**
4 $\mathtt{D}[x,s] := \mathtt{D}[x,s] - \mathtt{D}[x,u] + w(u,p)$
5 $\mathtt{VIA}[x,p] := u$
6 update any auxiliary data structures of **A**
7 **else**
8 $\mathtt{D}[x,p] := \mathtt{D}[x,u] + w(u,p)$
9 $\mathtt{VIA}[x,p] := \mathtt{VIA}[x,u]$
10 update any auxiliary data structures of **A**
11 **foreach** $k \in N(x) \setminus \{y,p\}$ **do**
12 send p_change$(p, w(u,p), u)$ to k

Figure 5: Procedure PERIPHERALCHANGE.

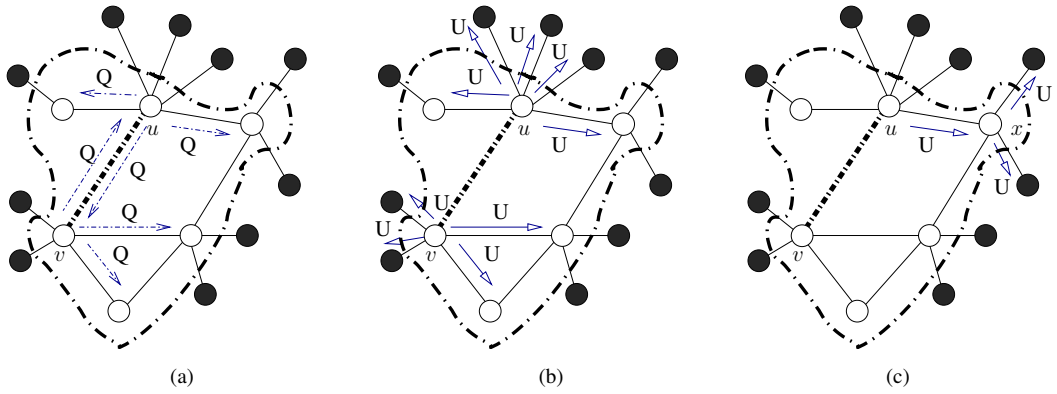


Figure 3: (a) Nodes u and v ask for information to their central neighbors by sending them message Q (query); (b) Nodes u and v propagate updated routing information to all their neighbors through message U (update); (c) A central node x receiving an update message U, propagates it to its neighbors.

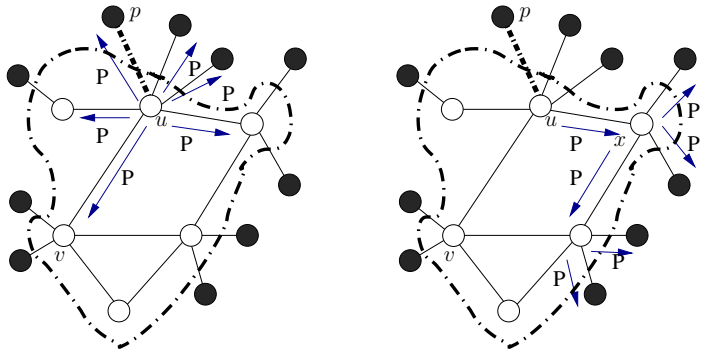


Figure 4: (a) Node u, as a consequence of a weight change on edge $\{u, p\}$, sends message p_change (P) to all its neighbors; (b) Node x receiving a message P, propagates it to the whole network.

distance as a consequence of c, there exists a time instant $t_f \geq t_c$ *such that for each* $t \geq t_f$, $\mathtt{D}[v, s](t) = d^{t_c}(v, s)$.

Proof. The proof is by case analysis on the two possible types of weight change operations on the edges: (i) central; (ii) peripheral. In what follows, we assume that the algorithm is correct at every time $t_0 < t_c$, and given any data structure $\mathtt{X}$ of **A**-DLP, we denote by $\mathtt{X}(t)$ the content of $\mathtt{X}$ at time instant t.

Case (i). If the weight of a central edge $\{x, y\}$ changes at time t_c, then node x (y, respectively) performs the procedure provided by **A** for the distributed computation of the shortest paths, only with respect to central nodes. In this step, the correctness of **A**-DLP follows from the correctness of **A**. In fact, the only difference between the behavior of **A**-DLP and that of **A** is that x (y, respectively) does not ask to peripheral nodes information concerning s, which are, for topological reasons, un-useful. Hence, the correctness of **A** guarantees that, for each central node $s \in V$, there exists a time $t_f \geq t_c$ in which **A**-DLP sets $\mathtt{D}[x, s](t_f) = d^{t_c}(x, s)$ ($\mathtt{D}[y, s](t_f) = d^{t_c}(y, s)$) and this value does not change anymore.

Once x (y, respectively) has updated its own routing information toward a central node s, it propagates the variation to all its neighbors through the $update$, $increase$ or $decrease$ messages of **A**, which carries $\mathtt{D}[x, s](t_f)$, that is the correct value $d^{t_c}(x, s)$.

When a generic node v receives an $update$, $increase$ or $decrease$ message concerning a central node s, it stores the current value of $\mathtt{D}[v, s](t_0) = d^{t_0}[v, s]$ in a temporary variable $\mathtt{D}_{old}[v, s]$. Now, if v is a central node, then it handles the change and updates its routing information by using the proper procedure of **A** (UPDATE, INCREASE, or DECREASE) and propagates the new information to its neighbors. Also in this case, the correctness of **A**-DLP follows by the correctness of **A**.

Otherwise, if v is a peripheral node whose owner is a central node c, then v handles the change and updates its routing information towards s by setting, at a certain time t_f greater than $t_c \geq t_0$, $\mathtt{D}[v, s](t_f) = \mathtt{D}[c, s](t_f) + w^{t_c}(v, c)$, where $\mathtt{D}[c, s](t_f) = d^{t_c}(c, s)$ is the received correct value. Hence, by Lemma 3.1 **A**-DLP properly assigns $\mathtt{D}[v, s](t_f) = d^{t_c}(v, s)$, and the statement of the theorem is true also in this case.

After updating the routing information toward the central node s, v calls procedure UPDATEPERIPHERALS, re-

ported in Figure 2, using s and $\mathtt{D}_{old}[v,s]$ as parameters, whose aim is to update, if needed, the routing information about the non-central nodes whose owner is s, if they exist. If the routing table entry of s is changed (line 1), then v sets, at a certain time $t_F \geq t_f$, for each peripheral node y whose owner is s, $\mathtt{D}[v,y](t_F) = \mathtt{D}[v,y](t_f) + \mathtt{D}[v,s](t_f) - \mathtt{D}_{old}[v,s](t_f)$. This assignment statement, by Lemma 3.1, is clearly correct and guarantees that $\mathtt{D}[v,y](t_F) = d^{t_c}(v,y)$ since: (i) $\mathtt{D}[v,y](t_f) = d^{t_0}(v,y)$; (ii) $\mathtt{D}[v,s](t_f)$ is the received value equal to $d^{t_c}(v,s)$; (iii) $\mathtt{D}_{old}[v,s](t_f) = d^{t_0}[v,s]$. Hence, also in this case **A**-DLP is correct.

Case (ii). If a weight change occurs in a peripheral edge $\{u,p\}$, then the central node u sends message $p\_change(p, w^{t_c}(u,p), u)$ to each of its neighbors (Figure 4(a)), while p sends a $p\_change(p, w^{t_c}(u,p), u)$ message to its owner u.

When a generic node x receives message p_change from a node y, at a certain time t, it simply performs procedure PERIPHERALCHANGE of Figure 5, which is a modified flooding algorithm to forward the message over the network (Figure 4(b)). Procedure PERIPHERALCHANGE first verifies at line 1 whether the message was already received by applying Lemma 3.1. If the message does not provide updated information, it is discarded.

Otherwise, the procedure needs to update the data structures at x. We distinguish two cases: if x coincides with p, then the procedure updates the routing table for all the nodes $s \in V$, by setting, at a certain time t_f greater than t, for all the nodes $s \in V$, $\mathtt{D}[x,s](t_f) = \mathtt{D}[x,s](t) - \mathtt{D}[x,u](t) + w^{t_c}(u,x)$, which is correct, again by Lemma 3.1, and guarantees that $\mathtt{D}[x,s](t_f) = d^{t_c}(x,s)$ as $\mathtt{D}[x,s](t) = d^{t_0}(x,s)$ and $\mathtt{D}[x,u](t) = d^{t_0}(x,u) = w^{t_0}(x,u)$. Therefore, **A**-DLP is correct in this case.

If $x \neq p$, then the procedure simply updates the routing table entry concerning p by setting, at a certain time t_f greater than t, $\mathtt{D}[x,p](t_f) = \mathtt{D}[x,u](t) + w^{t_c}(u,p)$, where $\mathtt{D}[x,u](t) = d^{t_c}(x,u)$ and $w^{t_c}(u,p)$ is the correct received value of the weight of edge (u,p). It follows that x again by Lemma 3.1 correctly assigns $\mathtt{D}[x,p](t_f) = d^{t_c}(x,p)$. At the end, the change is forwarded through the network by sending a message p_change to all the neighbors, except to nodes u and possibly p. Therefore, **A**-DLP is correct in in all cases. □

4 Combinations

In this section we briefly describe algorithms DUAL, DUST, and LFR, and how they can be combined with DLP.

4.1 Combination of DLP with DUAL

DUAL (Diffuse Update ALgorithm) [13] stores, for each node v and for each destination s, the routing table where the two fields are the distance $\mathtt{D}[v,s]$ and the *feasible successor* $\mathtt{S}[v,s]$, respectively. In order to compute $\mathtt{S}[v,s]$, DUAL requires that each node v is able to determine, for each destination s, a set of neighbors called the Feasible Successor Set, denoted as $\mathtt{FSS}[v,s]$. To this aim, each node v stores, for each $u \in N(v)$, the distance $\mathtt{D}[u,s]$ (the topology table) from u to s and computes $\mathtt{FSS}[v,s]$ by choosing neighbors which satisfy SNC, a condition, introduced in [13], that guarantees the algorithm to be loop-free. In detail, node $u \in N(v)$ satisfies SNC if the estimated distance $\mathtt{D}[u,s]$ from u to s is smaller than the estimated distance $\mathtt{D}[v,s]$ from v to s. If a neighbor $u \in N(v)$, through which the distance from v to s is minimum, is in $\mathtt{FSS}[v,s]$, then u is chosen as feasible successor. When a node v experiences a weight change operation in one of the adjacent edges, it executes procedure WEIGHTCHANGE, in order to update $\mathtt{FSS}[v,s]$.

DUAL includes an important sub-routine, named Diffuse-Computation, which is performed by a generic node v, when $\mathtt{FSS}[v,s]$ does not include the node $u \in N(v)$ through which the distance from v to s is minimum. The Diffuse-Computation works as follows: node v sends queries to all its neighbors with its distance through $\mathtt{S}[v,s]$ by using message *query*. From this point onwards v does not change its feasible successor to s until the Diffuse-Computation terminates. When a neighbor $u \in N(v)$ receives a *query*, it tries to determine if a feasible successor to s, after such update, exists. If so, it replies to the *query* by sending message *reply* containing its own distance to s. Otherwise, u continues the Diffuse-Computation: it sends out queries and waits for the replies from its neighbors before replying to v's original *query*. In [13] it has been proved that the Diffuse-Computation always terminates. When a node receives messages *reply* by all its neighbors it updates its distance and feasible successor, with the minimal value obtained by its neighbors and the neighbor that provides such distance, and finishes the Diffuse-Computation. At the end of a Diffuse-Computation, a node sends message *update* containing the new computed distance to its neighbors. A generic node that receive an *update* message handled it by performing procedure UPDATE. In order to guarantee mutual exclusion in case of multiple weight change operations, DUAL uses a finite state machine to process these multiple updates sequentially.

DUAL can be combined with DLP as described in Section 2. In addition, the generic procedures reported in Figure 2 and 5, are modified, by using the data structures of DUAL, to generate two specific procedures, called DUAL-PERIPHERALCHANGE and DUAL-UPDATEPERIPHERALS. The main changes can be summarized as follows: (i) in Procedure PERIPHERALCHANGE (at Lines 5 and 9) and in Procedure UPDATEPERIPHERALS (at Line 4) the data structure VIA is replaced by the data structure S; (ii) in Procedure UPDATEPERIPHERALS, Line 5 is removed, as the auxiliary data structures of DUAL, like e.g. the FSM data structures or the topology table, are used only in the single distributed computa-

tion phase and hence it is not necessary to store them with respect to peripheral nodes; (iii) in Procedure PERIPHERALCHANGE, Line 6 is removed for the same reason while at Line 10 the auxiliary data structures of DUAL are updated, as s is a central node, according to the algorithm.

4.2 Combination of DLP with DUST

DUST maintains only the routing table described in Section 2 and, for each node v and for each source s, $\texttt{VIA}[v,s]$ contains the set $\texttt{VIA}[v,s] \equiv \{v_i \in N(v) \mid \texttt{D}[v,s] = w(v,v_i) + \texttt{D}[v_i,s]\}$. Algorithm DUST starts every time an operation c_i on edge (x_i, y_i) occurs. Operation c_i is detected only by nodes x_i and y_i. If c_i is a *weight increase* (*weight decrease*) operation, x_i performs procedure WEIGHTINCREASE (WEIGHTDECREASE) that sends message *increase*(x_i, s) (*decrease*$(x_i, s, \texttt{D}[x_i, s])$) to y_i for each $s \in V$. Node y_i has the same behavior of x_i. If a node v receives message *decrease*$(u, s, \texttt{D}[u, s])$, then it performs procedure DECREASE, that relaxes edge (u, v). In particular, if $w(v,u) + \texttt{D}[u,s] < \texttt{D}[v,s]$, then v updates $\texttt{D}[v,s]$ and $\texttt{VIA}[v,s]$, and propagates the updated values to nodes in $N(v)$. If $w(v,u) + \texttt{D}[u,s] = \texttt{D}[v,s]$, then u is a new estimated via for v with respect to s, and hence v adds u to $\texttt{VIA}[v,s]$. If a node v receives *increase*(u, s), then it performs procedure INCREASE which checks whether the message comes from a node in $\texttt{VIA}[v,s]$. In the affirmative case, v removes u from $\texttt{VIA}[v,s]$. As a consequence, $\texttt{VIA}[v,s]$ may become empty. In this case, v computes the new estimated distance and via of v to s. To do this, v asks to each node $v_i \in N(v)$ for its current distance, by sending message *get-dist*(v, s) to v_i. When v_i receives *get-dist*(v, s) by v, it performs procedure SENDDIST which sends $\texttt{D}[v_i, s]$ to v, unless one of the following two conditions holds: (i) $\texttt{VIA}[v_i, s] \equiv \{v\}$; (ii) v_i is updating its routing table with respect to destination s. In this case v_i sends ∞ to v. When v receives the answers to the *get-dist* messages by all its neighbors, it computes the new estimated distance and via to s. If the estimated distance is increased, v sends an *increase* message to its neighbors. In any case, v sends to its neighbors *decrease*, to communicate them $\texttt{D}[v,s]$. In fact, at some point, v could have sent ∞ to a neighbor v_j. Then, v_j receives the message sent by v, and it performs procedure DECREASE to check whether $\texttt{D}[v,s]$ can determine an improvement to the value of $\texttt{D}[v_j, s]$.

DUST can be combined with DLP, by modifying its behavior as described Section 2. In addition, the generic procedures reported in Figures 2 and 5, are modified, by using the data structures of DUST, to generate two specific procedures, called DUST-PERIPHERALCHANGE and DUST-UPDATEPERIPHERALS. The main changes can be summarized as follows: (i) in Procedure PERIPHERALCHANGE (at Lines 5 and 9) and in Procedure UPDATEPERIPHERALS (at Line 4) the data structure VIA is modified to be a set instead of a single value variable; (ii) since DUST does not use any additional data structures, in Procedure PERIPHERALCHANGE Lines 6 and 10 are removed and in Procedure UPDATEPERIPHERALS Line 5 is removed.

4.3 Combination of DLP with LFR

LFR stores, for each node v, the estimated distance $\texttt{D}[v,s]$ and the *feasible via*, $\texttt{FVIA}[v,s]$, that is the node through which the distance to s is minimum and which satisfies SNC. In addition, node v maintains for each $s \in V$, the following data structures: $\texttt{ACTIVE}_v[s]$, which represents the state of node v with respect to a certain source s, in detail, v is in *active* state and $\texttt{ACTIVE}_v[s] = true$, if and only if it is trying to update $\texttt{FVIA}[v,s]$ after a generic weight change operation occurred on edge $\{v, \texttt{FVIA}[v,s]\}$; the *upper bound distance* $\texttt{UD}[v,s]$ which represents the distance from v to s through $\texttt{FVIA}_v[s]$, in particular, if v is active $\texttt{UD}[v,s]$ is always greater than or equal to $\texttt{D}[v,s]$, otherwise they coincide. In addition, in order to compute loop-free values of $\texttt{FVIA}[v,s]$, node v stores a temporary data structure tempD which is allocated only when needed, that is when v is active with respect to s, and it is deallocated when v turns back in passive state with respect to s. The entry $\texttt{tempD}[u,s]$ contains $\texttt{UD}[u][s]$, for each $u \in N(v)$.

The algorithm starts when the weight of an edge $\{x_i, y_i\}$ changes. As a consequence, x_i (y_i, respectively) performs procedure WEIGHTCHANGE, that sends to y_i (x_i, respectively) an *update* message carrying the value $\texttt{D}[x_i, s]$ ($\texttt{D}[x_i, s]$, respectively). Messages received at a node with respect to a source s are stored in a queue and processed in a FIFO order to guarantee mutual exclusion. If an arbitrary node v receives an *update* message from $u \in N(v)$, then it performs procedure UPDATE in which, basically, v compares the received value $\texttt{D}[u,s] + w(u,v)$ with $\texttt{D}[v,s]$ in order to determine whether v needs to update its estimated distance and its estimated $\texttt{FVIA}[v,s]$. The update procedure works as follows. If node v is active, the processing of the message is postponed by enqueueing it into the FIFO queue associated to s. Otherwise, if $\texttt{D}[v,s] > \texttt{D}[u,s] + w(u,v)$, then v performs a relaxing phase, updating both $\texttt{D}[v,s]$ and $\texttt{FVIA}[v,s]$, while if $\texttt{D}[v,s] > \texttt{D}[u,s] + w(u,v)$, node v performs a phase called Local-Computation in which it sends a *get.dist* to all its neighbor in order to know the corresponding estimated distances towards s. Each neighbor $u \in N(v)$ immediately replies to the *get.dist* message with its $\texttt{UD}[u,s]$. When v receives these values, it tries to determine whether a new $\texttt{FVIA}[v,s]$ exists, by comparing the received distances with $\texttt{D}[v,s]$. If this phase succeeds, node v updates its routing information and propagates the change, Otherwise, node v initiates a distributed phase, named Global-Computation. It sets $\texttt{UD}[v,s] = \texttt{UD}[\texttt{FVIA}[v,s], s] + w(v, \texttt{FVIA}[v,s])$ and sends to all its neighbors a *get.feasible.dist* message, carrying $\texttt{UD}[v,s]$. A node $k \in N(v)$ that receives such a message first verifies whether $\texttt{FVIA}[k,s] = v$ or not. In the first case, it replies immediately to v and terminates. In the second case, it performs the Local-Computation and possibly the Global-Computation, in order to update its routing

information and to reply to v. Note that this distributed procedure can involve all the nodes of the network. Finally, if $\mathtt{D}[v,s] = \mathtt{D}[u,s] + w(u,v)$, that is there exists more than one shortest path from v to s, the message is discarded and the procedure ends.

LFR can be combined with DLP, by modifying its behavior as described Section 2. In addition, the generic procedures reported in Figures 2 and 5, are modified, by using the data structures of LFR, to generate two specific procedures, called LFR-PERIPHERALCHANGE and DUST-UPDATEPERIPHERALS. The main changes can be summarized as follows: (i) in Procedure PERIPHERALCHANGE (at Lines 5 and 9) and in Procedure UPDATEPERIPHERALS (at Line 4) the data structure `VIA` is replaced by the data structure `FVIA`; (ii) in Procedure UPDATEPERIPHERALS, Line 5 is removed, as the auxiliary data structures of LFR, like e.g. `tempD` or the state `ACTIVE`, are used only in the distributed computation phases and hence it is not necessary to store them with respect to peripheral nodes; (iii) in Procedure PERIPHERALCHANGE, Line 6 is removed for the same reason while at Line 10 the auxiliary data structures are updated, as s is a central node, according to the algorithm.

5 Experimental analysis

In this section, we present the experimental evaluation we performed in order to check the practical effectiveness of DLP. In particular, we combined DLP with DUAL, DUST and LFR and then we implemented, in C++, and tested both the the original algorithms and the new combinations, namely DUAL-DLP, DUST-DLP and LFR-DLP.

In what follows, we report the results of such experimental study. Our experiments have been performed on a workstation equipped with a Quad-core 3.60 GHz Intel Xeon X5687 processor, with 12MB of internal cache and 24 GB of main memory and consist of simulations within the well-known OMNeT++ 4.0p1 environment [20]. The software has been compiled with GNU g++ compiler 4.4.3 under Linux (Kernel 2.6.32).

OMNeT++ is an object-oriented modular discrete event network simulator, useful to model protocols, telecommunication networks, and other distributed systems. An OMNeT++ model consists of hierarchically nested modules, that communicate through message passing. In our model, we defined a basic module *node* to represent a node in the network. A node v has a communication *gate* with each node in $N(v)$. Each node can send messages to a destination node through a *channel* which is a module that connects gates of different nodes (both gate and channel are OMNeT++ predefined modules). In our model, a channel connects exactly two gates and represents an edge between two nodes. We associate two parameters per channel: a *weight* and a *delay*. The former represents the cost of the edge in the graph, and the latter simulates a finite but not null transmission time.

5.1 Executed tests

As input to the algorithms we used both real-world and artificial instances of the problem. In detail, we used real-world networks of the *CAIDA IPv4 topology dataset* [15] and random networks generated by the *Barabási-Albert* algorithm [2], subject to randomly generated sequences of edge update operations.

Concerning CAIDA instances, we parsed the files provided by CAIDA to obtain a weighted undirected graph G_{IP} where a node represents an IP address contained in the dataset (both source/destination hosts and intermediate hops), edges represent links among hops and weights are given by Round Trip Times. As the graph G_{IP} consists of almost 35000 nodes, we cannot use it for the experiments, as the amount of memory required to store the routing tables of all the nodes is $O(n^2 \cdot \mathit{maxdeg})$ for any implemented algorithm. Hence, we performed our tests on connected subgraphs of G_{IP}, with a variable number of nodes and edges, induced by the settled nodes of a breadth first search starting from a node taken at random. We generated a set of different tests, each test consists of a dynamic graph characterized by a subgraph of G_{IP} (we denoted each n nodes subgraph of G_{IP} with G_{IP-n}) and a set of k random edge updates, where k assumes values in $\{5, 10, \ldots, 200\}$. An edge update consists of multiplying the weight of a random selected edge by a percentage value randomly chosen in $[50\%, 150\%]$. For each test configuration (a dynamic graph with a fixed value of k) we performed 5 different experiments (for a total amount of 200 runs) and we report average values.

Concerning Barabási–Albert instances, we randomly generated a set of different tests, where a test consists of a dynamic graph characterized by a n nodes Barabási–Albert random graph, denoted as G_{BA-n} and a set of k random edge updates, where k assumes values in $\{5, 10, \ldots, 200\}$. Edge weights are non-negative real numbers randomly chosen in $[1, 10000]$. Edge updates are randomly chosen as in the CAIDA tests. For each test configuration (a dynamic graph with a fixed value of k) we performed 5 different experiments (for a total amount of 200 runs) and we report average values.

5.2 Analysis

We performed experiments on subgraphs of G_{IP} and on Barabási-Albert graphs having 1200, 5000 and 8000 nodes. The results of our experiments are quite similar on these different instances and then we report only the results on graphs with 8000 nodes. These results are shown in Figures 6, 7, and 8, concerning the number of messages sent, and in Table 1 concerning the space occupancy per node.

In detail, in Figure 6, 7, and 8 we report the number of messages sent by DUAL and DUAL-DLP, DUST and DUST-DLP, and LFR and LFR-DLP, respectively, on a

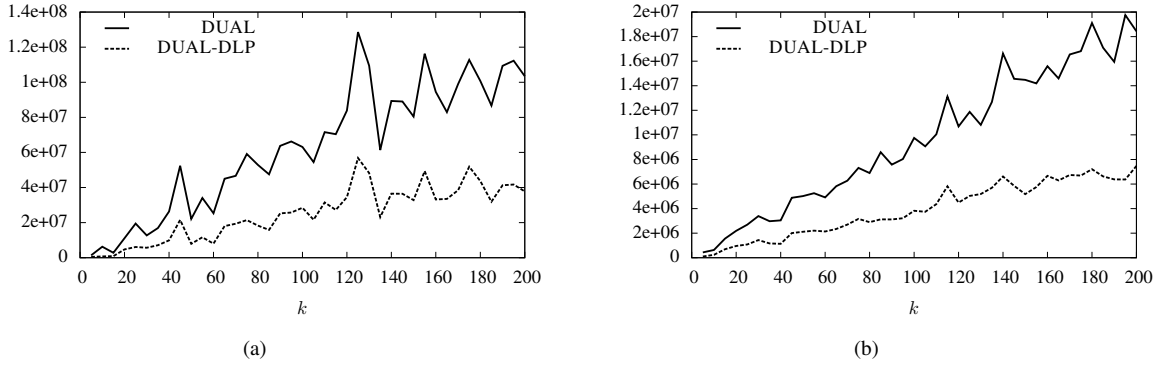


Figure 6: Number of messages sent by DUAL and DUAL-DLP on $G_{IP-8000}$ (a) and $G_{BA-8000}$ (b).

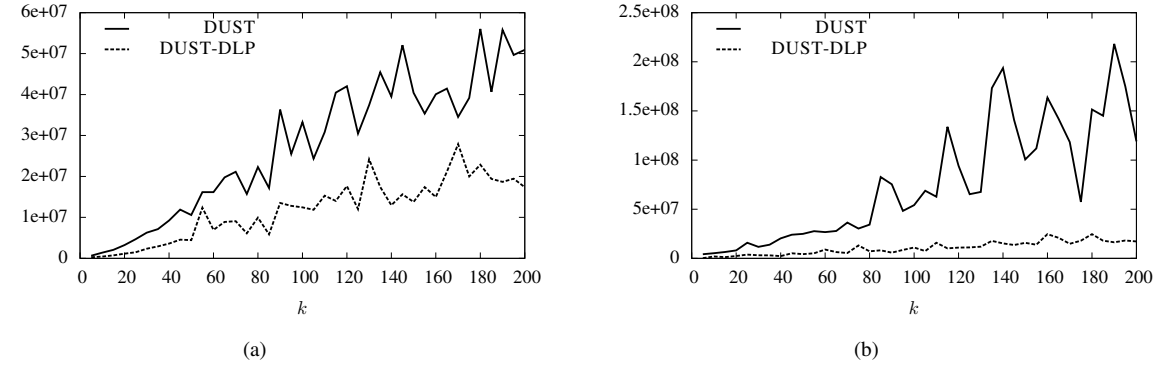


Figure 7: Number of messages sent by DUST and DUST-DLP on $G_{IP-8000}$ (a) and $G_{BA-8000}$ (b).

CAIDA graph having 8000 nodes and 11141 edges (a) and on a Barabási-Albert graphs having 8000 nodes and 12335 edges (b), when the number k of edge updates ranges from 5 to 200. These figures show in general that the combination of DLP with DUAL, DUST and LFR provides a significant improvement in the global number of messages sent by these algorithms.

Regarding $G_{IP-8000}$, we can observe what follows. In the tests of Figure 6(a), the ratio between the number of messages sent by DUAL-DLP and DUAL is within 0.12 and 0.46 which means that the number of messages sent by DUAL-DLP is between 12% and 46% that of DUAL. In the tests of Figure 7(a), the ratio between the number of messages sent by DUST-DLP and DUST is within 0.30 and 0.81. In the tests of Figure 8(a), the ratio between the number of messages sent by LFR-DLP and LFR is within 0.36 and 0.51.

Regarding $G_{BA-8000}$, we can observe what follows. In the tests of Figure 6(b), the ratio between the number of messages sent by DUAL-DLP and DUAL is within 0.23 and 0.48. In the tests of Figure 7(b), the ratio between the number of messages sent by DUST-DLP and DUST is within 0.04 and 0.43. In the tests of Figure 8(a), the ratio between the number of messages sent by LFR-DLP and LFR is within 0.38 and 0.47.

In summary, from our data follows that the algorithms integrating DLP send, on average, 0.42 (0.34, respectively) times the number of messages sent by the original algorithms on $G_{IP-8000}$ ($G_{BA-8000}$, respectively), which represents a substantial improvement in the practical applications where these algorithms are used.

To conclude our analysis, we have considered the space occupancy per node, which is summarized in Table 1, where we report the maximum and the average space occupancy per node of each algorithm, and the memory overhead required by the combination of that algorithm with DLP.

Note that DUAL requires a node v to store, for each destination, the estimated distance given by each of its neighbors and a set of variables used to guarantee loop-freedom, DUST only needs the estimated distance of v and the set `VIA`, for each destination and LFR requires to store, for each node v, the estimated distance of v and the feasible via to each source s, that is the node through which the distance to s is minimum, plus other variables needed to guarantee loop-freedom. Since in the sparse graphs we considered it is not common to have more than one via to a destination, the memory requirement of DUST is much smaller than that of DUAL, and smaller than that of LFR. Note also that the space occupancy of the data structure needed to implement DLP is not a function of the degree of the graph and is always bounded, in the worst case, by n. However, our experiments show that the use of DLP induces, in most of the cases, a clear improvement also in

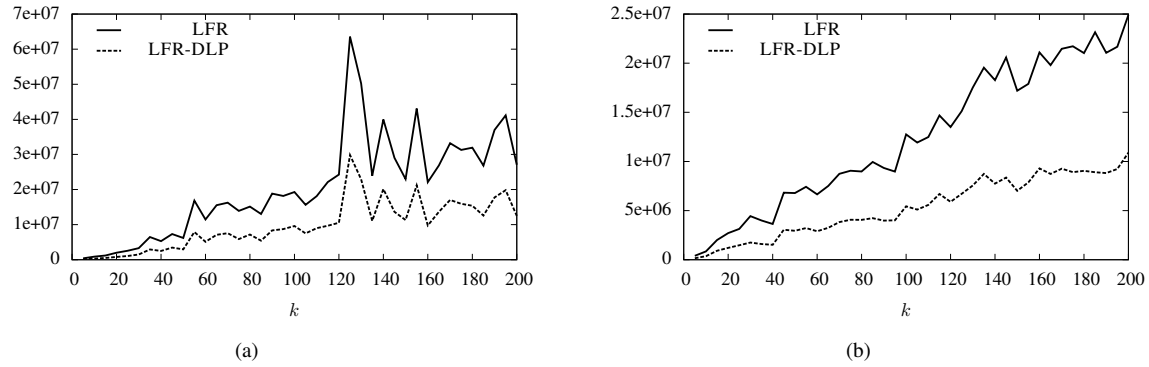


Figure 8: Number of messages sent by LFR and LFR-DLP on $G_{IP-8000}$ (a) and $G_{BA-8000}$ (b).

the maximum and in the average space requirements per node (see Table 1). This is due to the fact, that the overhead induced by the data structures needed to implement DLP, in most of the cases is overcome by the fact that DLP allows nodes to avoid to store some data concerning peripheral nodes, and the number of peripheral nodes in the networks used for the experiments is quite high. In particular, we can observe what follows. Concerning DUST, the combination with DLP determines an overhead which is always around 20% of the space occupancy of DUST, both in the maximum and in the average cases. This is due to the fact that DUST needs a really small amount of space and hence the data structures of DLP have a certain impact in the space occupancy of DUST. Concerning DUAL (LFR), we can observe that the use of DLP induces a gain in the maximum space occupancy per node which is around 38% (27%) on $G_{IP-8000}$ and around 45% (31%) on $G_{BA-8000}$. Notice that, the improvement is more evident in the case of DUAL, as its maximum space occupancy per node is by far higher than that of LFR. Concerning DUAL, this behavior is confirmed also in the average case, where the use of DLP induces a gain around 23% on $G_{IP-8000}$ and around 27% on $G_{BA-8000}$. On the contrary, our data show that the average space occupancy per node of LFR-DLP is slightly greater than that of LFR and that the use of DLP hence induces an overhead in the average space occupancy per node which is around 6% in $G_{IP-8000}$ and around 7% in $G_{BA-8000}$. This is due to the fact that the average space occupancy of LFR is quite low by itself and that, in this case, the space occupancy overhead needed to store the CT data structure is greater than the space occupancy reduction induced by the use of DLP.

6 Conclusions and future work

We have proposed a simple and practical technique, which can be combined with every distance vector routing algorithm based on shortest paths, allowing to reduce the total number of messages sent by that algorithm and the average space occupancy per node. We have combined the new technique with DUAL, DUST, and the recent LFR. We have given experimental evidence that these combinations lead to an important gain in terms of both the number of messages sent and the space occupancy per node.

Some research directions deserve further investigation: (i) to extend DLP in some way, for example considering nodes of small degree (greater than one) as peripherals; (ii) to know how DLP is scalable to bigger networks than those considered in this paper; (iii) to perform simulations on different power-law models, as for example the Generalized Linear Preference model (GLP) [4], which have particular properties that could impact on the performances of DLP.

Acknowledgments

Support for the IPv4 Routed/24 Topology Dataset is provided by National Science Foundation, US Department of Homeland Security, WIDE Project, Cisco Systems, and CAIDA.

References

[1] R. Albert and A.-L. Barabási. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex network. *Reviews of Modern Physics*, 74:47–97, 2002.

[3] D. Bertsekas and R. Gallager. *Data Networks*. Prentice Hall International, 1992.

[4] T. Bu and D. Towsley. On distinguishing between internet power law topology generators. In *Proc. of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 37–48. IEEE, 2002.

[5] S. Cicerone, G. D'Angelo, G. Di Stefano, and D. Frigioni. Partially dynamic efficient algorithms for distributed shortest paths. *Theoretical Comp. Science*, 411:1013–1037, 2010.

Graph	Algorithm	Max		Avg	
		Bytes	DLP **Overhead**	**Bytes**	DLP **Overhead**
$G_{IP-8000}$	DUAL	8 320 000	—	311 410	—
	DUAL-DLP	5 161 984	-37.96%	240 754	-22.69%
	LFR	757 268	—	192 984	—
	LFR-DLP	554 987	-26.71%	204 724	6.08%
	DUST	64 088	—	64 000	—
	DUST-DLP	76 376	19.17%	76 288	19.20%
$G_{BA-8000}$	DUAL	20 800 000	—	323 350	—
	DUAL-DLP	11 483 200	-44.80%	240 550	-25.61%
	LFR	685 498	—	193 267	—
	LFR-DLP	475 466	-30.64%	207 076	7.15%
	DUST	64 056	—	64 000	—
	DUST-DLP	76 344	19.18%	76 288	19.20%

Table 1: Space complexity - Results of a dynamic execution with $k = 200$ over $G_{IP-8000}$ and $G_{BA-8000}$.

[6] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and V. Maurizio. A new fully dynamic algorithm for distributed shortest paths and its experimental evaluation. In *Proc. 9th Symposium on Experimental Algorithms (SEA)*, volume 6049 of *Lecture Notes in Computer Science*, pages 59–70, 2010.

[7] S. Cicerone, G. D'Angelo, G. Di Stefano, D. Frigioni, and V. Maurizio. Engineering a new algorithm for distributed shortest paths on dynamic networks. *Algorithmica*, 66(1):51–86, 2013.

[8] S. Cicerone, G. Di Stefano, D. Frigioni, and U. Nanni. A fully dynamic algorithm for distributed shortest paths. *Theoretical Computer Science*, 297(1-3):83–102, 2003.

[9] G. D'Angelo, M. D'Emidio, D. Frigioni, and V. Maurizio. A speed-up technique for distributed shortest paths computation. In *Proc. 11th International Conference on Computational Science and Its Applications (ICCSA)*, volume 6783 of *Lecture Notes in Computer Science*, pages 578–593, 2011.

[10] G. D'Angelo, M. D'Emidio, D. Frigioni, and V. Maurizio. Engineering a new loop-free shortest paths routing algorithm. In *Proc. 11th Symposium on Experimental Algorithms (SEA)*, volume 7276 of *Lecture Notes in Computer Science*, pages 123–134, 2012.

[11] K. Elmeleegy, A. L. Cox, and T. S. E. Ng. On count-to-infinity induced forwarding loops in ethernet networks. In *Proc. of the 25th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1–13, 2006.

[12] D. Frigioni, A. Marchetti-Spaccamela, and U. Nanni. Fully dynamic algorithms for maintaining shortest paths trees. *Journal of Algorithms*, 34(2):251–281, 2000.

[13] J. J. Garcia-Lunes-Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Trans. on Networking*, 1(1):130–141, 1993.

[14] P. A. Humblet. Another adaptive distributed shortest path algorithm. *IEEE Trans. on Communications*, 39(6):995–1002, April 1991.

[15] Y. Hyun, B. Huffaker, D. Andersen, E. Aben, C. Shannon, M. Luckie, and KC Claffy. The CAIDA IPv4 routed/24 topology dataset. `http://www.caida.org/data/active/ipv4_routed_24_topology_dataset.xml`.

[16] G. F. Italiano. Distributed algorithms for updating shortest paths. In *Proc. of the International Workshop on Distributed Algorithms*, volume 579 of *Lecture Notes in Computer Science*, pages 200–211, 1991.

[17] J. McQuillan. Adaptive routing algorithms for distributed computer networks. Technical Report BBN Rep. 2831, Cambridge, MA, 1974.

[18] J. T. Moy. *OSPF: Anatomy of an Internet routing protocol*. Addison-Wesley, 1998.

[19] A. Myers, E. Ng, and H. Zhang. Rethinking the service model: Scaling ethernet to a million nodes. In *ACM SIGCOMM HotNets*. ACM Press, 2004.

[20] OMNeT++. Discrete event simulation environment. `http://www.omnetpp.org`.

[21] A. Orda and R. Rom. Distributed shortest-path and minimum-delay protocols in networks with time-dependent edge-length. *Distr. Computing*, 10:49–62, 1996.

[22] K. V. S. Ramarao and S. Venkatesan. On finding and updating shortest paths distributively. *J. of Algorithms*, 13:235–257, 1992.

[23] S. Ray, R. Guérin, K. W. Kwong, and R. Sofia. Always acyclic distributed path computation. *IEEE/ACM Trans. on Networking*, 18(1):307–319, 2010.

[24] S. R. Soloway and P. A. Humblet. Distributed network protocols for changing topologies: A counterexample. *IEEE Trans. on Communications*, 39(3):360–361, March 1991.

[25] N. Yao, E. Gao, Y. Qin, and H. Zhang. Rd: Reducing message overhead in DUAL. In *Proc. of the 1st International Conference on Network Infrastructure and Digital Content (IC-NIDC09)*, pages 270–274. IEEE Press, 2009.

[26] C. Zhao, Y. Liu, and K. Liu. A more efficient diffusing update algorithm for loop-free routing. In *Proc. of the 5th International Conference on Wireless Communications, Networking and Mobile Computing (WiCom09)*, pages 1–4. IEEE Press, 2009.

Random Search Algorithm for the p-Median Problem

Alexander N. Antamoshkin and Lev A. Kazakovtsev
Siberian State Aerospace University
prosp. Krasnoyarskiy rabochiy, 31, Krasnoyarsk 660014, Russian Federation
E-mail: levk@ieee.org

Keywords: continuous location problem, Weber problem, random search, genetic algorithms, discrete optimization

Received: November 5, 2012

Authors investigate the p-median location problem on networks and propose a heuristic algorithm which is based on the probability changing method (a special case of the genetic algorithm) for an approximate solution to the problem. The ideas of the algorithm are proposed under the assumption that, in the large-scale networks with comparatively small edge lengths, the p-median problem has features similar to the Weber problem. The efficiency of the proposed algorithm and its combinations with the known algorithms were proved by experiments.

Povzetek: Avtorji predlagajo nov hevristični algoritem za iskanje p-mediane v omrežjih.

1 Introduction

The aim of the location problem [13] is to determine the location of one or more new facilities in a set of possible locations (discrete or continuous). The main parameters of such problems are the coordinates of the facilities and distances between them [37, 14, 15]. Examples of the location problems include the location of warehouses [15], computer and communication networks, base stations of wireless networks [30] etc. They are also useful in the approximation theory, statistical estimation problem [28], signal and image processing and other engineering applications.

The Fermat-Weber problem (Weber problem) [35, 37] is the problem of searching for such a point that a sum of weighted distances from this point to the given points (demand points) is minimal. In the location theory, several generalizations of these problems are known [11]. The first one is the multi-Weber problem where the aim is to find optimal locations of p new facilities:

$$F(X_1, ..., X_p) = \sum_{i=1}^{N} w_i \min_{j \in \{\overline{1,p}\}} L(X_j, A_i) \to min. \quad (1)$$

Here, $\{A_i | i = \overline{1, N}\}$ is a set of the demand points, $\{X_j | j = \overline{1, p}\}$ is a set of new placed facilities, w_i is a weight coefficient of the ith demand point, $L()$ is a distance function.

One of the problems of the discrete location theory, a p-median problem, can also be considered as a generalization of the Weber problem [17, 23]. The medians are searched for in a finite set of graph vertices. Generally this problem is $\mathcal{NP}$-hard [24]. The polynomial-time algorithm is developed for trees only [20]. A procedure for network flow searching (an algorithm for the p-median problem solving) is adapted for location problems in a continuous space with the rectangular metric [10].

Despite the complexity of the problem, various heuristic algorithms could give good results for most problems in reasonable time. One of the simplest but efficient heuristics for the p-median problem is local search [31, 32]. Rabbani [29] proposes an algorithm based on new graph theory for small size problems. Using Lagrangian relaxation allows an approximate solving of huge-scale problems [5], up to 90000 vertices in a network. However, "good" solutions [6] were achieved by the analogous technique for problems with $n = 3795$ which were also considered as large-scale problems.

Hosage and Goodchild [19] proposed the first genetic algorithm for the p-median problem. In [9], authors propose a genetic algorithm providing rather precise results but its convergence is slow. In [1], authors propose a quick and precise genetic algorithm for the p-median problem. However, solving large-scale problems still takes too much time.

In [14], a continuous problem with an arbitrary l_p metric is solved. In [24] and [25], authors prove that the unconstrained Weber problems with Euclidean or rectangular metric are $\mathcal{NP}$-complete.

For the continuous location problems with Euclidean (l_2), rectangular (l_1) and Chebyshev (l_∞) metrics, the algorithms based on the Weiszfeld procedure are proposed [36]. However, a solution of the same problems with restricted zones [39], barriers [8] or modified distance functions is not trivial. In practically important problems, models based on the Euclidean or rectangular metric are usually rough approximations [27, 38] since they do not take into account characteristics of the space and transportation means, in particular, the presence and quality of roads, barriers, relief etc. Sometimes, the distance function is given algorithmically as a solution of another optimization problem [38]. Thus, if the problem is formulated as a Weber

problem, its discretization should be often useful [21].

We can always consider practical problems as discrete. Any scheme (or map) has finite resolution, digital copy of any map is always discrete. The implementation of the obtained solution of a problem is also performed by the tools with finite precision. However, such a discrete problem is a large-scale problem and any algorithms which guarantee an exact solution in polynomial time do not exist (polynomial time algorithm exists for a discretized generalized Weber problem with rectangular metric only [10]).

The unconstrained location problems with mixed coordinate systems (discrete and continuous) are considered in [33, 16].

Heuristic random search algorithms do not guarantee an exact solution. However, they are statistically optimal, i.e., the percentage of the "near optimal" solutions increases with growth of the problems dimension [3]. In case of discrete location problems, genetic algorithms [30, 26], greedy search [26, 18] and other methods are implemented.

The probability changing method initially proposed for unconstrained optimization is a random search method described by the pseudo-code below.

Algorithm 1. *Basic probability changing method*

1: Set $k = 0$; set the initial values of the components of the probability vector $P_0 = \{p_{0,1}, p_{0,2}, ..., p_{0,N}\}$ (here, the value of each element at the kth step is the probability (expectation) of generating a vector X with the corresponging element equal to 1: $p_{k,j} = \mu\{x_j = 1\}$);

2: In accordance with the distribution given by the elements of the vector P, generate a set of N_{POP} vectors $X_{k,i}$, $i \in \{\overline{1, N_{POP}}\}$;

3: Calculate the value of the objective function $F(X_{k,i})$ $\forall i \in \{\overline{1, N_{POP}}\}$.

4: Select some vectors $X_{k,i}$ (for example, a vector with the best and the worst value of the objective function);

5: Based on the results of the Step 4, modify the probability vector P;

6: $k = k + 1$; if $k < N_{STEPS}$ then goto 2 (other stop conditions are also possible);

7: STOP.

This simple method is a special variant of the genetic algorithm [12]. The modifications of this algorithm for the constrained optimization problems proposed in [22] can solve pseudo-Boolean problems (multi-knapsack problem, travelling salesman problem) with dimensions up to millions of Boolean variables.

2 Problem statement, known algorithms

Let $G = (V, E)$ be an undirected adjacent graph (a network), $V = \{v_1, ..., v_n\}$ be a set of its vertices (Fig. 1), $E = \{e_i | i = \overline{1, m}\}$ be a set of its edges, $e_i = (v_j, v_k)$, $j \in \{\overline{1, n}\}$, $k \in \{\overline{1, n}\}$, $i \in \{\overline{1, m}\}$ without loops $(e_i \neq (v_j, v_j)\ \forall i = \overline{1, m}, j = \overline{1, n})$. For each edge e_i, its length l_i is defined, $l_i \geq 0 \forall i = \overline{1, m}$. For an edge $e_i = (v_j, v_k)$, let us denote $l_{j,k} = l_i$. Weight $w_j \geq 0$ is defined for each vertex v_j. For each pair of the vertices (v_j, v_k), a distance function $L(j, k)$ is defined as the length of the shordest path from v_i to v_j.

$$L(j,k) = \sum_{q \in P^*_{j,k}} l_q = \min_{P \in P_{j,k}} \sum_{q \in P} l_q \tag{2}$$

Here, $P^*_{j,k}$ is a set of the edges of the shortest path between v_j and v_k, $P_{j,k}$ is a set of all possible paths between these vertices. We can formulate the the p-median problem as

$$\begin{aligned} \arg \quad & \min_{m_1,...,m_p \in \{\overline{1,n}\}} f(m_1, ..., m_p) \\ & = \arg \min_{m_1,...m_p \in \overline{1,n}} \sum_{i=1}^{n} w_i \min_{i=\overline{1,p}} L(m_j, i), \\ & p < n. \end{aligned} \tag{3}$$

The aim of this problem is to select p vertices so that the sum of weighted distances from each of the vertices of the graph to the nearest selected vertex is minimal.

Let

$$S_i = \{j | \exists e_j = (v_i, v_k), j \in \{\overline{1, m}\}, k \in \{\overline{1, n}\}\}$$

be a set of the edges of the vertices incident to the ith vertex;

$$C_i = \{k | \exists e_j = (c_i, v_k), j \in \{\overline{1, m}\}, k \in \{\overline{1, n}\}\}$$

be a set of the indexes of the vertices adjacent to the ith vertex;

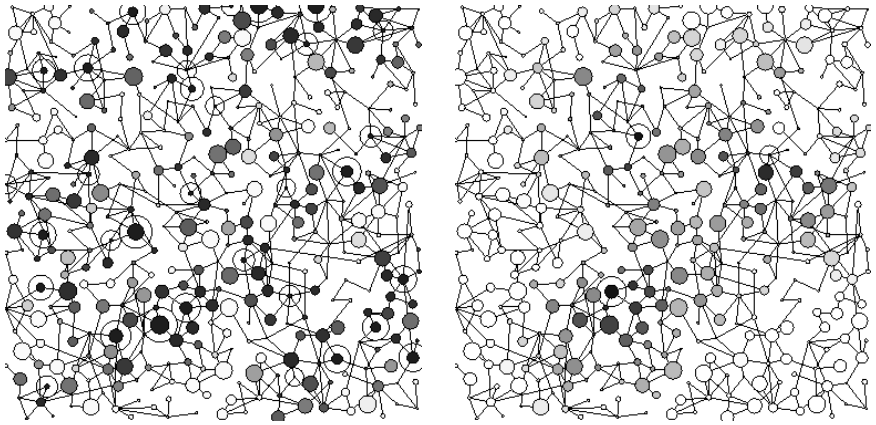
$$l^*_i = \min_{j \in \{\overline{1,p}\}} L(m_j, i)$$

be the length of the shortest path from the ith vertex to the nearest of p vertices $m_1, ..., m_p$.

For calculating the value of the objective function $f(m_1, ..., m_p)$, we use the algorithm below.

Algorithm 2. *p-median objective function calculation*

Require: indexes $m_1, ..., m_p$ of p selected vertices. **1:** $l^*_i = +\infty \forall i = \overline{1, n}$;
2: $l^*_{m_i} = 0 \forall i = \overline{1, p}$;
3: $V^* = \{m_1, ..., m_p\}$; $V^{**} = V^*$;
4: while $|V^{**}| \neq 0$ do
4.1: $V' = \emptyset$;
4.2: for $i \in V^{**}$ do
4.2.1: for $j \in C_i$ do
4.2.1.1: if $v_j \notin V^*$ then $V' = V' \cup \{j\}$; $l^*_j = l^*_i + l_{i,j}$;
4.2.1.2: else if $l^*_j = l^*_i + l_{i,j}$ then $l^*_j = l^*_l + l_{i,j}$;
4.2.1.3: next 4.2.1;
4.2.2: next 4.2;
4.3: $V^{**} = V'$; $V^* = V^* \cup V'$;
4.4: next 4;
5: return $f(m_1, ..., m_p) = \sum_{i=1}^{n} w_i l^*_i$.



a) solution for $p = 32$ b) solution for $p = 3$

Figure 1: Scheme of a problem and its solutions, $n = 400$

For comparison, we used the local search algorithm [32] with a random order of vertices evaluation (Algorithm 3) as one of the simplest but efficient algorithms.

Algorithm 3. *Local search*

Require: array of indexes $\mathcal{M} = \{m_1, ..., m_p\}$ of the vertices (initial solution), value of the objective function $f^* = f(m_1, ..., m_p)$.
1: shuffle elements of $\mathcal{M}$; $r = 0$;
2: for each element m of the array $\mathcal{M}$ do
2.1: for each vertex m^* which is adjacent to m do
2.1.1: $f^{**} = f(m_1, ..., m^*, ..., m_p)$ (here, the vertex m is replaced by m^*);
2.1.2: if $f^{**} < f^*$ then replace m by m^* in $\mathcal{M}$; $f^* = f^{**}$; $r = 1$;
2.1.3: next 2.1;
2.2: next 2;
3: if $r = 1$ then goto 1;
5: return new solution $(m_1, ..., m_p)$.

The genetic algorithm with greedy heuristic proposed in [1] includes a special crossover procedure (Algorithm 4). The "chromosomes" of this algorithm are sets of the vertices (solutions of the problem).

Algorithm 4. *Crossover procedure for the GA with greedy heuristic*

Require: sets of indexes $\mathcal{M}_1 = \{m_{1,1}, ..., m_{1,p}\}$, $\mathcal{M}_2 = \{m_{2,1}, ..., m_{2,p}\}$ of the vertices ("chromosomes").
1: $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2$; $p_{\mathcal{M}} = |\mathcal{M}|$;
2: while $p_{\mathcal{M}} > p$ do
2.1: $f* = +\infty$;
2.2: for each vertex m^* in $\mathcal{M}$ do
2.2.1: $\mathcal{M}^* = \mathcal{M} \setminus \{m^*\}$;
2.2.2: $f** = f(\mathcal{M}*)$;
2.2.2: if $f^{**} < f^*$ then $m^{**} = m^*$;
2.1.3: next 2.2;
2.3: $\mathcal{M} = \mathcal{M} \setminus \{m^{**}\}$; $p_{\mathcal{M}} = p_{\mathcal{M}} - 1$;
2.3: next 2;
5: return new solution ("chromosome") $\mathcal{M}$.

This method uses an original procedure of the initial population generation [1]. It does not use any mutation procedure.

The probability changing method is a pseudo-Boolean optimization method, the objective function must be a function of Boolean variables.

Let us introduce new variables $x_1, ..., x_n$:

$$x_i = \begin{cases} 1, & i \in \{m_1, ..., m_p\} \\ 0, & i \notin \{m_1, ..., m_p\} \end{cases}. \tag{4}$$

In this case, our problem has a constraint

$$\sum_{i=1}^{n} x_i = p. \tag{5}$$

The transformation of the problem back into the problem with integer variables is performed as follows:

$$\{m_1, ..., m_p\} = \{i | x_i = 1, i = \overline{1, n}\}. \tag{6}$$

The problem with the pseudo-Boolean objective function is stated as follows:

$$\begin{aligned} f_b(x_1, ..., x_n) &= f(\{j | x_j = 1, j = \overline{1, n}\}) \\ &= \sum_{i=1}^{n} w_i \min_{j | x_j = 1, j = \overline{1,n}} L(i, j) \end{aligned} \tag{7}$$

with the constraint (5).

In this form, our problem can be solved using many methods [3, 4, 2, 22] including the probability changing method.

3 An algorithm based on the probability changing method

Continuous location problems such as multi-Weber problem with Euclidean, rectangular or Chebyshev meric are

amenable to analysis and analytical solution methods. Weiszfeld [36] procedure and Trubin's procedure for rectangular metric [34] are based on the assumption that the partial derivatives of the objective function are defined, i.e., a small change of the location of any point in a feasible solution leads to some small change in the value of the objective function. If $X = (x_1, x_2)$ is a solution of some $2D$ unconstrained location problem then

$$\frac{\Delta f(X)}{\Delta X} = \frac{\Delta f(x_1, x_2)}{\Delta\sqrt{\Delta x_1 + \Delta x_2}} < const.$$

Let L_{max} be the maximum distance between 2 vertices:

$$L_{max} = \max_{i,j\in\{\overline{1,n}\}} L(i,j), \tag{8}$$

l_{avg} be the average length of the edge:

$$l_{avg} = \sum_{j=1}^{m} l_j/m. \tag{9}$$

Our algorithm is based on 2 hypotheses:

Hypothesis 1. *If l_{avg}/L_{max} is small ($l_{avg}/L_{max} \to 0$) then the character of the p-median problem shows features of the continuous problem. In particular, if $\{m_1, ..., m_p\}$ is a solution of the p-median problem then, having replaced any m_ith vertex of this solution to any jth point such that $L(j, m_i)$ is small, the change in the objective function value is also small:*

$$\exists l_\Delta, \Delta_{Fmax} > 0 : (j, m_i) < l_\Delta \Rightarrow$$
$$|f(m_1, ..., m_i, ..., m_p) - f(m_1, ..., j, ..., m_p)|$$
$$< \Delta_{Fmax}$$

Hypothesis 2. *If the solution $\{m_1, ..., m_i, ..., m_j, ..., m_p\}$ contains 2 vertices v_i and v_j such that $L(m_i, m_j)$ is small then there is high probability (expectation) that for the solution $\{m_1, ..., m_i, ..., k, ..., m_p\}$ with a vertex v_j replaced by another arbitrary chosen vertex v_k, the value of the objective function is "better" than for the original solution:*

$$\exists l_{min} :$$
$$L(m_i, m_j) < l_{min} \wedge L(m_i, k) > l_{min} \Rightarrow$$
$$\Rightarrow \mu \left\{ \begin{array}{l} f(m_1, ..., m_i, ..., m_j, ..., m_p) \\ \geq f(m_1, ..., m_i, ..., k, ..., m_p) \end{array} \right\} > 0.5$$

and

$$\lim_{l_{min}\to 0} \mu \left\{ \begin{array}{l} f(m_1, ...m_i, ...m_j, ...m_p) \\ \geq f(m_1, ...m_i, ...k, ...m_p) \end{array} \right\} \to 1.$$

Let us prove the consistency of the hypotheses for the special cases.

Lemma 1. *Let $L(m_i, j) = 0$. Then*

$$f(m_1, ..., m_i, ..., m_p) = f(m_1, ..., j, ..., m_p).$$

Proof. Let us choose arbitrarily the i^*th vertex, $i^* \in \{\overline{1,n}\}$.

If $i^* \in \{m_1, ..., m_i, ..., m_p\}$ then, obviously, $\min_{i''\in\{m_1,...m_i,...m_p\}} L(i^*, i'') = 0$.

$$\min_{i''\in\{m_1,...,j,...,m_p\}} L(i^*, i'')$$
$$= \min \left\{ \min_{i''\in\{m_1,...m_i,...m_p\}} L(i^*, i'') + L(m_i, j); \right.$$
$$\left. L(i^*, j) \right\}$$
$$= \min\{0 + 0; L(i^*, j)\} = 0.$$

If $i^* \notin \{m_1, ..., m_i, ..., m_p\}$, let us introduce the notation:
$P_{i^*,j}$ is a set of all possible paths from the i^*th vertex to the jth one,
P_{i^*,m_i} is a set of all possible paths from the i^*th vertex to the m_ith,
$P_{i^*(m_i)j}$ is a set of all possible paths from the i^*th vertex to the jth one through the m_ith vertex,
$P_{i^*(\overline{m_i})j}$ is a set of all possible paths from the i^*th vertex to the jth one which do not include the m_ith vertex.

$$L(i^*, j) = \min_{P\in P_{i^*,j}} \sum_{e_k\in P} l_k$$
$$= \min\{\min_{P\in P_{i^*(m_i)j}} \sum_{e_k\in P} l_k; \min_{P\in P_{i^*(\overline{m_i})j}} \sum_{e_k inP} l_k\}$$
$$= \min\{ \min_{P\in P_{i^*,m_i}} \sum_{e_k\in P} l_k + \min_{P\in P_{m_i,j}} \sum_{e_k\in P} l_k;$$
$$\min_{P\in P_{i^*(\overline{m_i})j}} \sum_{e_k\in P} l_k \}$$
$$= \min\{\min_{P\in P_{i^*,m_i}} \sum_{e_k\in P} l_k + 0; \min_{P\in P_{i^*(\overline{m_i})j}} \sum_{e_k inP} l_k\}$$
$$= \min\{L(i^*, m_i); \min_{P\in P_{i^*(\overline{m_i})j}} \sum_{e_k inP} l_k\}$$
$$\leq L(i^*, m_i).$$

$$L(i^*, m_i) = \min_{P\in P_{i^*,m_i}} \sum_{e_k\in P} l_k$$
$$= \min\{\min_{P\in P_{i^*(j)m_i}} \sum_{e_k\in P} l_k; \min_{P\in P_{i^*(\overline{j})m_i}} \sum_{e_k inP} l_k\}$$
$$= \min\{ \min_{P\in P_{i^*,j}} \sum_{e_k\in P} l_k + \min_{P\in P_{j,m_i}} \sum_{e_k\in P} l_k;$$
$$\min_{P\in P_{i^*(\overline{j})m_i}} \sum_{e_k\in P} l_k \}$$
$$= \min\{\min_{P\in P_{i^*,j}} \sum_{e_k\in P} l_k + 0; \min_{P\in P_{i^*(\overline{j})m_i}} \sum_{e_k inP} l_k\}$$
$$= \min\{L(i^*, j); \min_{P\in P_{i^*(\overline{j})m_i}} \sum_{e_k inP} l_k\} \leq L(i^*, j).$$

$$\begin{array}{lll} L(i^*, m_i) & \leq L(i^*, j) \wedge L(i^*, j) \leq L(i^*, m_i) & \\ \Rightarrow & L(i^*, m_i) = L(i^*, j). & (10) \end{array}$$

□

Lemma 2. *Let $\{m_1, ..., m_i, ..., m_j, ..., m_p\}$ be a solution of the p-median location problem on a network with n vertices, $w_i > 0 \forall i = \overline{1,n}$, $L(m_i, m_j) = 0$ and $\exists k \in \{\overline{1,n}\}$: $L(m_q, k) > 0 \forall q = \overline{1,p}$.*

Then

$$f \quad (m_1, ..., m_i, ..., m_j, ..., m_p) > f(m_1, ..., m_i, ..., k, ..., m_p). \qquad (11)$$

Proof. Let us introduce the notation

$$M_0 = \{m_1, ..., m_i, ..., m_j, ..., m_p\};$$

$$M_1 = \{m_1, ..., m_i, ..., k, ..., m_p\}.$$

Let us define a function

$$f_m(i', S) = \min_{i'' \in S} L(i'', i').$$

Its value for the sets denoted above is

$$f_m \quad (i', M_0) = \min\{ f_m(i', \{m_i\}); f_m(i', \{m_j\}); f_m(i', M_0 \setminus \{m_i\}) \} \ \forall i' \in \{\overline{1,n}\}.$$

Taking into account $L(m_i, m_j) = 0$, from Lemma 1, for the set of vertices of the solution

$$\begin{aligned} f_m \quad & (i', \{m_1, ..., m_i, ..., m_j, ..., m_p\}) \\ = \quad & f_m(i', \{m_1, ..., m_i, ..., m_i, ..., m_p\}) \\ = \quad & f_m(i', M_0) \ \ \forall i' \in \{\overline{1,n}\}. \end{aligned}$$

Further,

$$\begin{aligned} & f_m(i', M_1) \\ & = \min \quad \{ f_m(i', \{m_i\}); \ f_m(i', \{k\}); \\ & \qquad f_m(i', M_1 \setminus (\{m_i\} \cup \{k\})) \} \\ & \qquad \forall i' \in \{\overline{1,n}\}; \end{aligned}$$

$$\begin{aligned} f_m \quad & (i', M_0) \\ = \quad & \min\{f_m(i', \{m_i\}); f_m(i', M_0 \setminus \{m_i\})\} \\ = \quad & \min\{f_m(i', \{m_i\}); f_m((M_1 \setminus \{k\}) \setminus \{m_i\})\} \\ = \quad & \min\{f_m(i', \{m_i\}); f_m(M_1 \setminus (\{k\} \cup \{m_i\}))\} \\ \geq \quad & \min\{ f_m(i', \{m_i\}); \ f_m(i', \{k\}); \\ & \qquad f_m(i', M_1 \setminus (\{m_i\} \cup \{k\})) \ \} \\ = \quad & f_m(i', M_1) \ \ \forall i' \in \{\overline{1,n}\}; \end{aligned}$$

Thus,

$$\begin{aligned} f \quad & (m_1, ..., m_i, ..., k, ..., m_p) \\ = \quad & \sum_{i'=1}^{n} f_m(i', M_1) \leq \sum_{i'=1}^{n} f_m(i', M_0) \\ = \quad & f(m_1, ..., m_i, ..., m_j, ..., m_p). \end{aligned}$$

□

In our version of Algorithm 1, steps 2 and 5 are modified. At Step 2 (generation of the samples of vectors X), the constraint (5) must be taken into consideration. The solutions generated by algorithm below are always feasible.

Algorithm 5. *Step 2 of Algorithm 1*

Require: Probability vector $P = (p_1, ..., p_n)$.
1: $\chi = \emptyset$;
2: for each $i \in \{\overline{1,p}\}$ do
2.1: $r = Random() \cdot \sum_{j=1}^{n} p_i$; $S = 0$;
2.2: for each $j \in \{\overline{1,n}\}$ do
2.2.1: $S = S + p_i$;
2.2.2: if $S \geq r$ then $\chi = \chi \cup \{j\}$; goto 2.3;
2.2.3: next 2.2;
2.3: next 2;
3: return χ.

The result of this algorithm is a set χ. The corresponding vector X of boolean variables can be calculated in accordance with (4). Here, $Random()$ is a generator of the random value with continuous uniform distribution ($Random() \sim U[0, 1)$).

Let L_0 be the maximum distance considered as "small" in terms of Hypothesis 1 and Hypothesis 2. In accordance with Hypothesis 2,

$$\begin{aligned} \mu \quad & \{\exists m_i, m_j \in \chi : L(m_i, m_j) < L_0\} \\ < \quad & \mu\{L(m_i, m_j) \geq L_0 \forall m_i, m_j \in \chi\}; \end{aligned}$$

$$\lim_{L^* \to 0} \mu\{\exists m_i, m_j \in \chi : L(m_i, m_j) < L^*\} = 0.$$

Let us modify Algorithm 5.

Algorithm 6. *Step 2 of Algorithm 1, v. 2*

Require: Probability vector $P = (p_1, ..., p_n)$.
1: $\chi = \emptyset$;
2: for each $i \in \{\overline{1,p}\}$ do
2.1: $P^* = P$;
2.2: $r = Random() \cdot \sum_{j=1}^{n} p_i^*$; $S = 0$;
2.3: for each $j \in \{\overline{1,n}\}$ do
2.3.1: $S = S + p_i^*$;
2.3.2: if $S \geq r$ then $\chi = \chi \cup \{j\}$; $j' = j$; goto 2.3;
2.3.3: next 2.3;
2.4: for each $j \in \{k | k \in \{\overline{1,n}\} \wedge L(j', k) < L_0\}$ do: $p_j^* = p_j^* \cdot L(j, j') \cdot L_0$; next 2.4;
2.5: next 2;
3: return χ.

Step 5 of he Algorithm 1 (adaptation of the probability vector P) in its kth iteration must be performed in accordance with Hypothesis 2. We use the multiplicative adaptation.

$$P_{k,i} = p_{(k-1),i} \cdot d_{k,i}^b / d_{k,i}^w, \qquad (12)$$

$$d^b_{k,i} = \begin{cases} 1 + \frac{L_0}{1+L(i^b(i),i)}, & L(i,i^b(i)) < L_0 \\ 1, & L(i,i^b(i)) \geq L_0 \end{cases}, \quad (13)$$

$$d^w_{k,i} = \begin{cases} 1 + \frac{L_0}{1+L(i^w(i),i)}, & L(i,i^w(i)) < L_0 \\ 1, & L(i,i^w(i)) \geq L_0 \end{cases}. \quad (14)$$

Here,

$$i^b(i) = \arg_{i'} \min_{i' \in \chi^b} L(i,i'), \quad (15)$$

$$i^w(i) = \arg_{i'} \min_{i' \in \chi^w} L(i,i'), \quad (16)$$

χ^b and χ^w are the best and the worst samples of the sets of vertex indexes χ generated by Algorithm 2 or Algorithm 5. In the simplest case,

$$\chi^b = \arg_{\chi'} \min_{\chi' \in \mathcal{X}} f(\chi'), \quad (17)$$

$$\chi^b = \arg_{\chi'} \max_{\chi' \in \mathcal{X}_k} f(\chi'). \quad (18)$$

Here, $\mathcal{X}_k$ is a set of all samples of the vector χ at the kth iteration of Algorithm 1.

Note that the discretized Weber problem described in [21] is a special case of the p-median problem considered in this paper.

4 First results, adjusting parameters

For testing purposes, we used the p-median problems generated automatically by the special Algorithm 7.

Algorithm 7. *Sample problem generating*

Require: n.

1: for i in $\{\overline{1,n}\}$ do

1.1: $c^i_x = Random() \cdot 500$; $c^i_y = Random() \cdot 500$; $w_i = 1 + 9Random()$;

1.2: if $\exists j \in \{\overline{j, n-1}\}$: $\sqrt{(c^i_x + c^j_x)^2 + (c^i_y + c^j_y)^2} < 10$ then goto 1.1;

1.3: next 1;

2 Fill the adjacency matrix A with the zero values; $E = \emptyset$;

3: for i in $\{\overline{1,n}\}$ do

3.1:

$$D_i = \begin{cases} 1, & i \in \{\overline{[0.6n+1], n}\}, \\ 2, & i \in \{\overline{[0.4n]+1, [0.6n]}\}, \\ 3, & i \in \{\overline{[0.25n]+1, [0.4n]}\}, \\ 4 + [Random() \cdot 4], & i \leq [0.25n]. \end{cases}$$

3.2: for d in $\{\overline{1, \sum_{j=1}^{n} A_{i,j}}\}$;

3.2.1:

$j = \arg\min_{j \in \{\overline{1,n}\}, A_{i,j}=0} \sqrt{(c^i_x - c^j_x)^2 + c^i_y - c^j_y)^2}$;

$A_{i,j} = 1$; $A_{j,i} = 1$; $E = E \cup \{(i,j)\}$; $l_{i,j} = \sqrt{(c^i_x - c^j_x)^2 + c^i_y - c^j_y)^2}$;

3.2.2: next 3.2;

4: return adjacency matrix A, edges set E, edge lengths $\{l_{i,j}\} \forall (i,j) \in E$ and weights $w_i \forall i \in \{\overline{1,n}\}$.

Scheme of of such problem example is shown in Fig. 1. The lengths of the edges are proportional to ones shown in the scheme, the diameters of the vertices show their weights. In addition, in Fig. 1, the solutions calculated by our algorithm for $p = 32$ and $p = 3$ are shown. The vertices selected as the solution are shown in a circle. For each of the vertices, the color of the vertex shows the distance to the nearest selected vertex. The nearest vertices are shown in dark, the farthest are shown in white.

For our experiments, we used a computer Depo X8Sti (6-core CPU Xeon X5650 2.67 GHz, 12Gb RAM), hyperthreading disabled and ifort compiler with full optimization and implicit parallelism (option -O3). Comparison of the results reached with this hardware configuration with the results of the small system with 2-core Atom CPU N2701.6GHz, 1Gb RAM are shown in Fig. 2 (a combined method "probability changing+GA" is explained in Section 6).

Fig. 3 illustrates change in the probability vector for $p = 3$. The vertices with high value of the expectation of being included in the generated solutions are shown in white, the vertices with the smallest value of the expectation are shown in black.

The diameter L_0 of the consistency area of Hypothesis 1 and Hypothesis 2 is an important parameter of the algorithm. For the problem with $p = 12$, the comparison of the algorithm efficiency with various values of L_0 is shown in Fig. 4. The results of running the algorithm with use of Algorithm 5 as the generating procedure is shown as "L0=0". The best results are achieved with $L_o = 90$. The optimal value of L_0 depends on p. With $p = 32$, the best results are achieved with $L_0 = 60$. Nevertheless, the algorithm with a wide variety of the parameter values $L_0 \in [10, 350]$ gives the results better than the "classical" probability changing method (the case $L_0 = 0$).

For the further experiments, we used $L_0 = L_{avg}/3$ where L_{avg} is the expectation of the average distance to the closest facility in a randomly generated solution (arbitrary chosen set of p vertices):

$$L_{avg} = \mu\{\sum_{i=1}^{n} \min_{j \in \{\overline{1,p}\}} L(i,j)/n\}. \quad (19)$$

As the estimation the value L_{avg}, we used the average distance from 10 randomly chosen vertices to the closest vertex from 10 randomly generated sets of p vertices.

The calculation of L_{max} used in the conditions of the Hypotheses 1 and 2 takes significant time. Instead, we used $l_{avg}/L_{avg} \to 0$ (19) as the condition of applicability of our algorithm.

5 Comparison with the existing methods

For testing purposes, we used the local search method (Algorithm 3) with multistart from randomly generated initial

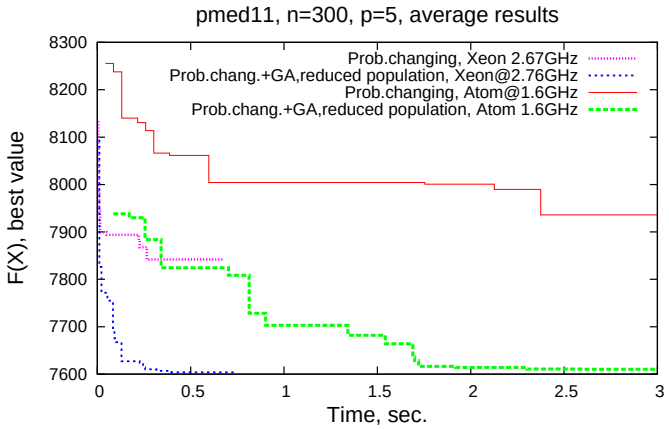
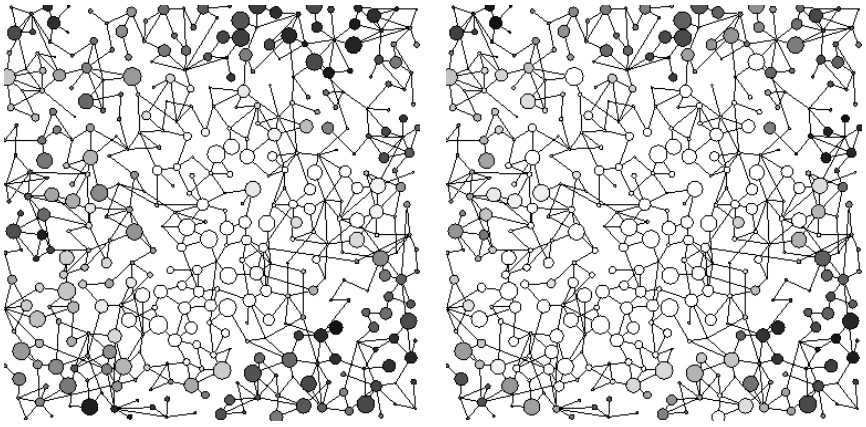


Figure 2: Comparison of the productivity on a medium (Xeon CPU) and small system (Atom CPU)



a) 3rd step b) 6th step

Figure 3: Probability values change $L_0 = 100$, $p = 3$

solution as one of the simplest methods and the genetic algorithm (GA) [1] with the crossover procedure given by Algorithm 4 (greedy heuristic) as one of the most efficient methods . As the testbed, we used the p-median problems from the OR Library [7]. The same library was used in [1]. Since this library contains problems with numbers of vertices up to $n = 900$, we used our Algorithm 7 for generating larger problems.

The results of our algorithm based on the probability changing method used standalone show its low convergence in comparison with the GA (Fig. 5). Problems "pmed22" and "pmed39" are included in the OR Library, a problem with $n = 5000$ was generated by Algorithm 7. This figure shows the average values for 10 runs and the worst result. The results of the combined method ("Probability changing+GA") are explained in the next section. To calculate the quantity of exemplars of the generated solutions in each population N_{POP}, we used formula

$$N_{POP} = \lceil \frac{\sqrt{n}C\left(\begin{array}{c} n \\ p \end{array}\right)}{100\lceil n/p \rceil} \rceil \lceil n/p \rceil. \quad (20)$$

The GA with greedy heuristic uses formula

$$N_{POP} = \lceil \frac{nC\left(\begin{array}{c} n \\ p \end{array}\right)}{100\lceil n/p \rceil} \rceil \lceil n/p \rceil. \quad (21)$$

6 Results of combined methods

The genetic algorithm [1] uses the regular method of filling of the initial population. In case of large-scale problems (pmed39, pmed32, generated problems with $n = 2000$, $n = 5000$), experiments with the randomly filled initial population decrease the accuracy of the results and the convergence insignificantly.

Our experiments with using the last generated population of the probability changing method as the initial population of the GA with greedy heuristic show significant speed-up of such combined algorithm in comparison with the original GA. We used two variants of the population size, standard population (21) and reduced population (20). Both variants show significant speed-up for most problems.

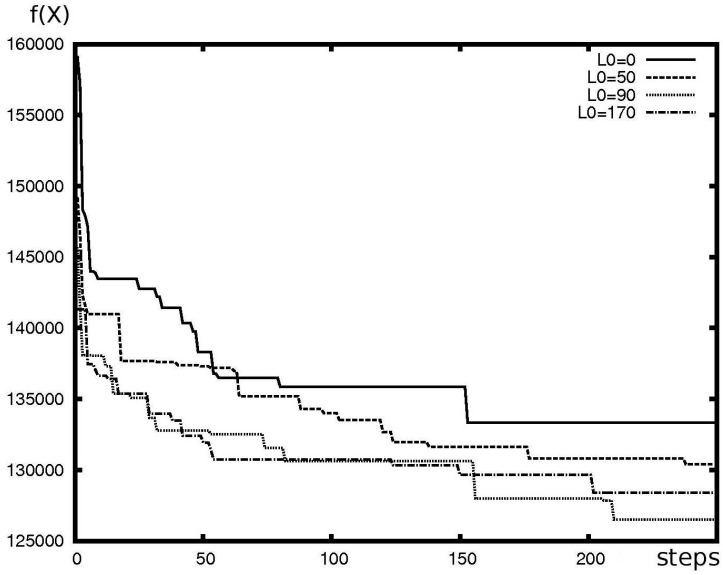


Figure 4: Comparison of the efficiency of the algorithm with various values L_0, $p = 12$

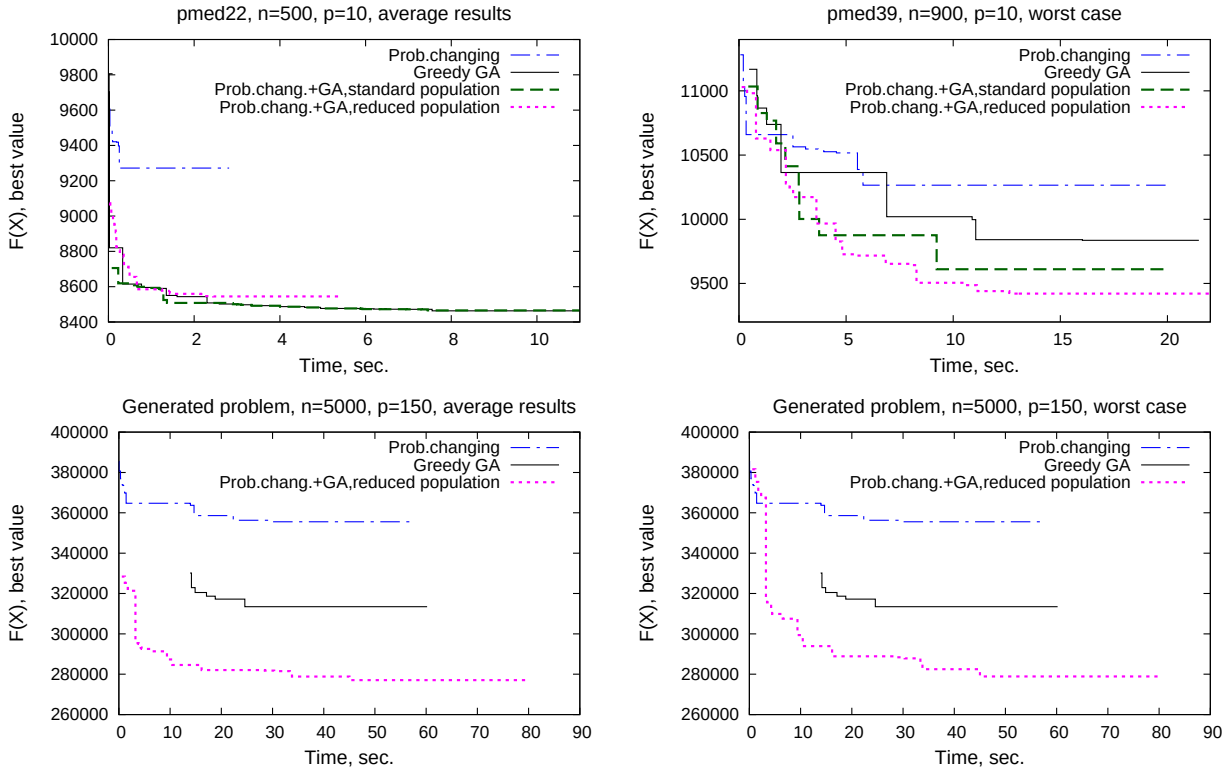


Figure 5: Comparison of the probability changing methods and its combinations with the GA

The variant with the reduced population shows worse accuracy but it can be useful for fast search for an approximate solution of the large-scale problems.

We performed 5 steps of Algorithm 1 ($N_{STEPS} = 5$ in the Step 6) with the probability adaptation (Algorithm 5) and used its last population $\{X_{5,i}|i = \overline{1, N_{POP}}\}$ as the initial population for the GA. The "chromosomes" $\mathcal{M} \in \{X_{5,i}|i = \overline{1, N_{POP}}\}$ are then passed to the crossover procedure (Algorithm 4).

The results shown in Fig. 5 and Fig. 6 were calculated for 10 runs of the original and combined algorithms (3 runs for $n = 7500$). The dependency of the average and worst

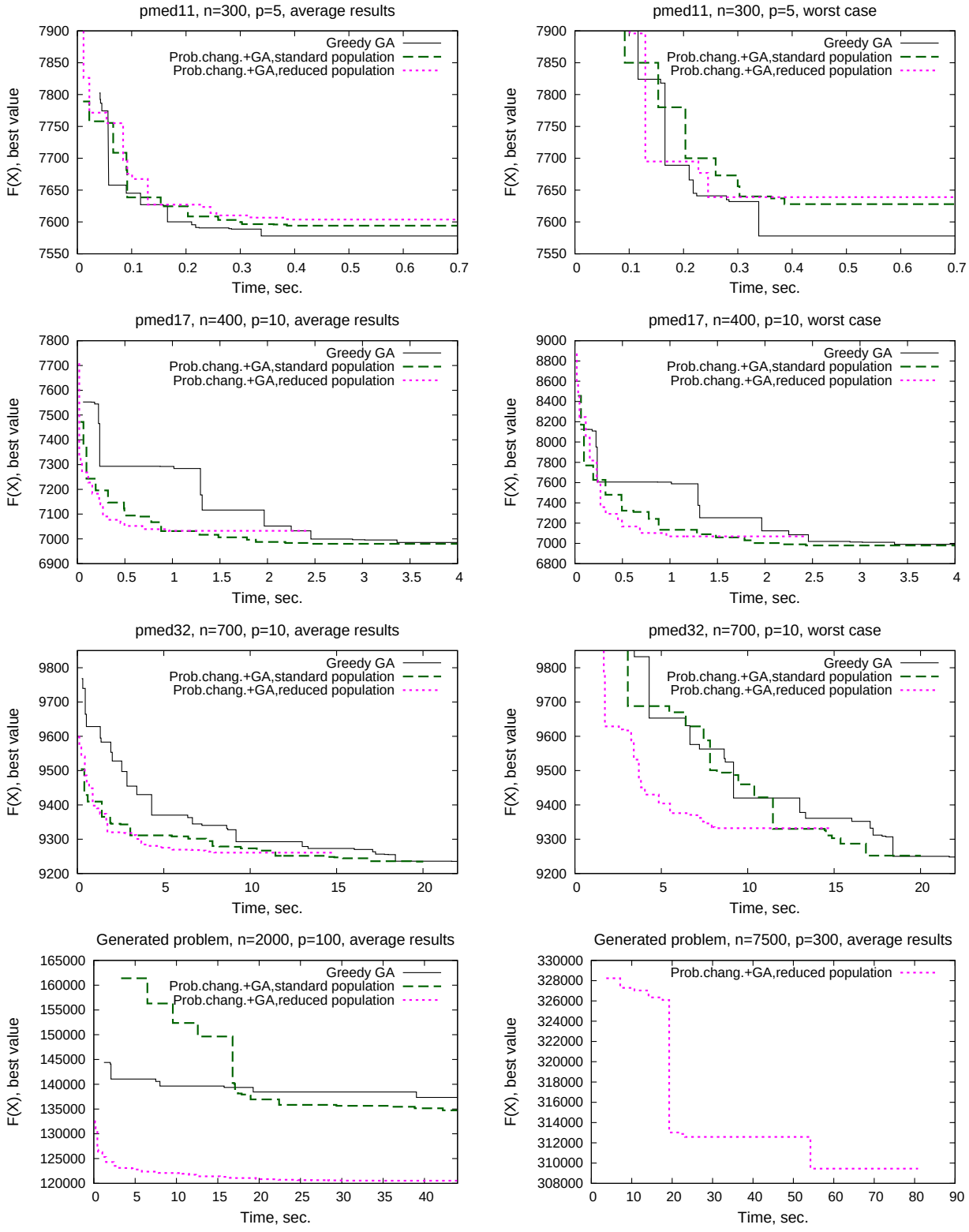


Figure 6: Comparison of the original and combined GAs

values achieved by the algorithms on the spent time was calculated for problems from the OR Library with comparatively small value of l_{avg}/L_{avg} (see (9) and (19)). The results for the problems "pmed11", "pmed12", "pmed14", "pmed16", "pmed18", "pmed19", "pmed21", "pmed23", "pmed35", "pmed31" show the analogous tendencies. We used a combination of 3 stop conditions: reaching the best result announced in the OR Library (if such exists), reaching $[\sqrt{n \cdot p}]$ steps which do not improve the best result or reaching the time limit.

For the problem with $n = 7500$, the results are shown for the combined algorithm with the reduced population only due to memory allocation problems in case of standard population (21).

In case of using the probability changing method, the standard deviation of the objective function in the population of decreases faster than in case of the GA (Fig. 7). In the combined algorithm, the search continues with a comparatively "good" population.

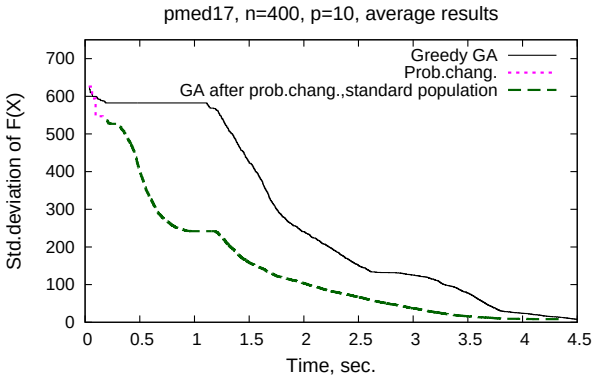


Figure 7: Comparison of the standard deviation of the original and combined GAs

We used the local search (Algorithm 3) with randomly selected initial solutions for testing purposes. Also, the local search was implemented as a procedure of the algorithm based on the probability changing method. We modified Step 2 of Algorithm 1 :

2: In accordance with the distribution given by the elements of the vector P, generate a set of N_{POP} vectors $X_{k,i}, i \in \{\overline{1, N_{POP}}\}$;

2.1: If $[k/5] = k/5$ and $k > 0$ then apply Algorithm 3 to each vector $X_{k,i}$;

The results are shown in Fig. 8. Both, original and combined algorithm ran 10 times. The size of the population of the probability changing algorithm was calculated in accordance with (20). For small size problems, local search in both variants is more efficient than the GA. For most problems included in the OR Library, the results of the combined method are the same as the results of the local search with multistart (case "a" on Fig. 8) because 2-100 starts of the local search procedure are enough for obtaining the exact solution. An exception is the problems with high density (p/n) and comparatively large size ("pmed19", "pmed24", "pmed25", "pmed29"). In this case (case "b" of Fig. 8), combined algorithm allows to reach the exact result faster. For the large scale problems ($n \geq 2000$, case "c") generated by Algorithm 7, the combined algorithm gives better results.

7 Conclusion

The proposed algorithm based on the probability changing method is useful for solving the p-median problem in a network under the assumption that the lengths of the edges are much smaller than the expectation of the path length from a randomly selected vertex to the closest vertex of the solution. Very slow convergence of the algorithm obstructs its implementation as a standalone algorithm. However, its running in combination with other algorithms improves their efficiency significantly. Adjusting the parameters of the algorithm is the subject to the future research.

References

[1] O. Alp, E. Erkut and Z. Drezner (2003) An Efficient Genetic Algorithm for the p-Median Problem, *Annals of Operations Research*, Vol.122, Issue 1–4, pp. 21–42, doi 10.1023/A:1026130003508

[2] A. Antamoshkin and I. Masich (2007) Pseudo-Boolean Optimization in Case of an Unconnected Feasible Set, *in Models and Algorithms for Global Optimization Optimization and Its Applications*, Springer Verlag, Vol. 4, pp 111–122

[3] A. N. Antamoshkin (1987) Optimizatsiya funktsionalov s bulevymi peremennymi (Optimization of Functionals with Boolean Variables), Tomsk University Press

[4] A. Antamoshkin, H. P. Schwefel, A. Toern, G. Yin, A. Zhilinskas (1993) *Systems Analysis, Design and Optimization. An Introduction*, Krasnoyarsk, Ofset

[5] P. Avella, M. Boccia, S. Salerno and I. Vasilyev (2012) An Aggregation Heuristic for Large Scale p-median Problem, *Computers & Operations Research*, 39 (7), pp. 1625–1632, doi 10.1016/j.cor.2011.09.016

[6] P. Avella, A. Sassano and I. Vasil'ev (2007) Computational Study of Large-Scale p-Median Problems, *Mathematical Programming*, 109(1), pp. 89–114, doi 10.1007/s10107-005-0700-6

[7] J. E. Beasley (1990) OR-Library: Distributing Test Problems by Electronic Mail, *Journal of the Operational Research Society*, 41(11), pp. 1069–1072

[8] M. Bischoff, T. Fleischmann and K. Klamroth (2009) The Multi-Facility Location-Allocation Problem with

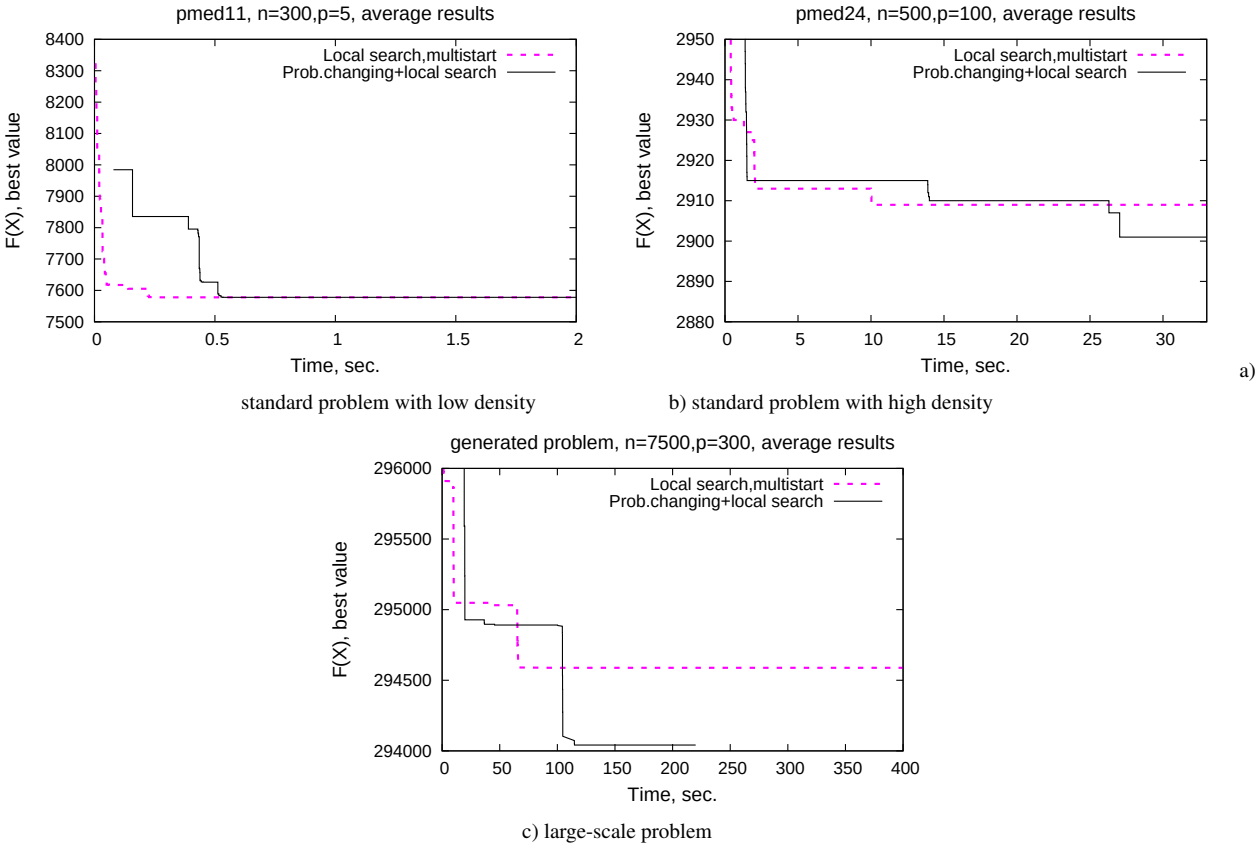


Figure 8: Comparison of the local search and the probability changing method with the local search procedure

Polyhedral Barriers, *Computers and operations Research*, 36, pp. 1376–1392

[9] B. Bozkaya, J. Zhang and E. Erkut (2002) A Genetic Algorithm for the p-Median Problem, *in Z. Drezner and H. Hamacher (eds.), Facility Location: Applications and Theory*, Springer

[10] A. V. Cabot, R. L. Francis and M. A. Stary (1970) A Network Flow Solution to a Rectilinear Distance Facility Location roblem, *American Institute of Industrial Engineers Transactions*, 2, pp. 132–141

[11] L. Cooper (1968) An Extension of the Generalized Weber Problem, *Journal of Regional Science*, Vol. 8, Issue 2, pp.181-197

[12] A. S. Degterev, F. V. Kanashkin and A. D. Sumarokov (2004) Obobshenie geneticheskikh algoritmov i algoritmov skhemy MIVER (Generalization of Genetic Algorithms and Algorithms Based on Probability Changing Method), *Issledovano v Rossii*, vol. 2004, pp. 1658–1665

[13] Z. Drezner and H. Hawacher (2004) *Facility location: applications and theory*, Springer-Verlag, Berlin, Heidelberg.

[14] Z. Drezner and G. O. Wesolowsky (1978) A Trajectory Method for the Optimization of the Multifacility Location Problem with lp Distances, *Management Science*, 24, pp.1507-1514

[15] R. Z. Farahani and M. Hekmatfar, editors (2009) *Facility Location Concepts, Models, Algorithms and Case Studies*, Springer-Verlag Berlin Heidelberg.

[16] M. Gugat and B. Pfeifer (2007) Weber Problems with Mixed Distances and Regional Demand, *Math. Methods of Orerations Research*, issue 66, pp. 419–449

[17] S. L. Hakimi (1964) Optimum Locations Of Switching Centers and the Absolute Centers and Medians of a Graph, *Operations Research*, 12(3), pp. 450-459

[18] P. Hansen, J. Brimberg, D. Urošević, N. Mladenović (2009) Solving large p-median clustering problems by primal–dual variable neighborhood search, *Data Mining and Knowledge Discovery*, vol. 19, issue 3, pp 351–375

[19] C. M. Hosage and M. F. Goodchild (1986) Discrete Space Location–Allocation Solutions from Genetic Algorithms, *Annals of Operations Research* 6, 35–46.

[20] O. Kariv and S. L. Hakimi (1979) An Algorithmic Approach to Network Location Problems. II: The P medians, *SIAM J. Appl. Math.* **37**, pp. 539–560.

[21] L. A. Kazakovtsev (2012) Adaptation of the Probability Changing Method for Weber Problem with an Arbitrary Metric, *Facta Universitatis, series Mathematics and Informatics*, vol. 27 (2), pp. 239–254

[22] L. Kazakovtsev (2012) Random Constrained Pseudo-Boolean Optimization Algorithm for Multiprocessor Systems and Clusters, *Proceedings of the IV International Congress on Ultra Modern Telecommunications and Control Systems 2012 (ICUMT)*, S. Petersburg, 23-25 September 2012, pp. 473–480 doi: 10.1109/ICUMT.2012.6459711

[23] V. Marianov and D. Serra (2009) Median Problems in Networks, *available at SSRN:* http://ssrn.com/abstract=1428362 or http://dx.doi.org/10.2139/ssrn.1428362

[24] S. Masuyama, T. Ibaraki and T. Hasegawa (1981) The Computational Complexity of the m-Center Problems on the Plane, *The Transactions of the Institute of Electronics and Comunication Engineers of Japan*, 64E, pp. 57–64

[25] N. Megiddo and K. Suppowit (1984) On the Complexity of Some Common Geometric Location Problems *SIAM Journal of Computing*, 13, pp. 182–196

[26] N. Mladenović, J. Brimberg, P. Hansen, J. A. Moreno-Perez (2007) The p-median problem: A survey of metaheuristic approaches, *European Journal of Operational Research*, Vol. 179, issue 3, pp.927–939

[27] J. G. Morris (1981) Convergence of the Weiszfeld Algorithm for Weber Problem Using a Generalized "Distance" Function, *Operations Research*, vol. 29 no. 1, pp. 37–48

[28] I. A. Osinuga and O. N. Bamigbola (2007) On the Minimum Norm Solution to the Weber Problem, *SAMSA Conference proceedings*, pp. 27–30

[29] M. Rabbani (2013) A Novel Approach for Solving a Constrained Location Allocation Problem, *International Journal of Industrial Engineering Computations*, published online, doi 10.5267/j.ijiec.2013.02.003, http://www.growingscience.com/ijiec/IJIEC_2013_8.pdf

[30] A. W. Reza, K. Dimyati, K. A. Noordin, A. S. M. Z. Kausar, Md. S. Sarker (2012) A Comprehensive Study of Optimization Algorithm for Wireless Coverage in Indoor Area, *Optimization Letters*, September 2012, published online, doi 10.1007/s11590-012-0543-z, http://link.springer.com/article/10.1007 %2Fs11590-012-0543-z?LI=true

[31] M. G. C. Resende (2008) Metaheuristic hybridization with Greedy Randomized Adaptive Search Procedures, *in TutORials in Operations Research, Zhi-Long Chen and S. Raghavan (Eds.)*, INFORMS, pp. 295–319

[32] M. G. C. Resende, C. C. Ribeiro, F. Glover and R. Marti (2010) Scatter search and path-relinking: Fundamentals, advances, and applications, *Handbook of Metaheuristics, 2nd Edition, M. Gendreau and J.-Y. Potvin, Eds.*, Springer pp. 87–107

[33] P. S. Stanimirovic, M. Ćirić, L. A. Kazakovtsev and I. A. Osinuga (2012) Single-Facility Weber Location Problem Based on the Lift Metric, *Facta Universitatis, series Mathematics and Informatics*, vol. 27 (2), pp. 31–46

[34] V. A. Trubin (1978) Effective algorithm for the Weber problem with a rectangular metric, *Cybernetics and Systems Analysis*, **14(6)**, DOI:10.1007/BF01070282, Translated from Kibernetika, **6** (November-December 1978), pp. 67–70.

[35] A. Weber (1922) *Über den Standort der Industrien, Erster Teil: Reine Theorie des Standortes*, Tübingen, Mohr

[36] E. Weiszfeld (1937) Sur le point sur lequel la somme des distances de n points donnes est minimum, *Tohoku Mathematical Journal*, **43** no.1, pp. 335–386.

[37] G. Wesolowsky (1992) The Weber problem: History and perspectives, *Location Science*, **1**, pp. 5–23.

[38] G. O. Wesolowsky and R. F. Love (1972) A nonlinear Approximation Method for Solving a Generalized Rectangular Distance Weber Problem, *Management Science*, vol. 18 no. 11, pp. 656–663

[39] G. G. Zabudski (2004) A Minimax Planar Location Problem with Forbidden Zones: Its Solution Algorithm, *Autom. Remote Control 65, No. 2, pp. 241–247*

Fingerprint Local Invariant Feature Extraction on GPU with CUDA

Ali Ismail Awad
Department of Computer Science, Electrical and Space Engineering
Luleå University of Technology, Luleå, Sweden
E-mail: ali.awad@ltu.se
Faculty of Engineering, Al Azhar University, Qena, Egypt

Keywords: biometrics, fingerprint images, processing time, SIFT, SURF, GPU, CUDA

Received: January 24, 2013

Driven from its uniqueness, immutability, acceptability, and low cost, fingerprint is in a forefront between biometric traits. Recently, the GPU has been considered as a promising parallel processing technology due to its high performance computing, commodity, and availability. Fingerprint authentication is keep growing, and includes the deployment of many image processing and computer vision algorithms. This paper introduces the fingerprint local invariant feature extraction using two dominant detectors, namely SIFT and SURF, which are running on the CPU and the GPU. The paper focuses on the consumed time as an important factor for fingerprint identification. The experimental results show that the GPU implementations produce promising behaviors for both SIFT and SURF compared to the CPU one. Moreover, the SURF feature detector provides shorter processing time compared to the SIFT CPU and GPU implementations.

Povzetek: Predstavljen je nov algoritem prepoznavanja prstnih odtisov.

1 Introduction

Biometrics authentication is an emerging technology, and it is a crucial need for different applications such as physical access and logical data access. It compensates some weakness of the traditional knowledge-based and toke-based authentication methods [1]. Fingerprint image, shown in Figure 1 (a), is one of the dominant biometric identifiers that keeps populate for its uniqueness, immutability, and low cost. Due to the high demand on fingerprint deployments, it receives a great research attention in order to enhance the overall performance of the automated fingerprint identification system. The aimed enhancements include the reduction of the system's response time, and the improvement of the system's identification accuracy [2].

Unfortunately, the system's response time comes from the summation of the consumed times by a consequence of system operations, which are defined as fingerprint enrolment, pre-processing, feature extraction, and features matching. Therefore, enhancing the response time should be carried out by investigating each system phase [3]. The accuracy of the fingerprint identification system depends on reducing the inter-user similarity by accurately extracting distinctive and robust features to image shift and rotation. The local invariant features are robust to image scale, shift, and rotation. These qualified features can be extracted using some local invariant feature detectors [4].

A local feature of an image is usually associated with a change of an image property such as intensity, color, and texture [5]. The advantages of using the local features in fingerprint identification are that they are computed at multiple points in the image, and hence, they are invariant to image scale and rotation. In addition, they do not need further image pre-processing or segmentation operations [6]. The dominant local feature extractors are the Scale Invariant Feature Transform (SIFT) [7], and the Speeded-Up Robust Feature (SURF) [8]. Due to their robustness and their time efficiency, SIFT and SURF have been deployed in a broad scope of applications such as object recognition [9], texture recognition, and image retrieval [10], [11]. Thus, the two feature detectors have been selected for this research, and they have been applied on a standard fingerprint images database.

Graphic Processing Unit (GPU) is a 3D graphic accelerator that is available in most of the recent computers, and it runs equivalent to the CPU with better performance in parallel processing tasks. The GPU programming opens doors to image processing, computer vision, and pattern recognition algorithms to run efficiently compared to the CPU implementations [12]. GPU supports different computing languages with minimal code changes to port the previously developed algorithms to GPU. The Compute Unified Device Architecture (CUDA) [13] is one of the GPU computing languages that supports "C" and "Fortran" languages for efficient algorithms deployment [12].

The response time degradation becomes even worse in the system identification mode because the system needs to run (1:N) matching operations over a huge size fingerprint database. Some research contributions tried to reduce the identification time by classifying fingerprint database [14], [15], while some other researchers focused on the matching process, and proposed an efficient Matching Score Matrix algorithm (MSM) [10], [16] over a fingerprint database to

(a) (b) (c)

Figure 1: Fingerprint local feature extraction and matching: (a) Raw fingerprint image, (b) Extracted local features using the SURF detector, and (c) Fingerprint matching using the extracted local features. The number of matched features have been reduced for a clear representation purpose.

achieve better identification time reduction. This paper targets the feature extraction time and the matching time reductions as two factors for enhancing the overall system's response or identification time.

The contribution of this paper is a step forward toward reducing the fingerprint feature extraction and features matching time, and hence, enhancing the overall system's response time. The contribution includes the development of SIFT and SURF local feature detectors on both CPU and GPU for processing a whole fingerprint database. Moreover, the behaviour of SIFT and SURF on both CPU and GPU with focus on the number of extracted features, the feature extraction time, and the features matching time are considered. The reported results in this research can be used as a start point and ground truth for developing many GPU based fingerprint algorithms in the future.

The rest of this paper is organized as follows. Section 2 reviews the preliminaries backgrounds for the local feature extraction, and the graphic processing unit. Section 3 sheds lights on the implementation methodologies, and explains the exhaustive evaluations of both SIFT and SURF feature detectors deployed on the both CPU and GPU, in terms of processing time, number of features, and features matching. Conclusions and future work are reported in section 4.

2 Preliminaries

This section clarifies the local feature extraction, and highlights the different structures of the SIFT and the SURF feature detectors. Moreover, it covers the GPU architecture compared to the CPU one.

2.1 Local Feature Detectors

SIFT is one of the popular methods for image matching and object recognition. The SIFT feature detector has been used by some researchers in biometrics based authentication with applications on fingerprints [10] and palmprints [17]. Due to its reliability, SIFT features are used to overcome different fingerprint degradations such as noise, partiality, and rotations.

The SIFT feature detector works through sequential steps of operations. These steps can be summarized as: 1) Scale space extrema detection, 2) Keypoints localization, 3) Keypoint orientation assignment, and 4) Building the keypoints descriptor [18], [19]. The Difference-of-Gaussian (DOG) is used to detect the keypoints as the local extrema of DOG function. The DoG function is defined as [18]:

$$\begin{aligned} D(x,y,\sigma) &= (G(x,y,k\sigma) - G(x,y,\sigma)) * I(x,y) \\ &= L(x,y,k\sigma) - L(x,y,\sigma), \end{aligned} \quad (1)$$

where $I(x,y)$ is the input image at point (x,y) pixels, and $G(x,y,\sigma)$ is the variable-scale Gaussian function that is defined as:

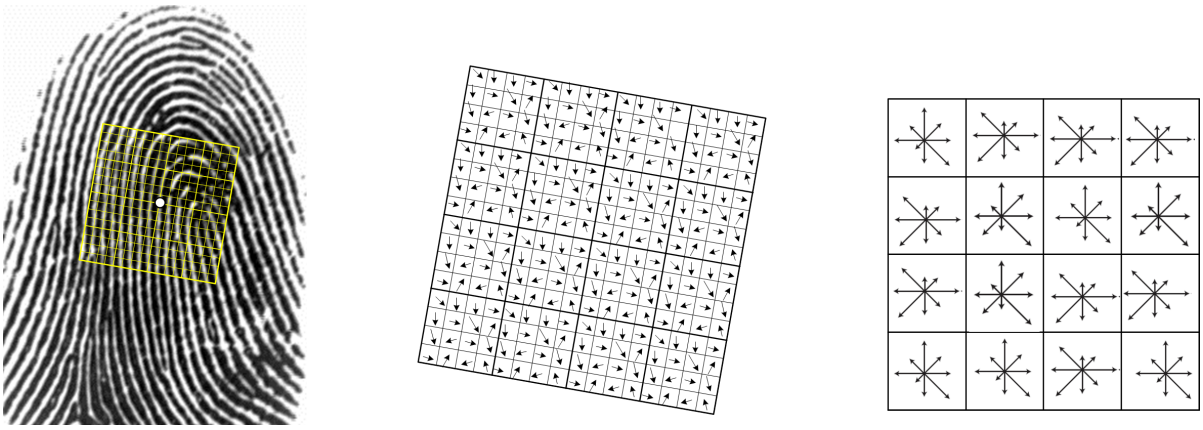
$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}, \quad (2)$$

and $L(x,y,\sigma)$ is the scale space of an image, and it is defined as:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y), \quad (3)$$

where $(*)$ represents a convolution operation.

The pixel is compared against 26 neighboring pixels (8 in the same scale, 9 in the above scale, and 9 in the below scale) to detect the local pixel extrema and minima. Following on, the detected keypoint is localized by setting its neighborhoods and examine them for contrast and edge parameters. The keypoints with low contrast and weak edge responses are then rejected. The keypoint neighborhoods region is used to build the histogram of the local gradient directions, and the keypoint orientation is calculated as the peak of the gradient histogram [11]. The default SIFT feature extraction produces keypoint associated with a descriptor of 128 element length. The descriptor is constructed from $(4 \times 4$ cells$) \times 8$ orientations [19]. The cascaded operations of building a single SIFT keypoint descriptor from fingerprint image, and the descriptor struc-



(a) (b) (c)

Figure 2: The process of building a single SIFT keypoint descriptor: (a) A single SIFT keypoint selected from fingerprint image, (b) 16×16 pixel gradients, and (c) A single 4×4 cells keypoint descriptor with 8 pixel orientations each. The default length of a single SIFT keypoint descriptor is $4 \times 4 \times 8 = 128$ element.

ture are shown in Figure 2. Applying SIFT feature extraction translates the fingerprint image into a set of keypoints according to the detected local maxima. The keypoint is associated with a descriptor related to the gradients of the enclosed pixels.

The SURF feature detector works in a different way from SIFT. The SURF keypoint detector uses Hessian matrix for keypoint extraction compound with the integral image to increase the computation efficiency. The Hessian matrix $H(x, \sigma)$ is calculated at a given point $p = (x, y)$ pixels of image I at scale σ as [8]:

$$H(x, \sigma) = \begin{bmatrix} S_{xx}(x, \sigma) & S_{xy}(x, \sigma) \\ S_{xy}(x, \sigma) & S_{yy}(x, \sigma) \end{bmatrix}, \tag{4}$$

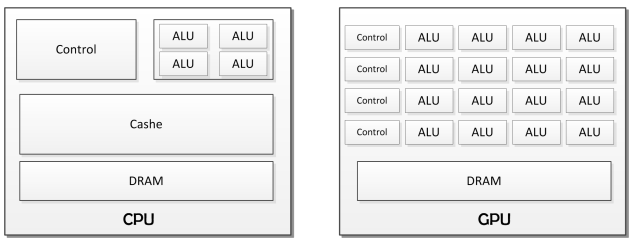
where $S_{xx}(x, \sigma)$ is the convolution $(*)$ of the Gaussian second order derivative with the image I in a point p.

The SURF descriptor is formed around the keypoints neighborhood by using Haar wavelet responses instead of gradient histogram applied in the SIFT. The standard length of the SURF descriptor can be 64 (4×4 subregions $\times$ 4 Wavelet components), 36, and 128 vector length. The SURF features are found to be an efficient compared to the SIFT one with preserved repeatability, robustness, and distinctiveness of the descriptors [8], [19]. Figure 1 (b) and (c) show the feature extracted using the SURF detector, and a sample of the local features matching process, respectively.

2.2 Graphic Processing Unit

The GPU is a multi-processor unit equipped with four types of memories that are defined as constant, texture, global, and shared memory for efficient data access. The GPU was initially designed for processing graphic functions, and it was required special skills for programming such functions via OpenGL [20]. The hardware architecture of the GPU is internally different from the CPU design. The two hardware design architectures are shown in Figure 3 (a) and (b) for the CPU and the GPU, respectively. The GPU architecture takes into its account the existence of many Arithmetic and Logic Units (ALUs) devoted for parallel processing and flow control [21].

The full power of the GPU architecture can be accessed via CUDA computing language as the threads are grouped into blocks, the blocks are grouped into grids, and the grid is executed as a single GPU kernel [20]. In real execution, the blocks are mapped to the GPU cores for efficient scalability. The CUDA computing language is designed to support general purpose computing on GPU. CUDA programming brings a development environment similar to "C" which is familiar to most of programmers and researchers. Additionally, minimal changes are required to port the CPU based code to the CUDA programming scheme, and be compiled using the provided NVCC compiler. Furthermore, CUDA introduces additional "C" libraries such as cuFFT for Fast Fourier Transform, and cuRAND for random number generation. Concisely, CUDA provides an integrated "C" similar environment for building the GPU based applications [13].



(a) (b)

Figure 3: The differences between the CPU and the GPU hardware design architectures: (a) The CPU hardware architecture, (b) The GPU hardware architecture with multiple Arithmetic and Logic Units.

3 Performance Evaluation

The open source computer vision library (OpenCV) [22] is a promising tool for developing most of computer vision and image processing algorithms. Recently, a new OpenCV module that provides a GPU acceleration, and covers the most significate part of the library was added to the OpenCV [20]. Although, the GPU module provides an implementation to the SURF feature detector, it does not provide the same implementation to the SIFT detector. Therefore, we have adopted, compiled, and worked with the SIFTS [23] as an open source library for evaluating the SIFT performance using the GPU implementation.

Prior to the evaluation process, OpenCV version 2.4.2 has been compiled with CUDA 4.2, the Threading Building Blocks (TBB) for the GPU, and the multi-core CPU supports, respectively. The local features matching was carried on the CPU using OpenCV `BruteForceMatcher` with `knnMatch` support. During the experimental work, the optimum `Knn` radius is set to 1 for the best matching performance.

3.1 Experimental Environment Setup

The experiments have been conducted using a PC with Intel® Core™ i3-2120 running at 3.30 GHz, and 8 GB of RAM. The PC is empowered by NVIDIA GeForce GT 430 GPU with 96 CUDA cores, and 1 GB of memory running on Windows® 64-bit. It is worth noticing that the quality of the extracted features is not considered in the present phase of this research.

The experimental work was applied on a standard fingerprint database, namely, the Fingerprint Verification Competition 2002 (FVC2002) [24] DB2_B subset. The FVC2002 DB2_B subset includes 80 fingerprint images come from 10 different persons. Therefore, the feature extraction using SIFT or SURF has been repeated 80 times. Whereas the matching process has been repeated over 80×80 images to produce a matching matrix with 6400 elements for evaluating the CPU based matching, and 80 elements for evaluating the GPU based matching with the average time.

3.2 Performance Evaluation of SIFT

According to the previous evaluation of the SIFT feature detector reported in [10], we set the optimum SIFT `threshold` to 0.01, and the other OpenCV SIFT parameters are set to the default values. The results of the SIFT evaluation on the CPU and on the GPU are shown in Figure 4. The plotted data are the average value of running the SIFT detector over the full database subset, and the matching time on the CPU is the average of 6400 matching processes.

The experimental results prove the significant reduction of the feature extraction time, and the features matching time when running the SIFT on the GPU. The time reduction comes from the powerful 96 CUDA cores supported by the NVIDIA GPU hardware compared to the two cores with two threads each supported by the CPU. The amount of the speed ups of running the SIFT detector on the GPU are 3.9X and 67.2X for the SIFT feature extraction, and the SIFT features matching, respectively.

The CPU and the GPU utilizations for a particular 14 seconds time period are drawn in Figure 5. From that figure, the CPU utilization starts low (around 25%) during the feature extraction phase, and it goes extremely high (around 98%) during the SIFT feature matching. On the other hand, the GPU load is fixed around 85% during the feature extraction and matching operations. Additionally, the figure gives an indication that is however, the matching time is extremely reduced, the full power of the GPU utilization still below the CPU one.

3.3 Performance Evaluation of SURF

The SURF feature detector has been evaluated on the CPU and the GPU using the same SIFT evaluation criteria, re-

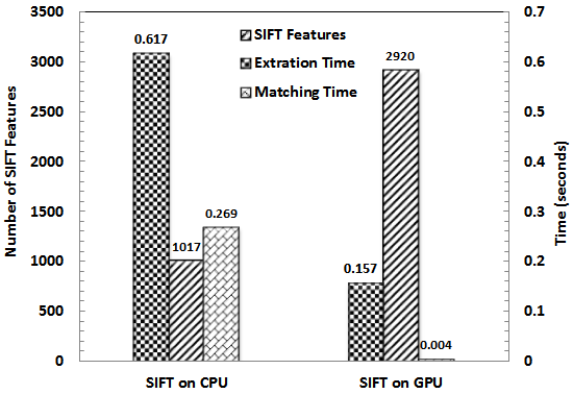


Figure 4: The evaluation of the SIFT detector on the CPU and the GPU with respect to the number of features, the extraction time, and the matching time.

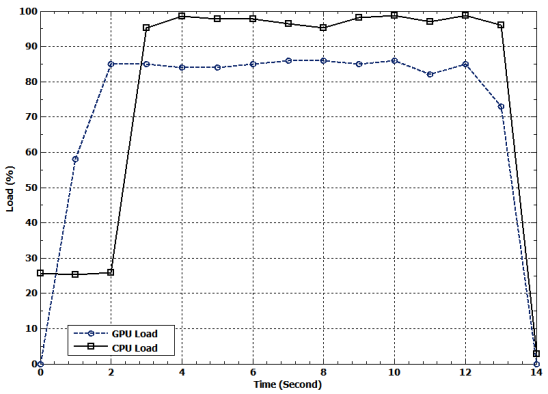


Figure 5: The CPU and the GPU utilizations of the SIFT detector measured in a 14 seconds long as a particular time slot.

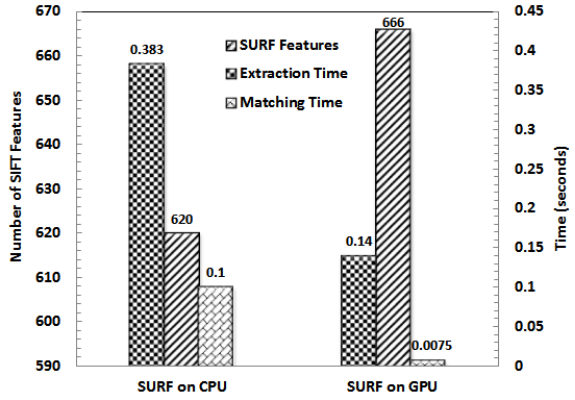


Figure 6: The evaluation of the SURF detector on the CPU and the GPU with respect to the number of features, the extraction time, and the matching time.

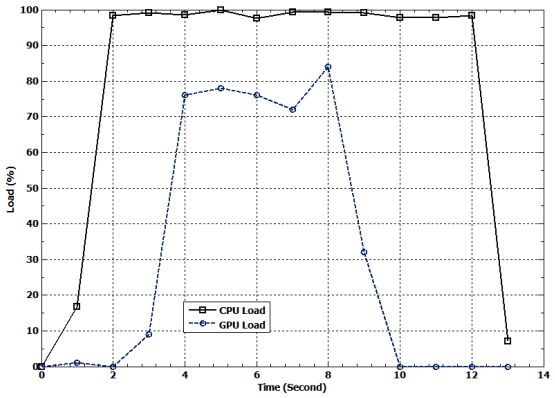


Figure 7: The CPU and the GPU utilizations of the SURF detector measured in a 14 seconds long as a particular time slot.

spectively. According to the evaluation results shown in Figure 6, the advantage of running the SURF detector on the GPU compared to the CPU is apparent. However, the number of extracted features on the CPU and the GPU are almost same. The feature extraction time on GPU is significantly reduced from 0.383 to 0.140 second. Moreover, the features matching time is reduced from 0.1 to 0.0075 second with almost the same number of features. The amount of the gained speeds up of using the GPU are 2.7X and 13.3X for the SURF feature extraction, and features matching.

Once again, the CPU and the GPU utilizations have been measured and reported in Figure 7. The SURF feature detector behaves very well on the GPU, and it consumes around 70 to 80% of the GPU power. On the other side, the SURF feature detector consumes around 98% from the total CPU power.

3.4 Discussion

The experimental work confirms the efficiency, and the superiority of the SURF feature detector compared to the SIFT feature detector. The evaluation factors are the feature extraction time, and the features matching time. From Figure 4 and Figure 6, the SURF feature detector runs twice faster than the SIFT one on the CPU due to the SURF enhancements such as using the integral images, Hessian matrix, and Haar Wavelet for keypoint detection and descriptor construction.

However, the SURF feature detector is optimized for a faster running on the CPU and the GPU; Figure 6 represents a higher matching time on the GPU (0.0075 second) compared to the SIFT (0.004 second). We do believe that the difference comes from the data transfer between the CPU and the GPU. The OpenCV implementation provides data upload and download between the CPU and the GPU for each matching process. This confirms the fact that the data transfer between the CPU and the GPU still a great challenge, and it should be avoided as much as possible for efficient GPU based implementations [20].

4 Conclusions and Future Work

This paper has presented a study for reducing the fingerprint identification time using two dominant local feature detectors, particularly SIFT and SURF implemented on CPU and GPU. The aim of the paper is to find the best implementation of a feature detector in terms of the number of features, the feature extraction time, and the features matching time. The experimental results proved the superiority of the SURF feature detector compared to the SIFT one. Moreover, the efficiency of using the GPU for both SURF and SIFT has been confirmed. However, SURF consumes longer matching time on the GPU, this time can be significantly reduced by avoiding the data transfer between the CPU and the GPU. Optimizing the SURF feature detector to work completely on the GPU, and avoiding the data transfer between the CPU and the GPU is considered as an appreciated future work. In addition, deploying the fingerprint related algorithms to work on the GPU is a common future direction.

5 Acknowledgments

We express sincere thanks to Professor Kensuke Baba, Kyushu University, and Ms Serina Egawa, Kyushu University, for providing the primary OpenCV source code for evaluating the SIFT detector on the CPU.

References

[1] Awad, A.I.: Machine learning techniques for fingerprint identification: A short review. In: Hassanien, A.E., Salem, A.B.M., Ramadan, R., Kim, T.h. (eds.) Advanced Machine Learning Technologies and Applications, Communications in Computer and Infor-

mation Science, Vol. 322, pp. 524–531. Springer Berlin Heidelberg (2012)

[2] Maltoni, D., Maio, D., Jain, A.K., Prabhakar, S.: Handbook of Fingerprint Recognition, Second Edition. Springer-Verlag (2009)

[3] Egawa, S., Awad, A.I., Baba, K.: Evaluation of acceleration algorithm for biometric identification. In: Networked Digital Technologies. Part 2, Communications in Computer and Information Science, Vol. 294, pp. 231–242. Springer (2012)

[4] Jain, A.K., Ross, A.A., Nandakumar, K.: Introduction to Biometrics. Springer (2011)

[5] Mikolajczyk, K., Tuytelaars, T.: Local image features. In: Li, S., Jain, A. (eds.) Encyclopedia of Biometrics, pp. 939–943. Springer US (2009)

[6] Tuytelaars, T., Mikolajczyk, K.: Local invariant feature detectors: A survey. Foundations and Trends in Computer Graphics and Vision 3(3), 177–280 (Jul 2008)

[7] Lowe, D.G.: Object recognition from local scale-invariant features. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1150–1157 (1999)

[8] Bay, H., Ess, A., Tuytelaars, T., Van Gool, L.: Speeded-up robust features (SURF). Computer Vision and Image Understanding 110(3), 346–359 (Jun 2008)

[9] Baran, J., Gauch, J.: Motion tracking in video sequences using watershed regions and SURF features. In: Proceedings of the 50^{th} Annual Southeast Regional Conference. pp. 256–261. ACM-SE '12, ACM, NY, USA (2012)

[10] Awad, A.I., Baba, K.: Evaluation of a fingerprint identification algorithm with SIFT features. In: Proceedings of the 3^{rd} 2012 IIAI International Conference on Advanced Applied Informatics. pp. 129–132. IEEE, Fukuoka, Japan (September 2012)

[11] Mikolajczyk, K., Schmid, C.: A performance evaluation of local descriptors. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(10), 1615–1630 (2005)

[12] Wynters, E.: Parallel processing on NVIDIA graphics processing units using CUDA. Journal of Computing Sciences in Colleges 26(3), 58–66 (January 2011)

[13] NVIDIA Compute Unified Device Architecture (CUDA), `http://www.nvidia.com/`

[14] Liu, M.: Fingerprint classification based on Adaboost learning from singularity features. Pattern Recognition 43(3), 1062–1070 (2010)

[15] Park, C.H., Park, H.: Fingerprint classification using fast Fourier transform and nonlinear discriminate analysis. Pattern Recognition 38(4), 495–503 (April 2005)

[16] Maeda, T., Matsushita, M., Sasakawa, K.: Identification algorithm using a matching score matrix. IEICE Transactions on Information and Systems E84-D(7), 819–824 (2001)

[17] Chen, J., Moon, Y.S.: Using SIFT features in palmprint authentication. In: Proceedings of the 19^{th} International Conference on Pattern Recognition. pp. 1–4. IEEE (2008)

[18] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)

[19] Saleem, S., Bais, A., Sablatnig, R.: A performance evaluation of SIFT and SURF for multispectral image matching. In: Campilho, A., Kamel, M. (eds.) Image Analysis and Recognition, Lecture Notes in Computer Science, Vol. 7324, pp. 166–173. Springer Berlin / Heidelberg (2012)

[20] Pulli, K., Baksheev, A., Kornyakov, K., Eruhimov, V.: Real-time computer vision with OpenCV. Communications of the ACM 55(6), 61–69 (June 2012)

[21] Poli, G., Saito, J.H.: Parallel face recognition processing using neocognitron neural network and GPU with CUDA high performance architecture. In: Oravec, M. (ed.) Face Recognition, pp. 381–404. In-Tech (2010)

[22] OpenCV: Open Source Computer Vision library, `http://opencv.willowgarage.com/wiki/`

[23] Wu, C.: SiftGPU: A GPU implementation of scale invariant feature transform (SIFT) (2012), `http://cs.unc.edu/~ccwu/siftgpu/`

[24] Maio, D., Maltoni, D., Cappelli, R., Wayman, J., Jain, A.K.: FVC2002: Second Fingerprint Verification Competition. In: Proceedings of the 16^{th} International Conference on Pattern Recognition (ICPR2002), Quebec City. pp. 811–814 (2002)

Bit-projection Based Color Image Encryption using a Virtual Rotated View

Brahim Nini
University of Oum El-bouaghi, Po. Box 358, 04000. Algeria. Tel. +213 661 487 057
E-mail: b.nini@univ-oeb.dz

Keywords: image encryption, bit permutation, pixel substitution, projection, security

Received: April 10, 2013

This paper presents a novel algorithm for a color image encryption which involves simultaneously two operations in one: permutation and substitution of pixels. It uses the rows and columns of images' bits as transformation units. Each bit is regarded as an observed entity having a light ray which intersects a new rotated image. The intersection position is used to paste the corresponding bit. Such projection makes the pixels' bits migrate between each other which generate a cipher image. Despite its simplicity, the algorithm shows a great resistance against many kinds of attacks through its sensitivity to the initial values used in encryption.

Povzetek: Opisan je nov algoritem za kodiranje barvnih slik.

1 Introduction

It becomes commonly known that any important data exchange, mainly through a network, should be subject to a previous encryption in order to avoid its misuse. The particular case of images and videos attracts more attention due to the bulk of direct interpreted information they hold.

There are three major kinds of methods used to construct secure encryption algorithms: permutation, substitution, and their combining form. The first kind, called in other works confusion, is an image shuffling which makes the content hidden and confusing. The second one, called in other works diffusion, is the coding of all pixels' values by new values instead of the original ones. The aim of both techniques, either used separately or combined, is to make the retrieval of the original image by attackers either impossible or very difficult. Unfortunately, totally secure cryptographic algorithms are difficult to build because solutions can always be found to defeat them.

Many kinds of algorithms have been extensively addressed in the literature. Recently, chaotic systems have been widely used in data encryption [5, 6, 7, 16, 17, 18]. Many ideas are proposed. The one presented in [6], for instance, is based on image permutation and proposes the scrambling/permuting binary bit of every pixel with pseudo-random number sequence generated by chaotic logistic map. Another method is proposed in [5]. It is a cryptographic algorithm which uses Maximum Distance Separable (MDS) matrices as part of its diffusion element. From another point of view, many other kinds of algorithms are proposed [8, 9, 11, 14, 15]. For instance, a position permutation algorithm by magic cube transformation for the permutation process is used in [8]. It is based on a transformation of magic cube's face values. It was proposed by [14] who improves it through consecutive work. Furthermore, another approach is proposed in [11] which is based on combinations of hybrid magic cubes which are generated from a magic square and two orthogonal Latin squares. Likewise, a matrix scrambling is used in [15]. It is based on shifting and exchanging rule of bi-column bi-row circular queue. In sum, several ideas are used for the purpose of image encryption algorithms development and each one has its advantages and drawbacks. In the reference [19], one may find out many recent advances in the field through the presented survey.

Though the quality of many works, results improvement is always required. For this, some works are oriented to analysis and/or refining of existing algorithms [2, 3, 4, 10, 12, 13]. These works focus on finding out weaknesses of developed methods and the way they can be strengthen. In [12] for instance, some means are proposed to strengthen the overall performance of the security of Fridrich's algorithm. Though there is no general model for security analysis suitable for all cryptosystems using different methods, some means are used to measure the strength of security and the speed of encryption process. They are based on the study of the behavior of encryption algorithms against attacks. The key and encrypted content are then analyzed whether they are safe or not against brute force, statistical, known plaintext, select plaintext, differential, and other attacks.

Following the domain's stream, this work proposes a new idea of color image encryption extending a previous one [1]. The latter was limited to an image permutation for its encryption purpose; but this one generalizes it to

image encryption. It merges both permutation and substitution of pixels in one indiscernible operation.

In this work, an image is considered as rows and columns of bits. Each bit is regarded as an observed entity having a light ray. On another hand, a virtual viewer, who is assumed revolving about the middle axis of the image, involves a new view of the image that looks as if it is rotated so that its plane is perpendicular to the direction of the view. Such view is used as a new image onto which a projection is done. Accordingly, the intersection of a virtual bit's light ray to the viewer with the rotated image is used as the position of projection. This makes the bits moving within different pixels.

Since the same idea of projection in the former work [1] is used, the same effects are met. The projection process leads to some parts of the original image having their projection positions either out of bounds or being the same. The former case is evident due to the image plane surface limits. The latter is so, because the computed positions should be rounded to their nearest integers, and hence, some bits may have the same target positions. So, in order to avoid information loss, only one bit is projected in each position and the others are delayed together with out-projected bits. As a result, the new reconstructed image contains many holes where some positions are undefined, i.e., not originating from the original image. The idea of the proposed algorithm consists in refilling these holes by the delayed bits in a reverse order.

Our algorithm is nearly close in its function to the one proposed in [9] in that it repeats the permutation process several times with different values of the key. The difference is that, in our case, the key does not change dynamically during the permutation process, but might have different combining forms.

Moreover, the proposed algorithm holds some important features that make it a powerful encryption technique. This is proved through some measures based on some means proposed in [12] and detailed later. Moreover, this algorithm's features are challenging comparatively to other works in the following ways:

– The algorithm is based on a key whose parts are the parameters of a virtual viewer's position. Hence, the values of the key are understandable and measurable physical quantities,

– The encryption process does not rely on the key values only. Their combining form is also important. It has a flexible structure which is not previously established and depends on the user's choices,

– Permutation and substitution are combined within the same operation,

– The algorithm does not depend on the machine's precision neither in encryption nor in decryption processes. It uses very low precision in computing and provides high sensitivity.

The rest of the paper is organized as follows. The next section explains the projection process. It gives the underneath mathematical basis of the proposed projection. The third section explains how an image is encrypted. It gives the algorithm, its complexity, the structure of the used key, and explains how the reverse process of decryption is done. The fourth section gives an analysis about the key space values and its sensitivity. The fifth section is interested in the analysis of the resistance of the algorithm against some types of attacks. Finally, a conclusion is given summarizing the work and its extensions.

2 Projection process

Figure 1 shows a cut of the reference position from which an image is assumed to be viewed in its normal appearance, and another view's position rotated about the central axis. The latter position involves a rotated plane that is perpendicular to the direction of the viewer's position onto which the image is to be projected. Such position is defined by its inclination angle from the reference one.

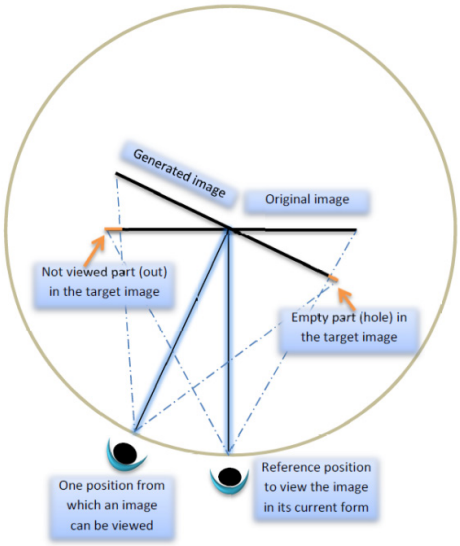


Figure 1: Transformation of an original image into a new one when viewed from another position.

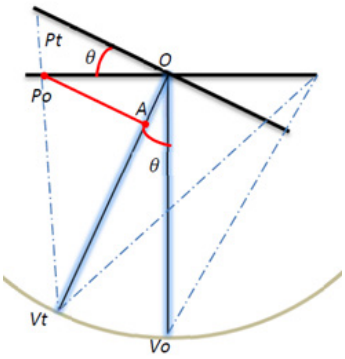


Figure 2: Projection of an original point P_o onto P_t in the new image.

The projection process is based on two assumptions. Firstly, the image is considered as three color planes in their low level data arrangement; i.e. the rows of each plane are vectors of bits where pixels are the groups of consecutive b bits depending on the image's gray level, and columns are the bits of the same order of image's columns of pixels. Secondly, each bit is assumed having

a light ray that converges towards the viewer's position. Such ray crosses the rotated plane at a given point which is displaced from the bit's initial position. The new position is then used to paste the bit's value.

Figure 2 gives the schema of how a point P_o is projected at the position P_t when the viewer moves around the central axis of the image from the position V_o to the position V_t. Therefore, the achievement of the projection relies on the mathematical expression which governs the amount of OP_t.

Let the angle between the directions of the reference position and the new one be θ (figure 2). This angle is the same between the original image plane and the rotated one. Let now A be the perpendicular projection of P_o on OV_t. Using ΔOP_tV_t and ΔAOP_o, it is easy to see that $AP_o = OP_o.cos\theta$, and $OA = OP_o.sin\theta$. Using these expressions, and based on the truth of expression (1):

$$\frac{OP_t}{AP_o} = \frac{OV_t}{AV_t} \tag{1}$$

the amount of OP_t can be computed for the half part of the image being on the same side of the viewer's position (2):

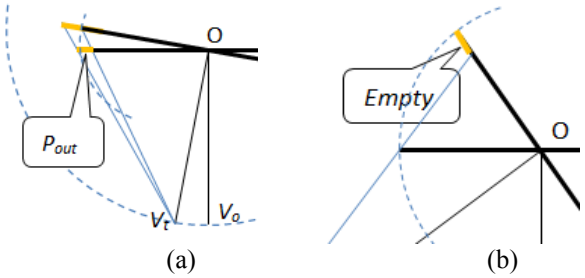
$$OP_t = \frac{OV_t.OP_o.cos\theta}{OV_t - OP_o.sin\theta} \tag{2}$$

This theory is applied in either horizontal or vertical direction. Hence, expression (2) is used to project the bits of each half line or half column of the original image onto the same half line or half column of the new image. So, each bit in the original image at the position P_o is projected onto the position P_t. Such transformation needs two basic parameters to be known: the angle θ and the distance $OV_o = OV_t$. Practically, we take the value of the distance as being $d.lc$ where lc stands for the number of lines or columns, and d is a real such that $d \geq \frac{1}{2}$. The latter condition is important since it ensures that $OV_t - OP_o.sin\theta \neq 0$, because in this case we get $sin\theta = \frac{OV_t}{OP_o} = \frac{d.2OP_o}{OP_o} = 2d \geq 1$ which is impossible according to the previous condition. The case where $sin\theta = 1$ is meaningless since it is practically impossible. It gives $\theta = \frac{\pi}{2}$, making the projection of the whole image being one point and encryption process is reduced to inversion of bits' positions.

Note that the relationship (3) that links P_t and P_o on the opposite side of the viewer's position is so complicated, and consequently, has a negative impact on the algorithm complexity and time processing. This is why, the only expression used in this work on both sides of an image is (2), and since expression (3) is not used, it is given without demonstration.

$$OP_o = OP_t.cos\theta + \frac{OP_t.sin\theta}{cot\left(arctan\left(\frac{OP_t}{OV_t}\right) - \theta\right)} \tag{3}$$

The projection of each line or column of the original image on a new one has some effects. If the angle θ is small, a set of collated bits on the border, called P_{out}, are projected outside the boundary of the new image (figure 3(a)). This creates stretched lines or columns with many spread undefined bits, since a reduced number of bits become spread over the whole line or column. If θ is big, it results in an undefined part in the projected image (figure 3(b)). Furthermore, due to the rounding of the estimated value of OP_t to its nearest integer, some bits are going to have the same projection positions, especially when an empty part is created for a big θ. In this case, only one bit is pasted at the shared position and the others are not. All those which are not pasted create the set called P_{rep}. The two sets, P_{out} and P_{rep}, are initially delayed from projection. As a result, the obtained image might contain several undefined bits called holes.



(a) (b)

Figure 3. Effects of θ and d on the projection: (a) some bits are projected outside, (b) part of the border remains empty.

The value of θ_l which is the limit between those that create either P_{out} or an empty part is given when OP_t and OP_o become equal. In other words, if $\theta > \theta_l$ then $P_{out} = \emptyset$. So, θ_l is the angle when $OP_o = OP_t = \frac{lc}{2}$. Expressing this condition in (2) gives $lc = \frac{2d.lc^2.cos\theta_l}{2d.lc - lc.sin\theta_l}$, which results in the following equation:

$$2d.cos\theta_l + sin\theta_l = 2d \tag{4}$$

The solution of (4) is the one of:

$$cos(\theta_l - \beta) = \frac{2d}{\sqrt{1+4d^2}} \tag{5}$$

where $tan\beta = \frac{1}{2d}$. The solution to (5) is then:

$$\theta_l = arccos\left(\frac{2d}{\sqrt{1+4d^2}}\right) + arctan\left(\frac{1}{2d}\right) \tag{6}$$

and

$$\lim_{d \to +\infty} \theta_l = 0 \tag{7}$$

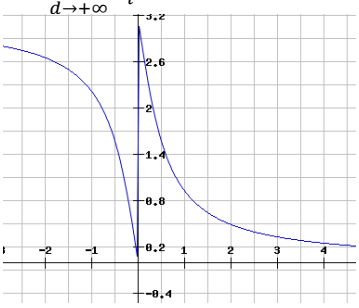


Figure 4: Graph of θ_l related to function (6).

The graph of θ_l (figure 4) shows that the bigger d is, i.e. the viewer is far from the image, the smaller θ_l is. In the graph of figure 4, the plot of the function where $d \geq \frac{1}{2}$ is the only part of our interest.

3 Image encryption

3.1 Cipher procedure

The first step of shuffling procedure is the projection of each line or column on the same line or column respectively of the new image. This goes through a vector calculation linking original to target bits' positions according to expression (2). For the lines, all bits are pasted in their new estimated positions. However, not all columns of bits are concerned with vertical projection. The bits of high and middle orders of each pixel are the only ones concerned. This leads to an exchange of only these bits between the pixels of the same column. This is so because if all bits of pixels are used, the projection has a tendency to glide a whole line of pixels the same way. The vertical projection becomes then a simple shift of the image. Moreover, the bits of high order are the most probable to make a significant change on pixels even though this depends on the content of each one.

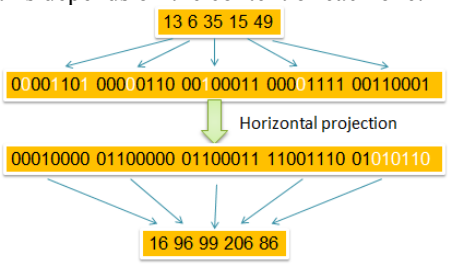


Figure 5: A sample of horizontal projection resulting in the bits shifted to the left (for this case) and some ones (white) reintroduced from the right.

The next step after image projection is to fill the created holes with the delayed bits, i.e. P_{out} and/or P_{rep}. To do so, each bit is reintroduced into the opposite half part from which it does not originate and in an opposite order of its stacking. In addition, P_{out} are reintroduced first in order to be stretched out in the image because they are initially contiguous. Then, P_{rep} are simply pasted into the remaining holes in a similar way of P_{out} (Figure 7).

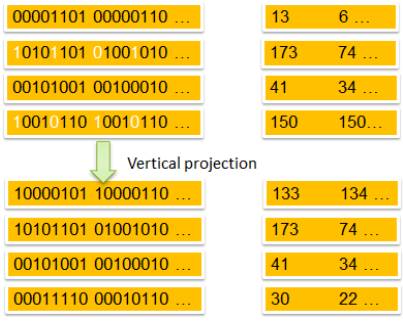


Figure 6: A sample of vertical projection resulting in the bits (3 and 7 of each pixel) shifted in vertical direction to the bottom and some ones (white) reintroduced from the top. The right column shows the corresponding decimal values.

To sample what has been explained, let us consider figures 5 and 6. When the transformation is applied in horizontal direction, some bits of each pixel either shift or migrate towards another one of the same line, and consequently, all pixels of each line change their values (figure 5). Likewise, when the projection is applied in vertical direction, the exchange of bits is done between the pixels' bits of one column (figure 6). In this case, only the bits of high and middle orders are projected as this is explained later.

The shuffling procedure of an image is, in general, based on repeating n times the cycle of new image projection and reinsertion of delayed bits. In fact, several repetitions of such process on the successively new obtained images are enough to get a complete cipher image. Furthermore, the algorithm combines several couples (θ, d) in several ways. It may apply the shuffling procedure alternatively in vertical and horizontal directions with different values, i.e. (θ_x, d_x) and (θ_y, d_y), at different steps with different number of each, and with different combinations for each color plane of the image.

(a) Original (b) Projected and filled

Figure 7: An original image (a) and its projected one (b) using $\theta = 1°$ and $d = 0.5$ for both horizontal and vertical directions.

3.2 Permutation and substitution of pixels

The power of the previously described process is that it results in both permutation and substitution of image's pixels (figures 5 and 6). In fact, as it is already explained, due to the projection process, the bits are displaced from their positions. At first glance, the consecutive pixel's bits are dragged together. This can be seen as pixels' movement resulting in a confusion procedure. From another point of view, pixels do not move really, they are the bits which do it. So, depending on θ, some bits of each pixel are going to move to another one. Thus, each pixel gets a new value; i.e. it is substituted or diffused. This is what happens mainly in vertical projection. Consequently, based on both points of view, the pixels' values can be considered changing when moving. This is what creates a simultaneous permutation and substitution of pixels within one operation without being discernible.

3.3 Key structure

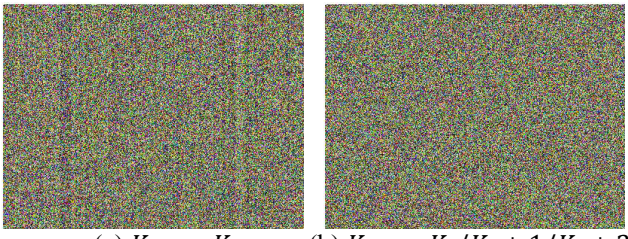
In this work, the parameters θ and d serve as parameters of encryption key. They are used in couple at any step of shuffling. Therefore, different forms of the key may be used, and can be as complex as needed. For this, the algorithm does not depend on any structure. It can be adapted easily to anyone. Considering our experimentations, the adopted structure is as follows:

$[nb_{iter}]$ $[v_i]$
$[\theta_{x_1}, \dots, \theta_{x_{v_i}}][d_{x_1}, \dots, d_{x_{v_i}}]$
$[h_i]$
$[\theta_{y_1}, \dots, \theta_{y_{h_i}}][d_{y_1}, \dots, d_{y_{h_i}}]$

The used structure implies a global number of iterations (nb_{iter}) applied to local couples vertically (v_i) and horizontally (h_i). Each of which is of the form (θ_{axis_i}, d_{axis_i}), where $axis = x$ or y.

As an example of its use, figure 8 shows the encryption of the image of figure (7(a)) with two different keys. The one of figure (8(b)) is a special case where each color plane is scrambled apart with different key values.

As it is clear, the used key is compounded of many parts having different sizes. For each angle, 1 bit is used for the integer part and 10 bits for the decimal part. Since the distance is a multiple of the line or column's length, it uses 4 bits for the integer part and 8 bits for the decimal part. In sum, each couple (θ, d) uses a total of 25 bits together with two bits used for the number of local couples (θ, d) in a given direction. The total number of global iterations uses 6 bits allowing the maximum of 64 iterations which might be highly paid in term of execution time. The size of the key is then 6+25(v_i+h_i) bits. At least, this size is 56 bits.



(a) $Key = K_1$ (b) $Key = K_1/K_1 + 1/K_1 + 2$

Figure 8: Scrambling of the image in figure 7(a). (a) is shuffled using $K_1 = [4,1,\frac{5\pi}{18},2.0,1,\frac{\pi}{3},1.6]$, but in (b) K_1 is used for only the color plane R. G and B color planes use K_1 where each couple (θ, d) is increased respectively by the values (1, 1) and (2, 2).

3.4 Algorithm complexity

The basis of the whole process is equation (2). Considering that $OV_o = OV_t = d.lc$, the expression becomes:

$$OP_t = \frac{d.lc.OP_o.cos\theta}{d.lc - OP_o.sin\theta} \quad (8)$$

In (8), two multiplications, one division, and one subtraction are uses. This is so, because $d.lc.cos\theta$ and $sin\theta$ are computed once at the beginning. So, taking $N \times N$-sized image, computing complexity of projection calculation for a key of (v_i+h_i) couples (θ, d) is $\frac{(v_i+Lh_i)(a+3b)}{2}N$, where a is the cost of a subtraction operation, b is the one of a multiplication or division operation, and L is the number of each pixel's bits. We get Lh_i because in horizontal direction the computing is related to bits, whereas in vertical one it is related to regular lines. The division by 2 is due to the consideration of (8) on only one half of the image.

In horizontal direction, the projection is applied on lines being N vectors. Thus, let p be the cost of changing one bit, the complexity is then LpN^2. In vertical direction, it is $2pN^2$ since only two bits of each pixel are concerned. For the whole algorithm, the complexity is then $\frac{(v_i+Lh_i)(a+3b)}{2}NL + h_i pLN^2 + v_i 2pN^2$ and with an iteration time n, it becomes:

$$\frac{(v_i + Lh_i)(a + 3b)}{2}N + np(h_i L + 2v_i)N^2 \quad (9)$$

For vertical direction processed the same way of horizontal one, we get:

$$\frac{(v_i + h_i)(a + 3b)}{2}LN + np(h_i + v_i)LN^2 \quad (10)$$

In its lowest level of security, this algorithm uses a key compounded of two couples (θ, d), one for each direction. In this case, $h_i = v_i = 1$ which makes the complexity being $(a + 3b)\left(\frac{L+1}{2}\right)N + np(L + 2)N^2$. Compared to the ones of the three compared algorithms in [12], this complexity is lower than the lowest one of Baker map algorithm which is $2(a + b)N^2 + n(a + b)N^2$. As can be seen, $\left(\frac{L+1}{2}\right) \ll N$ which makes the other coefficients negligible. Even if all bits of each pixel are considered in vertical direction, the complexity $(a + 3b)LN + 2npLN^2$ remains the lowest.

3.5 Reconstruction of the original image

The retrieval of the original image is based on a reverse process of its encryption. Knowing the values of encryption key, the reconstruction of the original image follows exactly the reverse steps. In other words, the algorithm piles up the operations which are then executed from the last one to the first one having its steps reversed for the following points:

– lines - columns. For instance, if for a given step of the encryption process the lines were scrambled first then the reconstruction should begin by the columns first,

– d and θ. This is the case when the key form uses different values for these parameters at the same iteration. For instance, if (d_1, θ_1) and (d_2, θ_2) are two couples of values used in this order for image encryption, the order (d_2, θ_2)-(d_1, θ_1) should be used during the decryption,

Considering P_{out} and P_{rep}, they are first identified through the initially created holes in the cipher-image. The reverse operation consists then in copying the bits at the positions of the holes from the encrypted image into their initial positions in the reconstructed image. This is done with respect to the reverse order of their initial copy during image encryption. This is evident for P_{out}. For P_{rep}, however, the algorithm should differentiate between the first bits which were directly projected and those which were delayed. So, the latter are the only ones that are retrieved from the holes. In sum, these operations are executed whenever they were done during the encryption process for either P_{out} or P_{rep}.

4 Key analysis

4.1 Key space

The numbers of significant values of θ and d together with their combining form have direct effects on the possible values of a key. These last are in close relation to the sensitivity of encryption to the key values. In the following, a presentation of the key space according to the used structure in our experiments then to its general form is given.

According to the chosen structure of the key, one can see that the key subspace of the couple (θ, d) is $2^{11}.2^{12} = 2^{23}$. For a maximum number of local different couples being 3, this value becomes $(2^{23})^3 = 2^{69}$. Since this is applied in horizontal and vertical directions with different values, the total number is $2^{69}.2^{69} = 2^{138}$. This is for a key size of 156 bits. If different keys are used for each color plane, the total number is then $2^{138}.2^{138}.2^{138} = 2^{414}$. Additionally, if different keys are used at different iteration times n, this value increases to become $(2^{138})^n$ or $(2^{414})^n$.

According to the general form of the key, let the number of values of θ be θ_v and those of d be d_v. The space values of a couple (θ, d) is $\theta_v \times d_v$. For a number of different couples used in a combining form of the key, this becomes $(\theta_v \times d_v)^{(h_i+v_i)}$. If different keys are used for each color plane, we get $(\theta_v \times d_v)^{3(h_i+v_i)}$. Moreover, if different keys are used in different iterations, then the key space becomes $(\theta_v \times d_v)^{3n(h_i+v_i)}$ which is very big. Furthermore, if high level of security is not required, d_v may be very big.

As can be seen, the key space increases with h_i, v_i, and n. However, not all possible values of θ and d are used for secure encryption. The next section explains the raisons. Hence, the choice of h_i, v_i, and n depends on the required security and complexity. So, for high level of security and when some values of θ and d are avoided, the values of h_i, v_i, and n should increase, and vice versa. Note that all next experiments use $h_i = v_i = 1$ and $n \leq 4$ which provide the lowest security level of key structure in order to show the power of this algorithm.

4.2 Key sensitivity

In order to specify the significant values of a key that lead to a given level of security against differential attack, some experiments have been done. One of them is the use of ciphertext difference rate (Cdr) defined in [12, 17] and adapted to our case for color images as:

$$Cdr = \frac{\sum_c \left(Diff_c(Y, Y_1) + Diff_c(Y, Y_2)\right)}{6NM} \quad (11)$$

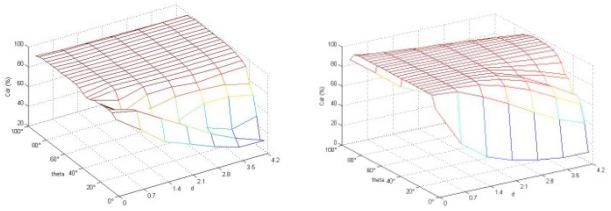
Whereas index c indicates one color plane R, G, or B and

$$Diff_c(A_c, B_c) = \sum_i \sum_j Difp_c\left(A_c(i,j), B_c(i,j)\right) \quad (12)$$

and

$$Difp_c\left(A_c(i,j), B_c(i,j)\right) = \begin{cases} 1 \text{ if } A_c(i,j) \neq B_c(i,j) \\ 0 \text{ if } A_c(i,j) = B_c(i,j) \end{cases} \quad (13)$$

Y is the cipher image of size $N \times M$ under a key K, Y_1 under the key $K + \Delta K$, and Y_2 under the key $K - \Delta K$.

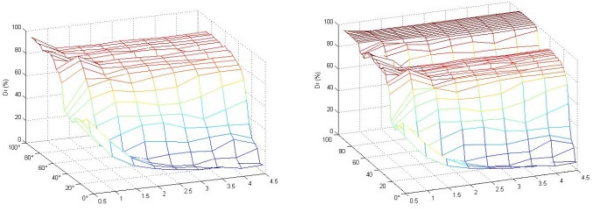


(a) (b)

Figure 9: Cdr computing based on a slight difference of θ using 3 iterations and (a) $\Delta\theta = 0.001°$ (b) $\Delta d = 0.01$.

In our tests, we used $\Delta\theta = 0.001°$ for θ (figure 9(a)) and $\Delta d = 0.01$ for d (figure 9(b)) separately. In these experiments, the structure of the very low security of the key is used; i.e. each key has only one and the same couple (θ, d) for each direction and for all color planes. Figure 9(a) shows that for $\theta \in [20°..89°]$, the security of the proposed cryptosystem is acceptable against brute-force attacks since $Cdr \geq 90\%$. Figure 9(b) shows again this power when $d \in [0.5..3]$ for approximately the whole θ range. Note that these results are obtained for only three iterations. This is as good as Cat map algorithm in [12] which is the best of the three presented algorithms according to this test.

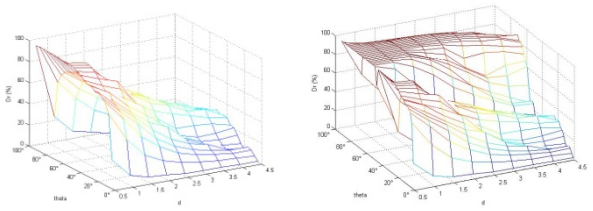
Consequently, for high level of security, it is advisable not to use all values of θ and d. In this case, $\theta \in [20°..89°]$, $d \in [0.5..3]$ and four iterations are the most advised. What is important in the algorithm is that for such high level of security, it does not require high cost. In addition, the key sensitivity is of high level since a difference of 1°/1000 for θ and 1/100 for d has considerable effects on the cipher image.



(a) (b)

Figure 10: Dr computing based on $\Delta\theta = (10^{-3})°$ using (a) n=1 and (b) n=4.

The previous results are supported by other conducted experiments which aim to specify the tolerance interval $\pm\Delta\theta$ and $\pm\Delta d$ of decryption sensitivity. These experiments have been conducted according to two ways. The first one is subjective. It consists in considering the opinions of some people whether they are able or not to identify the content of the decrypted image using an erroneous key. This shows that the values of $(10^{-3})°$ for θ and 10^{-2} for d are the limits of image identification if some particular values are avoided. In other words, $\pm 10^{-3}$ and $\pm 10^{-2}$ are the tolerance intervals in order for the use of a false key does not decrypt the image.



(a) (b)

Figure 11: Dr computing based on $\Delta d = 10^{-2}$ using (a) n=1 and (b) n=4.

Based on these results, the second tests are based on the computing of the difference between the original image and the decrypted one. For this, we propose the calculation of the decryption rate (Dr) as:

$$Dr = \frac{\sum_c Diff_c(Y, Y')}{3NM} \quad (14)$$

The function $Diff_c$ is defined in (12), Y is the original image, and Y' is the decrypted image using the key $K' = K \pm \Delta K_i$, $(i = \theta, d)$. K is the encryption key and ΔK_i =(10^{-3} or 10^{-2}) is the slight change of only one occurrence of one parameter of K.

These tests use the form of the key which is of the very low security level as in Cdr tests. The graphs of figures (10) and (11) show the obtained results. The used iteration time is either 1 or 4. As can be seen in all cases, Dr increases with the number of iteration time n.

The graph of figure 10 is obtained for $\Delta\theta = (10^{-3})°$ and $\Delta d = 0$, and the one of figure 11 for $\Delta\theta = 0°$ and $\Delta d = 10^{-2}$. What is noticeable for the graph of figure 10 is that for $\theta \geq 20°$, Dr remains very acceptable even when d increases, whereas in figure 11, it falls off quickly for big values of d. This leads us to conclude again that the decryption process is more sensitive to θ than to d.

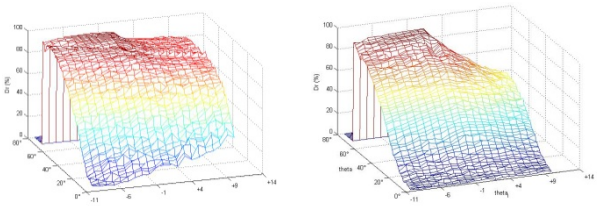
Throughout the conducted experiments, it can be concluded that small iterations (≤ 3) and small values of $\theta \in [1°..10°]$ are not advisable for high security if the used key is of low security level.

4.3 Effect of θ_l on encryption

To study the effect of θ_l on the encryption, Cdr and Dr are again considered. Since θ_l exists for each value of θ related to a specific distance d, we used to vary θ_l from 10° to 80°, compute the corresponding d, and evaluate Cdr and Dr with regard to all angles of the interval $\theta_{l_i} \pm j°$ $(i = 10°..80°,\ j = 1°..10°)$. For this purpose, expression (4) which links θ_l to d is used to determine each d_i related to a given θ_{l_i}. In this case, we get:

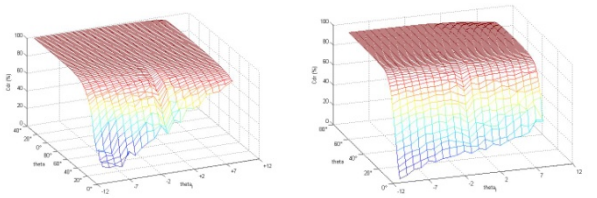
$$d = \frac{sin\theta_l}{2(1 - cos\theta_l)} \quad (15)$$

Through the graphs of figures 12 and 13, one can see that for good Dr and Cdr, the best values are obtained for $\theta < \theta_l$ and the quality decreases for $\theta > \theta_l$, even though those of Cdr are not so notable. In other words, the results of this cryptosystem are better when P_{out} are not empty and its efficiency decreases with the creation of an empty part but remains very acceptable. This is logic since P_{out} leads to more scrambling of bits than when an empty part is created. Moreover, the sensitivity is more important for $\theta < \theta_l$ than the one of $\theta > \theta_l$ except for some very small values of θ as has been already noticed. Note also that the graphs of figures 12 and 13 highlights again the sensitivity of the cryptosystem to θ than to d.



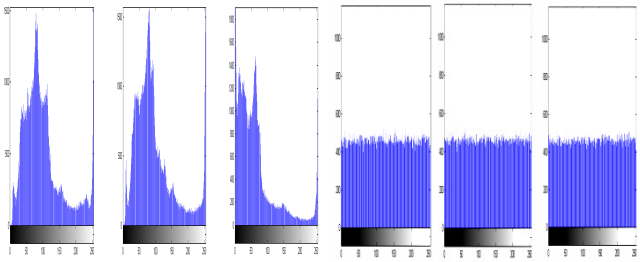
(a) (b)

Figure 12: Dr computing for every $\theta_l \in [10°..80°]$ using 3 iterations based on (a) $\Delta\theta = (10^{-3})°$ and (b) $\Delta d = 10^{-2}$.



(a) (b)

Figure 13: Cdr computing for every $\theta_l \in [10°..80°]$ using 3 iterations based on (a) $\Delta\theta = (10^{-3})°$ and (b) $\Delta d = 10^{-2}$.



(a) red (b) green (c) blue (d) red (e) green (f) blue

Figure 14: Colour histograms of the two previous images: (a), (b), and (c) of the original image (Figure 7(a)), and (d), (e), and (f) of its corresponding encrypted form (Figure 8(b)).

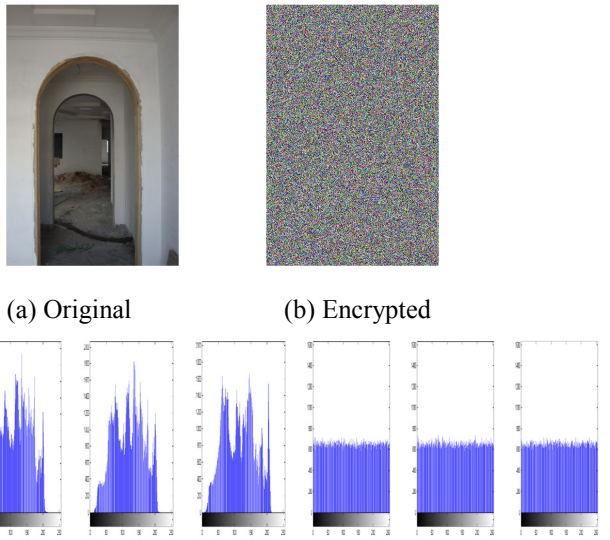
5 Resistance against other attacks

5.1 Statistical attack

Figure (8(b)) shows the result of the encryption process applied to the image of figure (7(a)), and their histograms are shown in figure 14. It is clear that the histograms of the two images, from statistical point of view, are significantly different. The ones of the encrypted image are flat and totally misleading, whereas those of the original image are curved.

Moreover, figures (15), (16), and (17) show other images which were used in our experiments, and on which this encryption algorithm is applied. These images

have been chosen based on the different shapes their color histograms have. It is clear that the histograms of the original images and encrypted ones are totally different. Even though the ones of the three images are of different forms, those of encrypted images are completely flat.

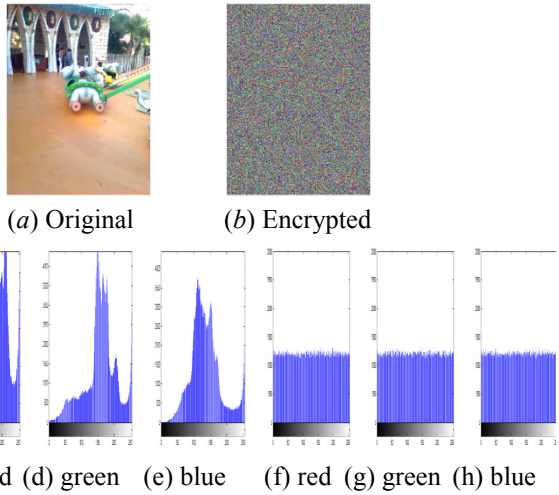


(a) Original (b) Encrypted

(c) red (d) green (e) blue (f) red (g) green (h) blue

Figure 15: Histograms of an original image (a) and its encrypted form (b) where the histograms of the former ((c), (d), and (e)) have several peaks concentrated in the middle.

5.2 Select plaintext attack



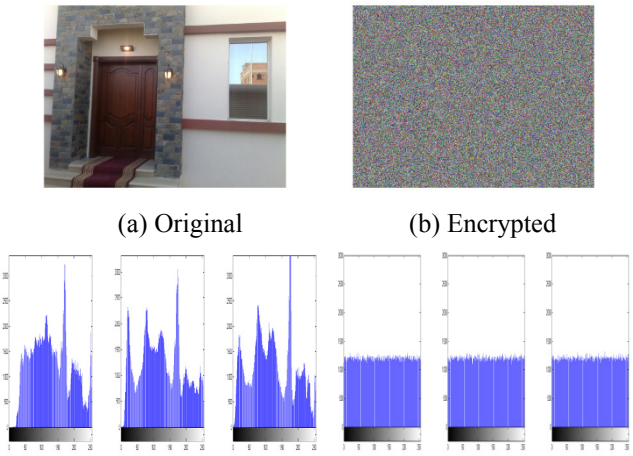
(*a*) Original (*b*) Encrypted

(c) red (d) green (e) blue (f) red (g) green (h) blue

Figure 16: Histograms of an original image (a) and its encrypted form (b) where the histograms of the former ((a), (c), and (d)) have approximately one peak.

Another feature the algorithm should resist is select plaintext attack. It is commonly used to analyze plaintext sensitivity; i.e. the effect of the algorithm on parts with little differences. To test such feature, we use pixel change rate (Pcr) defined in [12] and adapted to our case as:

$$Pcr = \frac{\sum_c Diff_c(Y, Y'')}{3NM} \quad (16)$$



(a) Original (b) Encrypted

c) red (d) green (e) blue (f) red (g) green (h) blue

Figure 17: Histograms of an original image (a) and its encrypted form (b) where the histograms of the former ((a), (c), and (d)) have many peaks spread out along the dynamic range.

In this case, Y and Y'' are encryption results using the same key of images I and $(I + \Delta I)$. ΔI is a slight change in one bit for instance; i.e. an image and its transformed one having one bit different are ciphered. The graph in figure 18 shows that this algorithm maintains Pcr at a stable level depending on the initial change. This is explained by the fact that initial bits of the original images are the same about $\frac{L-1}{L}$ for one pixel different. Since there are no significant differences, the two images evolve the same way during encryption. Even though Pcr does not reach 100%, it remains so important being $\geq 64\%$ in all cases.

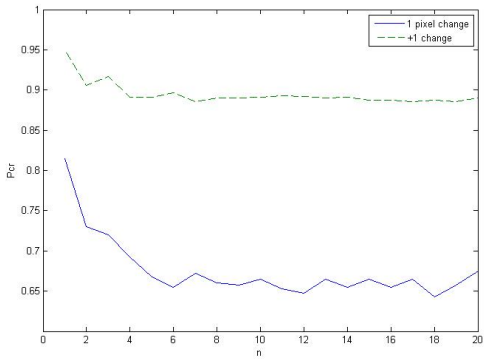


Figure 18: Pcr graph when a slight change happens to each pixel of an image.

6 Conclusion

A new algorithm of image encryption based on a geometrical transform is presented. The algorithm projects an original image on a plane perpendicular to the direction of a given view's position. It considers the image as rows and columns of bits. So, the projection of each bit is done on the position where its light ray

intersects the perpendicular plane to the direction of the view. As a result, a cipher process takes place. This process is repeated several times to ensure a given level of security.

The strength of the algorithm depends mainly on the structure of the used key. The core of the algorithm is based on two parameters: the angle of view and the distance of the viewer from the image. They are used in couples, and combined in many ways so that the combination itself becomes a key. Hence, the power of this algorithm is that the combinations depend on the genuineness of the user.

The efficiency of this algorithm is demonstrated through several tests. Its key sensitivity from many points of view is about $(10^{-3})°$ for the angle and 10^{-2} for the distance. Moreover, its resistance to many common attacks is proved through many graphs of the very low level of security provided by both key structure and values.

Further extensions for this work may be made in different directions. One of them is the study of the algorithm resistance to different attacks apart from the security considerations. For example, to what extent an encrypted image may resist to JPEG compression in order to reconstruct the original one since the initial bits are not explicitly changed. The structure of the key is also important. One possible improvement is to make a formal study of it. Another important study is about the general form of the projection. It may be part of the key.

References

[1] B. Nini and C. Melloul (2011), Pixel Permutation of a Color Image Based on a Projection from a Rotated View. *JDCTA: International Journal of Digital Content Technology and its Applications*, Vol. 5, N° 4, pp. 302-312.

[2] Chengqing Li, Kwok-Tung Lo (2009). Security analysis of a binary image permutation scheme based on Logistic map. http://arxiv.org/pdf /0912.1918

[3] Chengqing Li (2008). On the Security of a Class of Image Encryption Scheme. *Int. Symposium on Circuits and Systems,* IEEE, pp. 3290-3293.

[4] Ercan Solak, Cahit Cokal and Olcay Taner Yildiz (2010). Cryptanalysis of fridrich's chaotic image encryption. *International Journal of Bifurcation and Chaos*, *DOI: 10.1142/S0218127410026563*, Vol. 20, n° 5, pp. 1405–1413.

[5] Daemen, J., and Rijmen, V. (2002). The Design of Rijndael: AES – The Advanced Encryption Standard. Springer,.

[6] G. Ye (2009). Image scrambling encryption algorithm of pixel bit based on chaos map. *Pattern Recognition Letters, DOI: 10.1016/j. patrec. 11.008*.

[7] Haojiang Gao, Yisheng Zhang, Shuyun Liang, and Dequn Li (2006). A new chaotic algorithm for image encryption. *Chaos, Solitons and Fractals 29*, pp. 393-399.

[8] Jianbing Shen, Xiaogang Jin, and Chuan Zhou (2005). A Color Image Encryption Algorithm Based on Magic Cube Transformation and Modular Arithmetic Operation. *PCM, Part II, LNCS,* Y.S. Ho and H.J. Kim (Eds.), 3768, pp. 270-280.

[9] John Vreugdenhil, Kane Iverson, and Raj Katti (2009). Image Encyption Using Dynamic Shuffling and XORing Processes. *IEEE International Symposium on Circuits and Systems*, ISCAS, pp. 734–737.

[10] Li C., Li S., Zhang D., and Chen G (2004). Cryptanalysis of a Chaotic Neural Network Based Multimedia Encryption Scheme. *Advances in Multimedia Information Processing – PCM Proceedings,* Part III, LNCS, Vol. 3333, pp. 418-425.

[11] Sapiee Jamel, Tutut Herawan, and Mustafa Mat Deris (2010). A Cryptographic Algorithm Based on Hybrid Cubes, *ICCSA*, Part IV, LNCS 6019, pp. 175-187.

[12] Shiguo Lian, Jinsheng Sun, Zhiquan Wang (2005). Security Analysis of A Chaos-based Image Encryption Algorithm. *Physica A: Statistical and Theoretical Physics, Elsevier*, 351(2-4): pp. 645-661.

[13] Shujun Li, Chengqing Li, Guanrong Chen, and Kwok-Tung Lo (2008). Cryptanalysis of the RCES/RSES image encryption scheme. *The Journal of Systems and Software*, *DOI:10.1016/j.jss.2007.07.037*, pp. 1130–1143.

[14] Trenkler M. (2005). An Algorithm for making Magic Cubes, *The Π ME Journal*, vol. 12, n°2, pp. 105-106.

[15] Wu, S., Zhang, Y., and Jing, X. (2005). A Novel Encryption Algorithm based on Shifting and Exchanging Rule of Bi-Column Bi-row Circular Queue. *International Conference on Computer Science and Software Engineering*. IEEE, Los Alamitos.

[16] Zhang Linhua, Liao Xiaofeng, and Wang Xuebing (2005). An image encryption approach based on chaotic maps, *Chaos, Solitons and Fractals 24*, pp. 759-765,

[17] Shiguo Lian, Jinsheng Sun, Zhiquan Wang (2005). A Block Cipher Based on a Suitable Use of the Chaotic Standard Map. *International Journal of Chaos, Solitons and Fractals, Elsevier*, 26(1): 117-129.

[18] Yaobin Mao, Guanrong Chen and Shiguo Lian (2004). A Novel Fast Image Encryption Scheme Based on the 3D Chaotic Baker Map, *International Journal of Bifurcation and Chaos, World Scientific Publishing*, vol. 14, no. 10, pp. 3613-3624.

[19] Komal D Patel, Sonal Belani (2011). Image encryption using different techniques: A review. *International Journal of Emerging Technology and Advanced Engeneering,* vol. 1 no. 1, pp. 30-34.

An ASM-based Model for Grid Job Management

Alessandro Bianchi, Luciano Manelli and Sebastiano Pizzutilo
Department of Informatics, University of Bari, via Orabona, 4, 70125 - Bari - Italy
E-mail: alessandro.bianchi@uniba.it, luciano.manelli@gmail.com, sebastiano.pizzutilo@uniba.it

Keywords: grid systems, OGSA services, asynchronous distributed Abstract State Machine.

Received: July 11, 2012

Job Execution Management Services in Grid systems are generally implemented using specific Grid middleware, and they are considered very critical for the success of the entire system. In order to better manage complexity and criticality, literature suggests the use of robust formal models to describe and analyze these services. This paper abstracts strategic services in Grid Systems, proposes an Abstract State Machine-based model to design them, and implements them by the coreASM tool. The obtained results lead to consider the usage of Abstract State Machine models as a concrete control appliance for Grid systems.

Povzetek: Avtorji predlagajo novo orodje coreASM, s katerim upravljajo storitve mreže.

1 Introduction

A Grid system is built to allow researchers in different domains to use multiple heterogeneous resources for problem solving in high performance computational infrastructure. It provides efficient use of multiple machines for executing jobs, in particular for parallel applications, which can distribute the parallel units of their work across the allocated resources through parallel execution. To this end, a Grid system is defined as a set of independent, distributed and cooperating computational units, which are able to achieve a particular computational goal in dynamic multi-institutional Virtual Organizations [1], [2].

Several organizations have proposed standards for requirements, architecture, specification and services for Grid systems. Among them, the Globus Project (now Globus Alliance [3]) proposed the Open Grid Service Architecture (OGSA) in 2006: it provides a set of important requirements and considers several services needed to support Grid systems and applications [4]. In particular, the requirements for *Execution Management Services* (EMSs) deal with managing the execution of jobs during their lifetimes by searching remote resources for scheduling and executing jobs, and by returning output to users.

1.1 Motivation

Grid systems represent fundamental assets in large-scale scientific and engineering research, but several difficulties are encountered in building reliable Grid applications, mainly because of the high complexity of these systems, [24], [25], and because of the lack of proper conceptual framework and supporting tools [26].

These issues are highlighted in [40], where authors state the huge gap between the proposed architectures usually expressed by informal diagrams, and the existing implementations, so concluding that "a good conceptual reference model for grid is missing".

The lack of precise specification heavily impacts EMSs that are generally implemented using specific middleware. Practically, Grid middleware supports the development of service-oriented computing applications and provides services with the aim to lead users to access remote distributed resources and run jobs on them. Different organizations developed different Grid middleware, sometimes implementing the same proposal. For example, two of the most popular middleware are *Globus Toolkit* [3] and *gLite* [5], both obtained by OGSA proposal and both used in several Grid applications. Unfortunately, a uniform coordination of different software is difficult or impossible across two different middleware, so, in the example above jobs originated on Globus Toolkit cannot be forwarded to gLite. According to [39], the lack of this uniform coordination results in different, often incompatibles interface between middleware and services, and drives to non-interoperable "Grid islands" that cannot share resources.

So, a challenge in realizing efficient Grid systems is the development of a specific middleware for the management and the execution of jobs in different environments. The difficulty increases when different Grid architectures are taken into account. According to [20], the need to give clear, formal definition of Grid systems architecture and services is an urgent task that can help in solving these problems.

1.2 Purpose of the work

Our research contributes to execute this "urgent task" by proposing a formal model of Grid system services. Similarly to [40], we do not propose a new architecture, but a formal executable model that precisely describe logical modules in EMSs, with the aim to better understand and improve analysis of job EMSs. Thanks to the abstraction provided by formalism, the system is expressed focusing only the main requirements and not

considering specific implementation issues. In this way the model strictly defines the overall organization and specifies the services provided, without imposing formats for accessing services. Note that the present paper does not face the possible incompatibilities among existing resources, but this is one of the future directions of our research.

The preliminary study [6] examined the lifetime of a job in OGSA Grid architecture and described a simple model for its management and execution. The present paper moves from that lifetime for formally modeling *execution management* components through an approach based upon Abstract State Machine (ASM) [7], [34], and for implementing it using the *coreASM* tool [8], [9].

The *coreASM* provides an executable language and a tool environment for high-level design and experimental validation of ASM models by supporting execution of ASM specifications. It consists in a platform-independent engine implemented as a Java component. Moreover, a graphical interactive environment in form of a plug-in for Eclipse platform provides interactive visualization and control. Finally, the Control State Diagram editor (CSDe, another plug-in for Eclipse [38]) provides an abstract structure for a first simple design of the system in form of diagrams. In this way, our investigation enlarges knowledge and experience in formalizing Grid systems using ASM, and the obtained results represent a first step in the development of a benchmark for formally analyzing alternative implementation of Grid services and architectures.

Next section overviews the literature concerning formal models applied to Grid computing. Section three provides the background on ASM-based approach. Section four designs and models Grid job EMSs. Section five reports on *coreASM* model implementation and on its execution in some typical cases for validating the model. Conclusion and future work are discussed in Section six.

2 Related work

A rich literature deals with the application of formal methods to the design of Grid systems architecture and services, to their verification and validation, and to the analysis of the most critical properties, which can heavily affect the efficiency of provided services. Examples are represented by π-calculus [35], used in [36] for formally specifying dynamic binding and interactive concurrency of Grid services, and by Z notation [28] and Hoare's Communication Sequential Processes - CSP [29], both adopted in [27] for modeling an identity management architecture for accessing Grid systems. Note that these works adopt the formalism for facing just one specific issue, and they do not generalize the approach to the entire system: they are optimal solutions to specific problems, but do not represent a general-purpose model for EMSs. Conversely, our research is aimed at providing a formal framework for specifying Grid services: in fact, in this work we focus on the Job EMS, but the approach we adopt can be easily extended form modelling the entire system.

Petri nets are adopted in researches with wider purposes. For example, composition of applications in a Grid is orchestrated into a unique Petri net-based workflow in [30]. Even though this paper provides high-level access to Grid services, it operates only on one middleware, and interoperability among different middleware is not taken into account.

More general approaches are presented in [31], [32], [10], and [40]. Guan and colleagues ([31]) propose the design and prototype implementation of a scientific Grid infrastructure using a Petri-net-based interface; OGSA resource management and task scheduling is extended by Liu and colleagues ([32]) using Timed Petri nets for allowing services definition; a Colored Petri net-based model is depicted by Zhao and colleagues ([10]) for studying job scheduling problems; Colored Petri nets are also used by van der Aalst and colleagues ([40]) for specifying and validating a reference model for Grid architectures.

All these examples show that Petri nets are effective in designing and analyzing the workflow structure and services provided by the Grid and for studying typical properties of the system, e.g. liveness. Unfortunately, Petri nets present some disadvantages. Storrle and Hausmann [11] summarize Petri nets problems in modeling behavior of exceptions, streaming and traverse-to-completion, and they discuss problems arising when trying to analyses these advanced features. Sarstedt and Guttmann [12] underline that Petri nets are not adequate and intuitive to describe the semantics of many system diagrams, also due to the lack of a unified formalism. Moreover, Eshuis and Wieringa [37] show that Petri nets are suitable for modeling closed and active systems rather than open and reactive/proactive systems, like Grid. Analogously, Atlas and colleagues argue that PN-based approaches are fairly rigid and are not suitable for dynamic environments [41].

For these reasons, we consider ASM approach [34] more suitable for modeling Grid architecture. This choice is justified by different issues. Several similarities exist between Petri nets and ASM and [7] shows that the run in a Petri net can be expressed by ASM rules, in particular it emphasizes the capability of asynchronous distributed ASMs in modeling the distributed nature of Petri nets. The capability of ASM in describing agent based models allows us taking into account the entire system in both its static and dynamic aspects, through a set of ASMs implementing an asynchronous distributed ASM.

Resuming, we applied ASM approach considering the advantages it provides under three different points of view. When the model expressivity is considered, a rich literature (e.g. [9], [11], [13]) agrees that ASMs show versatility in modeling complex architectures and they have excellent capabilities to capture the behavioral semantics of complex, dynamic, open, and reactive systems, like Grid, where several different processes occur, often with the need to properly model exceptions, streaming and traverse-to-completion. Secondly, considering software engineering development issues, it is worth noting that, starting from the ASM formalism, a

development process has been defined and successfully applied in several complex domains, e.g. telecommunication, programming languages, control systems, and so on [7]. Finally, considering the implementation point of view, the lack of specific environments for translating PN models in executable code is overcome by using an ASM-based approach thanks to tools like AsmL [33] and *coreASM* [8], [9], [22]. Both AsmL and *coreASM* are high-level executable languages based on ASM: the main difference is that the former is integrated with Microsoft .NET platform, the latter with the Eclipse platform.

The existing literature, which applies ASM to Grid systems, encourages our choice. Several papers use ASM to model communication systems behavior and to control their management. A general, abstract communication model for studying message-based communication networks in distributed systems is presented in [14]: the model is implemented and tested using *asmL* environment.

Lemcke and Friesen propose in [15] a composition algorithm of web services defining the execution of business processes and orchestrations by providing ASM representations for these processes and executing them with *coreASM* . Lamch and Wyrzykowski [16] develop a software environment in *asmL* implementing a hybrid approach for integration of a formal model and existing middleware components: the model-based testing approach is useful for investigating properties of specification of Grid middleware using ASM. Both these papers focus on integration issues, but they do not provide a comprehensive view of Grid systems.

In [17], [18], authors propose a semantic model for Grid systems and for traditional distributed systems describing differences between the two systems. They used ASM at a very high level for describing resources and users belong to a Virtual Organization (VO) without developing any tools for simulation. In a different way, [19] analyzes the similarities between distributed and Grid systems, the characteristics and the requirements of Grid systems and programming models. They develop an ASM model and study the validity in Grid environments. Finally, in [20] Zou and others propose a general framework for Grids and model the virtual organization based on ASM, considering quality issues for the user requirements. With respect to these papers, our work models the system at a lower abstraction level and validates its implementation.

In [6], a preliminary, monolithic ASM was built to model the standard mechanisms defined in OGSA for job EMSs. That paper informally details the general description of job EMSs, provides an overview of the ASM-based model, and statically analyzes it. The present work improves that paper, mainly with respect to two issues: firstly, the model is structured in a number of agents, each modeled by an ASM, so obtaining a more realistic Distributed ASM. Secondly, the model is implemented using *coreASM*, and dynamically validated in some typical scenarios.

3 Background on ASM

Abstract State Machine, ASM for short, is a powerful computational model [34], which has been successfully applied in several cases for modeling critical, complex systems, both in industry and in academia: wide discussion about ASM application is in [7]. It simulates every algorithm's behavior through a step-by-step way: each step computes a set of updates with given transition rules. After the completion of a step, all updates are committed simultaneously.

The concept of *abstract state* in ASM extends the usual notion of *state* occurring in Finite State Machines: it is an arbitrarily simple or complex structure, i.e. a collection of domains, with arbitrary functions and relations defined on them. On the other hand, *rules* are basically nested if-then-else clauses with a set of functions in their bodies. In each state, all conditions guarding rules are checked, that is all rules whose conditions are evaluated to true are simultaneously executed, so determining the state transition. All the parameters are stored in a set of so called *locations* and at each time the particular configuration of parameters values determines the current state of the ASM.

The transition from one state to another is described through a set of formulas, in the form:

$$\{\text{if } condition_i \text{ then } updates_i\}_{\ i=1,..,n}$$

where each $condition_i$ (the guard of the i-th rule) is an arbitrary first-order formula, whose interpretation can be true or false, and each $updates_i$ is a finite set of assignments

$$f(t_1,\ldots,t_n) = t$$

whose execution is to be understood as changing the value of the function f at the given parameters t_i, that leads to a new value of the parameter t. The parameters are stored in a set of *locations* and the configuration of parameter values at each step determines the current *state* of the ASM.

For the unambiguous determination of a next state, it is necessary that updates are *consistent*. An update set is consistent if it contains no pair of updates which assign different values to the same location; otherwise, the update set is inconsistent.

The concepts related to modeling monolithic systems through ASMs can be extended to distributed systems. The case of job management in Grid systems is a typical case of *asynchronous distributed systems*. It is *distributed* because its overall behavior is the composition of different, independent, remote elements, each operating on its own. It is *asynchronous* because all involved logical components operate and communicate concurrently, each according to its internal behavior. So, job EMSs in Grid can be modeled by a *Distributed Asynchronous ASM - asyncASM* [7], [21].

Essentially, *asyncASM*s generalize simple ASMs to an arbitrary finite number of independent agents [23], each executing an ASM in its own local state. Formally, an *asyncASM* is given by a family of pairs *(a, ASM(a))* of pairwise different agents, elements of a possibly dynamic finite set *Agent*, each executing its ASM, *ASM(a)*. In this sense, each agent *a* executes its own program, operating

on its own states, so determining a partial view of the entire system. The relation between global and local states is supported by the reserved keyword *self*, used to denote the specific agent executing a rule, and to store information relevant to itself. A new agent can be introduced into the *asyncASM* at any time by extending the set *Agent*.

An ASM-based process for developing complex systems is presented in [7]: it allows capturing system model requirements and refining them through intermediate models to any desired level of detail in a validated and verifiable code. In the present work, modeling and implementation activities have been carried out with the support of the *coreASM* framework [8], [9], [22]. It follows mathematical definition of ASMs and inherits several typical features of the ASM modeling. Its main purpose is to make ASM-based models executable. To this end, the framework includes some language constructs aimed at making easy the development, as, for instance, *forAll*, which allows executing all rules satisfying a given guard condition; *choose*, aimed at expressing non-determinism in the choice of a rule to execute when a condition is satisfied; *seqblock/endseqblock* are the delimiters of block, whose rules must be executed sequentially; *par/endpar* are the delimiters of block, whose rules must be executed concurrently; *enqueue/dequeue* are the operators for adding / removing elements to a queue.

4 Modeling the Grid job EMSs

In order to model the job EMSs, firstly the Grid capabilities and features are informally described, then they are abstracted and formally defined, finally the *asyncASM* is created in the *coreASM* environment [8].

4.1 Informal description of a Grid system

4.1.1 Services

The OGSA standard describes requirements (interoperability and resource sharing, optimization, quality of service, job execution, data services, security, scalability and extensibility), and considers six important independent services that implement such requirements and are needed to support Grid systems and applications [4]:

- ***Execution Management Services*** manage jobs to completion: they concern job submission, description instantiating, scheduling, and provisioning resources (e.g. RAM, disk, CPU, etc.).
- ***Data Services*** focus on the management, access and update of data resources (e.g. files, streams, DBMS, etc…), and provide remote access facilities, replication services and managing of metadata.
- ***Resource Management Services*** manage physical and logical resources.
- ***Security Services*** provide security-related policy, and manage access for cross-organizational users.
- ***Self-Management Services*** support the reducing of the cost and complexity of operating on an IT infrastructure. They provide self-configuration and self-optimization of system components (e.g. computers, networks and storage devices).
- ***Information Services*** concern the manipulation of information about applications, resources and services, support reliability, security, and performance in a Grid. They provide registry, notification, monitoring, discovering, and logging.

The specific implementation of these services is delegated to Grid middleware. Practically, when a job is submitted, the middleware creates a job manager process for that job. The job manager manages the single job's lifetime, matches job requirements with the needed resources, and controls the relative allocation in different way for different Grid middleware. The allocation consists in assigning and queuing the job to local manager in resources. Note that in the following we only focus Execution Management Services.

4.1.2 Architecture

We assume a Grid is over three levels, operating on distributed heterogeneous resources, namely *Grid Application* level, *Middleware* level, *Resources Pool* level. Note that, this layered view is an abstraction of OGSA proposal, often used in literature, for instance in [40], and it fits our purposes.

The *Grid Application* is a higher-level structure built on top of the architecture that for our purposes is only aimed at partitioning each user job in *jobs* to submit to the lower level. In fact, since it is usually considered that a *job* is the smallest unit managed by a Grid environment, we assume a user submits one or more *User Jobs* (say UJ_1, UJ_2, … ,UJ_n), each composed by one or more *jobs* (say job_{11}, job_{12},…, job_{1k} for UJ_1; job_{21}, job_{22},…, job_{2h} for UJ_2;… job_{n1}, job_{n2}, …, job_{nm} for UJ_n). Note that possible interactions among jobs are managed by Grid Application level, but this issue is outside the purposes of present work.

The *Middleware level* implements fundamental functionality needed by Grid applications and required by OGSA. It interacts with the *Application level*, managing *jobs* until their completions and returning results to users. In this context the EMSs concern searching candidate resource and executing and managing *jobs* until end, so users can transparently execute their *UJ*-s on distributed resources. These tasks are critical because their incorrect execution can heavily affect the quality of provided services [24], so adoption of formal method is useful and sometimes mandatory.

The *Resources Pool level* is characterized by distributed and heterogeneous resources of a Grid. For the purposes of our work *a resource* is a logical entity with specific features needed to job execution, with its local workload and managed by a local resource manager.

For sake of simplicity, the architecture is summarized in figures 1 to 3: Figure 1 is an overview of the three levels: the box labeled "*Grid Application Level*" is outside the purposes of present work, so it is not further detailed. The boxes "*Grid Middleware Level*" and

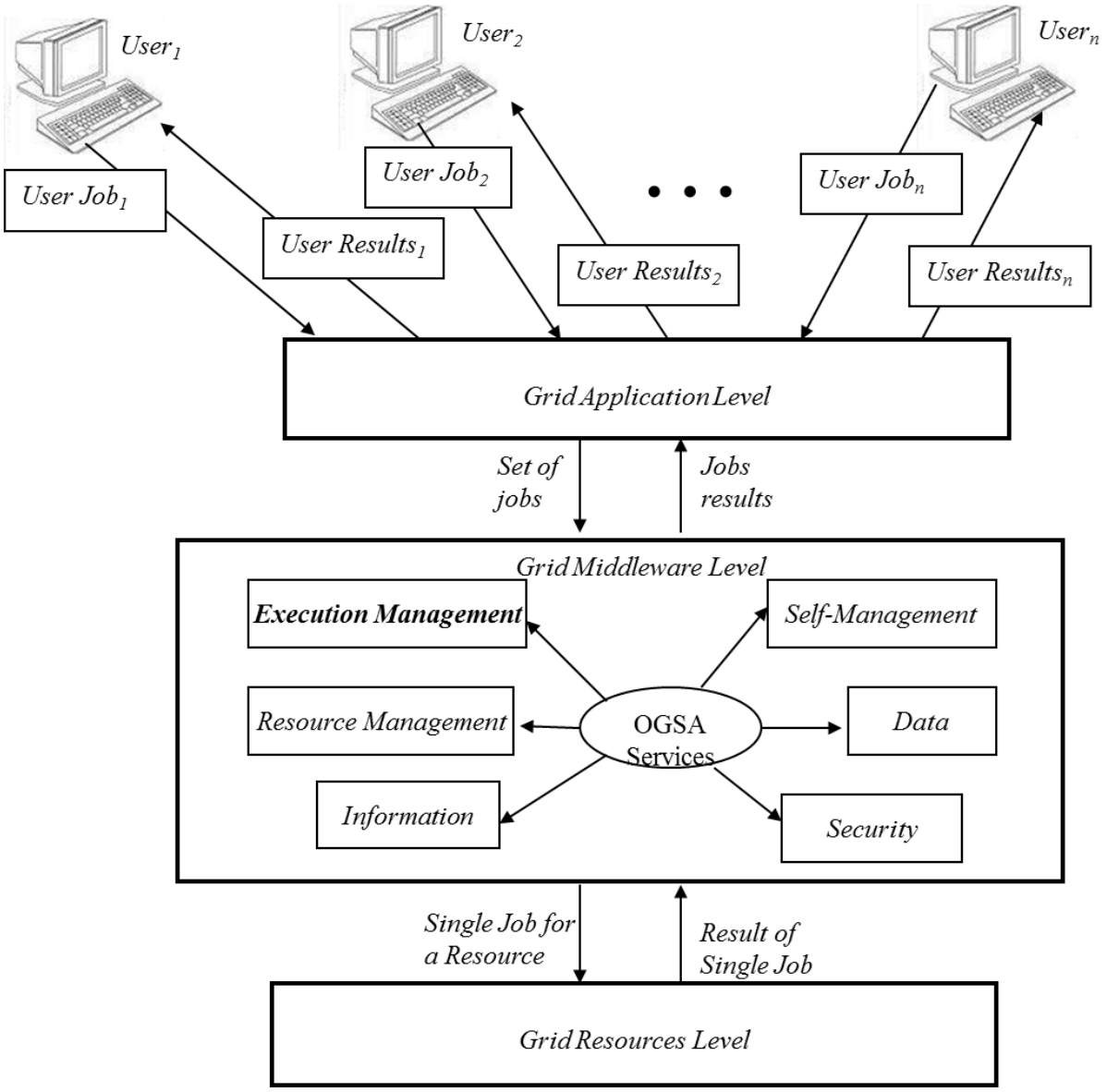


Figure 1: Architecture Overview.

"*Grid Resources Level*" are detailed in Figure 2 and Figure 3, respectively.

The *UJ*-s submitted by users are processed by the *Application level* and each of them is partitioned into composing *jobs*. Next, each *job* is sent to the *Grid Middleware level*, through a queue called *waitingJobs*, according to a First-In-First-Out (FIFO) policy. Moreover, the *Application level* receives the results of *jobs* executions by lower level and it re-composes them so that to obtain the results of UJs to send back to users.

A *job* must be executed on a resource that satisfies all its computational constraints. To this end, the EMSs chooses the proper resource in a shared pool, according to *job* performance requests, allocates the *job* on the selected resource for execution, and controls the *job* lifetime until completion.

In order to achieve these goals, we assume that Execution Management module is composed by two logical components: a *Dispatcher*, and a set of *Job Managers* (Figure 2), each accessed in mutual exclusion. The former schedules execution of each *job* to a *Job Manager*. Therefore, the role of the *Dispatcher* consists in the following activities: extracting the *job* at the top of the queue, searching for a free instance of a *Job Manager*, and activating the *Job Manager* instance on the submitted *job*.

After activation, the *Job Manager* becomes unavailable for other *jobs*. Then, it searches a suitable *Resource* in the *Resource Pool level*, matching the required *job* performance. If a resource is found, the Job Manager assigns it the *job*, waiting for control return, and it manages the lifetime and the state of the *job* until completion. In case of failure, an error is reported. Each *Job Manager* becomes again available for other *jobs* only after *job* completion.

A *Resource* in the *Resources pool* receives the *job* by the *Job Manager*, enqueues the *job* in the *local queue* and processes it, according to the position in the queue.

The set of resources is assumed to be fixed (from *1* to *R*), and during execution resources cannot be added or

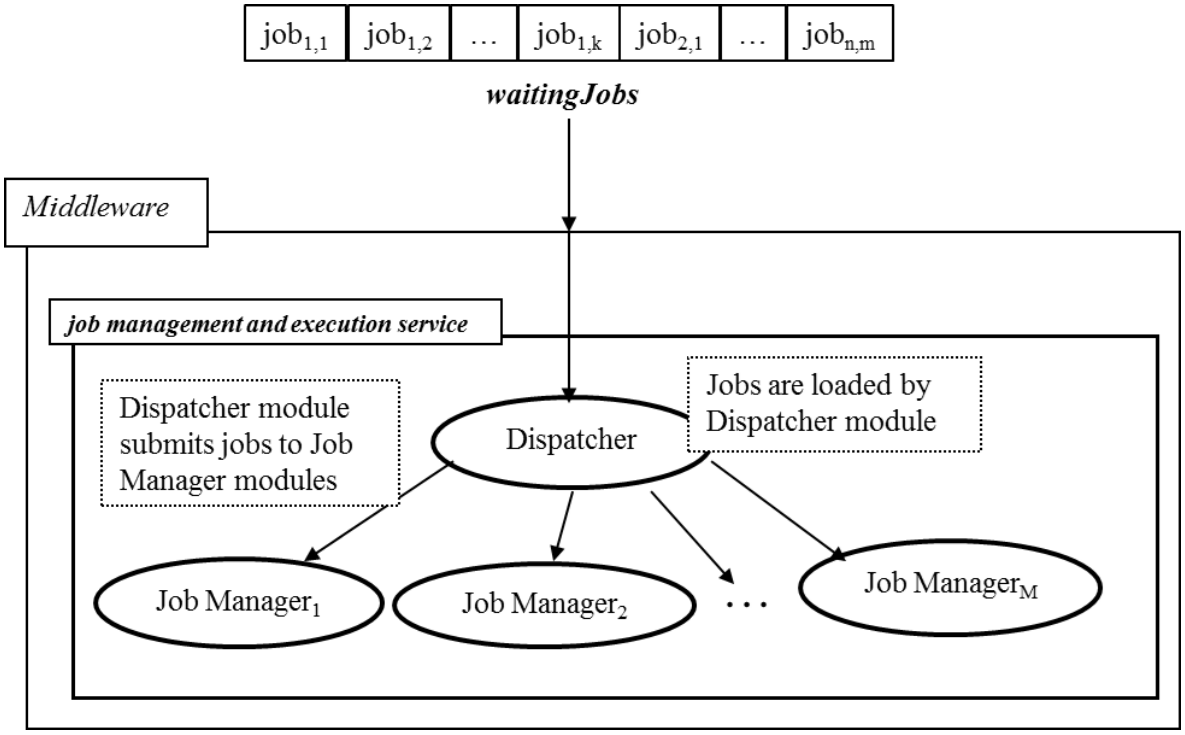


Figure 2: Overview of Grid Middleware Level.

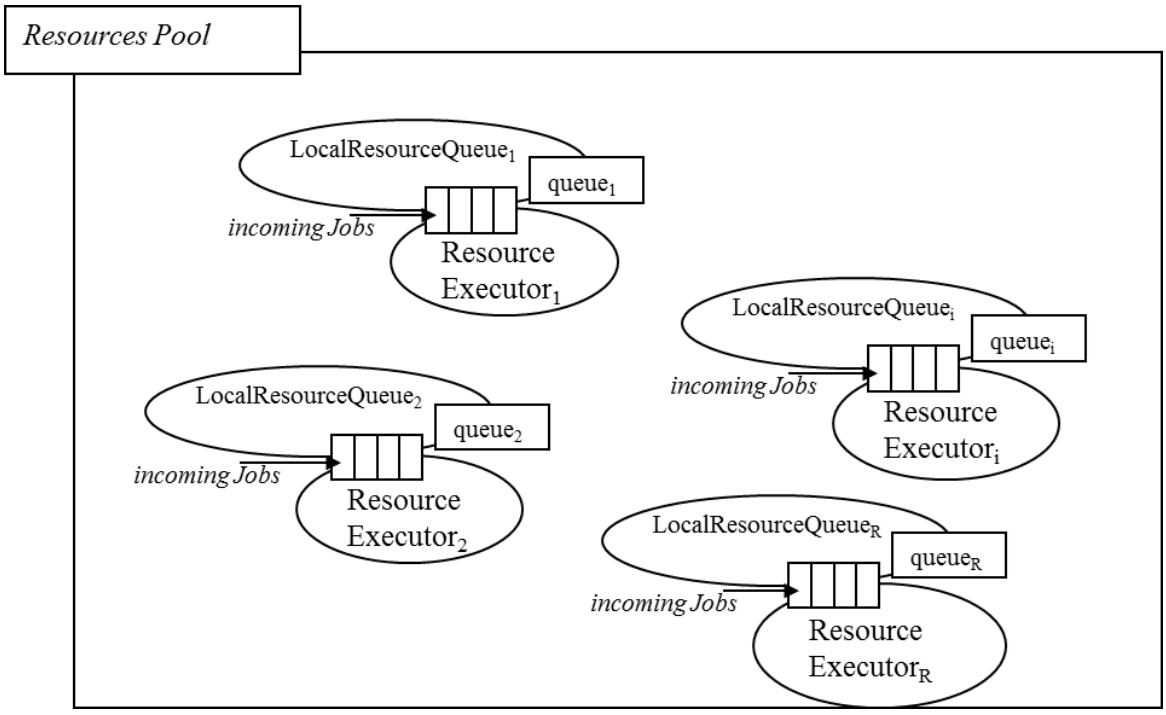


Figure 3: Representation of the Resource pool.

removed. For abstraction purposes, we suppose that a *Resource* computes *jobs* with a FIFO policy. If no resource in the pool is able to satisfy *job* requirements, the *job* fails due to a lack of available resources. Moreover, we assume that a *Resource* is composed by two logical components that work on the same *job*: a *ResourceLocalQueue* module, devoted to enqueuing the incoming *jobs*, and a *ResourceExecutor* module, for *jobs* execution.

If problems occur in resource during *job* execution, the *Job Manager* catches errors and stops the computation. When the execution completes, either successfully or unsuccessfully, middleware sends a message to the user application. In Figure 3 a representation of Resources pool is presented.

As a final remark, it is worth noting that users can cancel their *user jobs* (*UJ*-s) before completion; if so, all corresponding *jobs* are removed by the Grid system.

4.1.3 Abstraction

Informal description above can be abstracted in the following set of requirements:

Req.1 A Grid receives requests for *UJ-s* from clients; each *UJ* is decomposed in a set of atomic *jobs*, and each *job* is queued waiting for service.

Req.2 A *Dispatcher module* in middleware loads a *job*, and sends it to a specific *Job Manager module*.

Req.3 For each *job*, *Job Manager* finds the most adequate resource satisfying *job* requirements and runs it on that resource.

Req.4 That *Resource* enqueues and processes the *job*.

Req.5 If there are no failures, the *job* is completed with success.

Req.6 If errors occur the *job* is aborted.

Req.7 A user can cancel or remove a User Job, so cancelling/removing the corresponding *jobs*.

Req.8 At the end of the computation *Resource* is released.

Req.9 At the end of the computation the result is communicated to the end user.

Moreover, the model of system operations must be able to execute the following actions:

Act.1 The Dispatcher takes the first *job* in the *waitingJob* queue and sends it to the first available instance of Job Manager, if the system is ready.

Act.2 The system needs to find available resources necessary for the computation: if so, resources are reserved, else the *job* is immediately rejected.

Act.3 The *job* sent to a Resource is enqueued in its local queue and when possible the computation starts;

Act.4 The *job* can complete the computation with success and the system traces the result of the computation; if problems arise, the execution fails and the systems returns to the idle state.

Act.5 At the end of the computation, the system resets and returns in a state of inactivity and user is noticed about the result.

Act.6 The user can cancel UJ before execution or remove it before completion.

4.2 The ASM models

Modeling job management services in a Grid system is so reduced to modeling one *Dispatcher Agent*, a set of *JobManager Agents* and a pool of *ResourceExecutor Agents*, each with a *Resource Local Queue Agent*.

Note that the ASMs of the agents will be described separately, but all of them are a unique Distributed Asynchronous ASM - *asyncASM*.

In the resulting *asyncASM* there is one instance of the *Dispatcher ASM*, up to M instances of the *Job Manager ASM*, and R instances for both *Resource Local Queue* and *Resource Executor ASMs*. In real word, M is established by the capability of the middleware, and R depends on the actually available resources.

Figures 4 to 7 show the graphical view of the ASMs modeling Dispatcher, Job Manager, Resource Local Queue and Resource Executor agents, respectively. They are screenshots of the graphical representation of the ASMs, obtained by the ASM Control State Diagram editor (CSDe [38]). According to the usual notation of ASMs, circles represent *states*, diamonds represent *conditions* and boxes represent *rules*. Moreover, we added asterisks "*", "**", "***" to the screenshots for indicating the logical link between Dispatcher and Job Manager, between Job Manager and Resource Local Queue, and between Resource Local Queue and Resource Executor, respectively. For space reasons, in the following each ASM will be described in general and only some parts will be detailed.

After activation, Dispatcher ASM (Figure 4) goes in "IDLEDISP" state. When one or more *jobs* are in *waitingJobs* queue, the condition labeled "OneJob" evaluates to true, so the "LOAD" rule is activated. As a result, the *job* at the top of *waitingJobs* queue is loaded in Dispatcher. Then the ASM waits for the availability of at least one JobManager. The ASM stops waiting when the value of the location parameter *inactive* for a JobManager JM_i evaluates to true, for some *i*, i.e., JM_i is not currently processing, so it is available for executing that *job*.

Then the rule "ACTIVATION" is fired: this results in activating JobManager JM_i, so changing the value of the location parameter *inactive* to false, the state of the Dispatcher ASM evolves to "JOBSUBMITTED", and the Job Manager ASM (Figure 5) can start its execution. The computations of both Dispatcher ASM and Job Manager ASM then continue asynchronously: Dispatcher ASM checks if more *jobs* are waiting; meanwhile, Job Manager ASM processes the *job*.

After activation, if user does not cancel the *job*, the Job Manager ASM checks whether a proper resource, which satisfies *job* requirements exists in resource pool. To this end the condition labeled "exists resource in Resource" is evaluated. If the condition is not satisfied, then the *job* is rejected and the run stops, otherwise, "ACCEPT" rule is executed. The result of rule is assigning *job* execution to one of the available proper resources, selected in a non-deterministic way using the "choose" construct of *coreASM* language, and the location parameter *submitJobToResource* is set to the couple <identifier of chosen resource; identifier of job>.

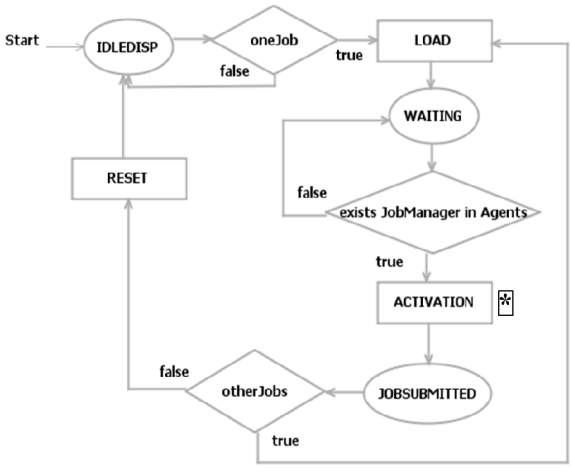


Figure 4: Representation of the Dispatcher ASM.

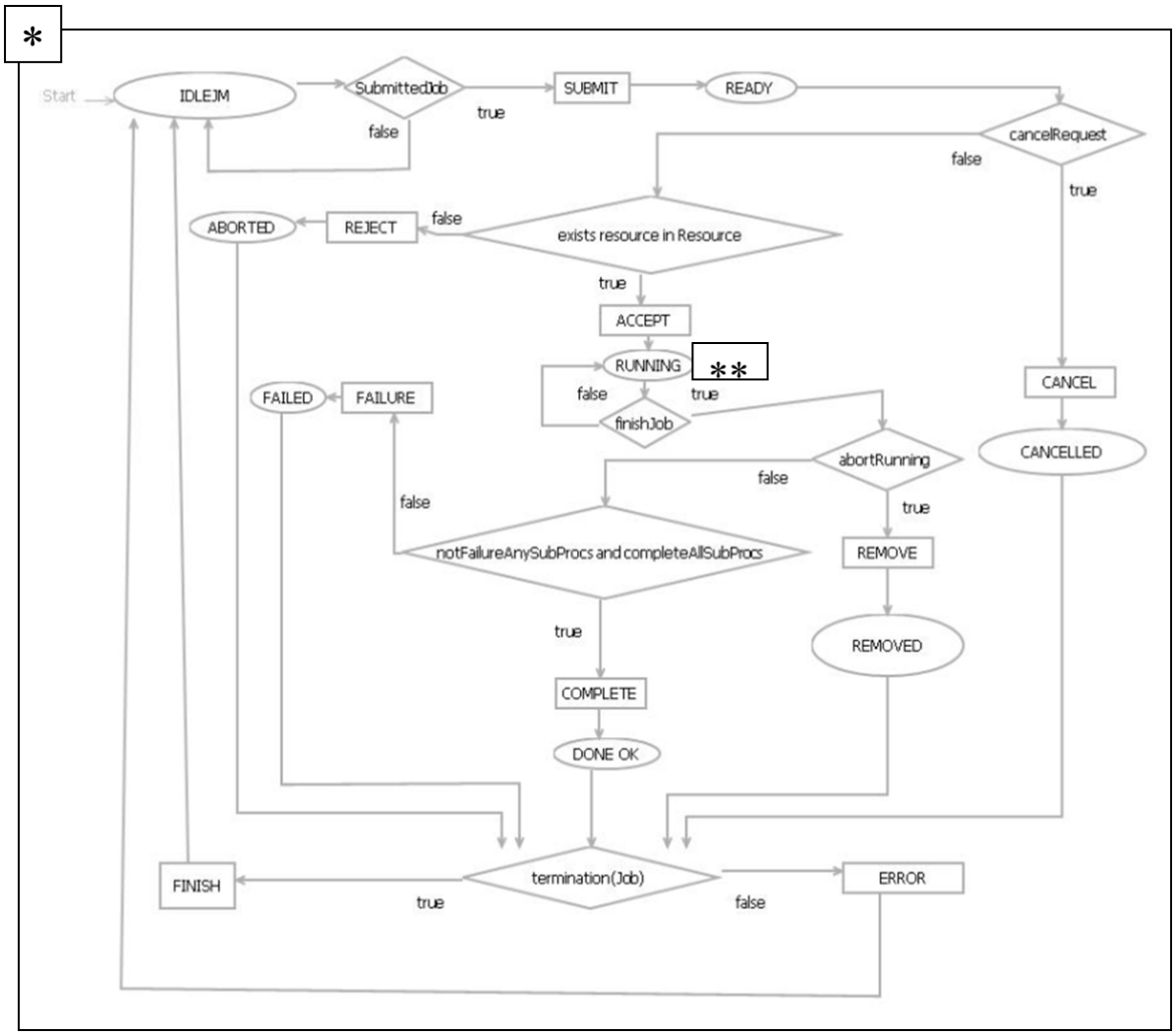


Figure 5: Representation of the Job Manager ASM.

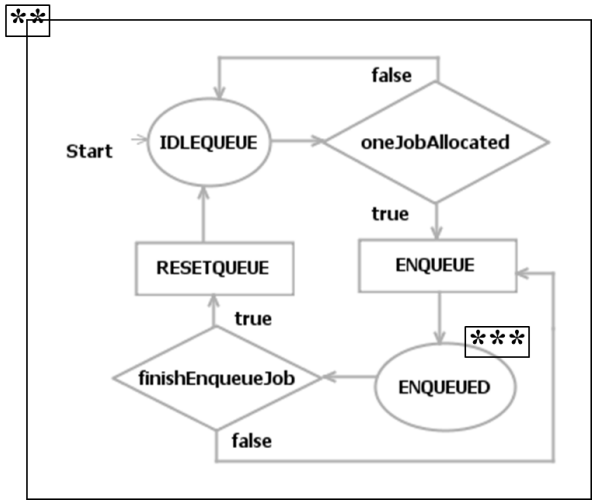


Figure 6: Representation of the Resource Local Queue ASM.

After execution of this rule, the Job Manager ASM evolves in "RUNNING" state, and the Resource Local Queue ASM (Figure 6) can start its execution, asynchronously with respect to Job Manager ASM.

The condition labeled "OneJobAllocated" in Resource Local Queue ASM is satisfied when the location parameter *submitJobToResource* establishes the relation between the resource and a *job*. Therefore, after activation of Resource Local Queue ASM, this condition evaluates to true and the rule "ENQUEUE" fires. This rule sets to true location parameter *jobAllocated*, so the condition labeled "OneJobInQueue" in the Resource Executor ASM (Figure 7) becomes true, and therefore the rule "SCHEDULE" can be executed. Next, Resource Local Queue ASM and Resource Executor ASM perform their computations asynchronously.

5 Model implementation and simulation

According to the ASM-based method for designing critical systems presented in [7], the modeled behavior is validated through simulation. So, after implementation, the ASMs shown above have been executed in some typical scenarios. Note that simulations are aimed at validating the main functionality, without the purpose to verify the correct behavior of the whole system.

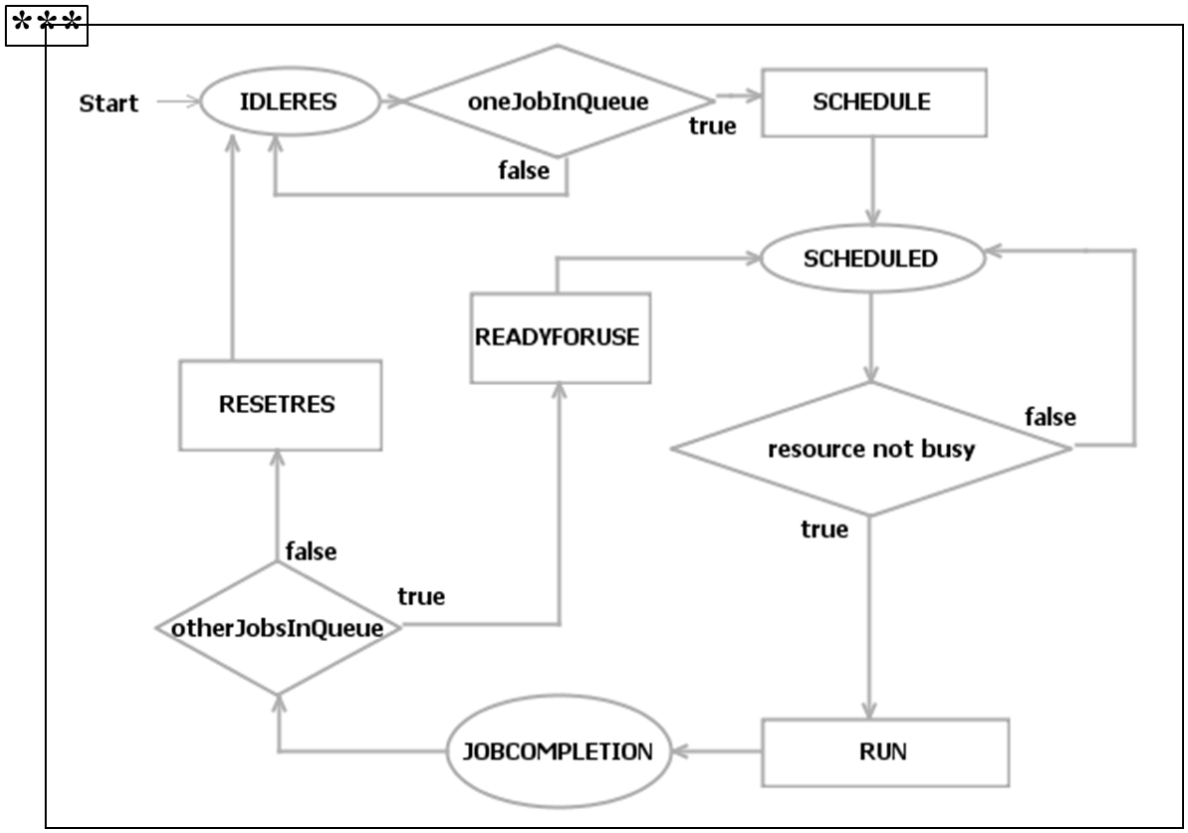


Figure 7: Representation of the Resource Executor ASM

5.1 Implementation

The executable code of the modeled ASMs, have been obtained through a 4-steps process. In the first step each ASM is edited using the Control State Diagram editor (CSDe [38]), an Eclipse plugin for creating and modifying ASMs and translating them into *coreASM* specifications. The second step is the production of the *coreASM* specification for every ASM. It is automatically executed by the CSDe. In the third step the four files obtained so far, one for each ASM, are merged in a unique *coreASM* file, which specifies the unique *asyncASM* modeling the system. Finally, the specific behavior of the obtained *asyncASM* is manually customized by the programmer by adding instructions for rules and conditions.

The entire process was executed in about four days by one person, without any previous experience in *coreASM* language. The step that consumed more time was the last, which required about three days. Note that the few time spent for designing the ASMs during the first step is due to the previous study of the model, which required much more effort.

The result of the process is one file, about 650 lines long: about 300 were automatically generated and about 350 manually produced.

5.2 Simulation setting

After implementation, the model has been simulated in five typical scenarios.

Scenario 1 is the standard ideal scenario, in which the Grid system is able to process all *jobs* submitted, all resources are available for all submitted *jobs*, and no user stops the submitted UJ. It is expected that each user receives the result concerning the computation of the submitted *job*.

In **Scenario 2** the system is able to process all *jobs* submitted, but constraints for some of them are not satisfied by any resource. Moreover, in this scenario the users do not stop computation before end. It is expected an error message to users that submitted unsatisfied *jobs*.

Scenario 3 is analogous to the first one, i.e. the system is able to process all submitted *jobs*, and all resources are available for all submitted *jobs*, but one of them (say, *stopped_job*) is stopped by an explicit user request, so a message confirming *job* deletion due to user action is expected.

Scenario 4 simulates the behavior of the Grid when the number of submitted *jobs* is greater than the total number of available Job Managers. Moreover, in this scenario the users do not stop the computations. It is expected that some *jobs* are not dispatched to a *Job Manager Agent* as well as they are submitted, because all *Job Manager Agents* are busy, so their execution must be delayed; nevertheless they should be correctly processed later.

Scenario 5 is aimed at validating the behavior of the Grid when only one resource satisfies the constraints for the submitted *jobs*. So, it is expected that if the users do not stop computations, all the submitted *jobs* are queued and they are processed according to arrival order.

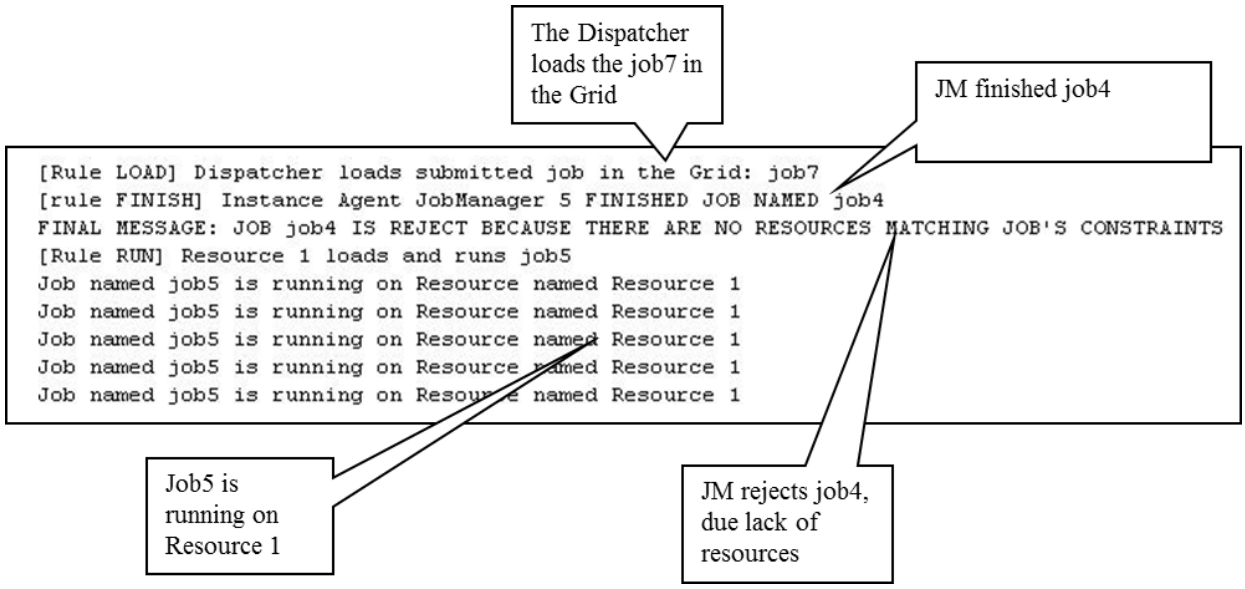


Figure 8: Console Output for Scenario 2.

For each scenario, ten simulations have been executed, and in all cases the Grid system is composed of one instance of Dispatcher ASM, 10 instances of Job Manager ASM, and 10 instances of both Resource Local Queue ASM and Resource Executor ASM. In scenarios 1, 2, 3, and 5 the number of *jobs* queued in the *waitingJob* is variable, but always lower than the total number of Job Managers. Instead, in scenario 4 the number of *jobs* is always greater.

5.3 Simulation execution

For each scenario, after setting the initial condition, the computation executed by the involved ASMs have been observed, looking at both the locations at each step, and the output shown in *coreASM* console. An example of the *coreASM* console output produced during execution of scenario 2 is in Figure 8.

For all scenarios, the *Dispatcher Agent* activates a *JobManager* instance for each submitted *job*, then the *JobManager Agent* searches for an adequate resource matching job constraints. During simulation of scenarios 1 and 3, all needed resources are found and reserved for *job* execution. Then, in scenario 1, each resource processes its *job* until completion and finally the expected message is sent back to each user. Instead, in scenario 3 the *JobManager* associated to the *stopped_job* ends its own execution when the guard condition *abortRunning* is encountered. In this case, *stopped_job* is removed by the *JobManager Agent*, and a message confirming deletion is correctly sent to the user.

Execution of scenario 2 shows that due to the lack of proper resources for some *job*s, they cannot be satisfied. In other words, some *JobManager Agents* fail in finding adequate resources, and their *job*s are rejected. The final messages for these *job*s correctly show the failure.

In scenario 4 some *job*s correctly wait for *Job* Manger availability; when a *Job Manager Agent* becomes ready, it accepts the *job* at the top of the queue. The final message sent to users correctly shows the result of *job* execution.

In scenario 5 all submitted *job*s are correctly enqueued, each waiting for the availability of the resource, and they are all correctly executed, according to their arrival order.

Therefore, in all cases we executed, the model correctly evolves according to the expected behavior, and, after completion all resource are correctly released, so becoming available for processing new *job*s, and the agents return to the idle state.

6 Conclusion and future work

Grid technology makes available a great extent of computational power for solving many application problems with acceptable resource consumption. In this context, the specific middleware adopted for executing Users Jobs is a very critical requirement, which can affect success of the system. Its high complexity requires the use of formal methods for guaranteeing correct behavior within required quality of service. This paper is part of our research aimed at building a formal framework for studying Grid systems. Since ASMs have proven their practical benefits for the specification and analysis of several complex systems, we apply this formalism in Grid systems domain.

Here, we provide a formal description of the job Execution Management Services in terms of *asyncASM*s, and its implementation into *coreASM* tool. Job EMS is expressed as a composition of interoperable, always refineable, building blocks, and the resulting *asyncASM* model is an effective choice for defining a precise semantic foundation of Grid system. This solution allows coordination among different logical components in the Grid.

The simulation-based validation of the model provides an informal evidence of requirement satisfaction, and it makes possible a preliminarily analysis of some system properties. For example we can

see that each state can be reached starting from the initial idle state, and that all rules can always be fired, so the modeled system is deadlock free. Moreover, we can observe that it is always possible returning to the initial state, so that it is always possible implementing a proper recovery procedure in case of failures. Finally, it is worth noting that the Dispatcher Agent can be a bottleneck for the system, because it has to manage a lot of *job*s.

The ASM approach can be seen as a reference model for Grids studies: it can help both researchers and practitioners to better understand Grid behavior, to clarify concepts at the abstract desiderate formal level, to improve the efficiency and reduction of development costs, and to compare different solutions. In fact, thanks to the abstraction process, and to tools like *coreASM*, it is quite easy building an implementation of the model, spending few efforts, and different Grid strategies, depending on different middleware, can be derived as different refinements of the same abstraction. In this way, researchers can verify and validate the behavior of solutions they propose, and practitioners can easily compare the implementation of different proposals.

Future development of research is aimed at a twofold goal: on one hand, the model will be completed for encompassing proper management of some side aspects, for example resource allocation policy, and management of a pool of Dispatchers. On the other hand, since ASM-based approach enables the analysis of the model for evaluating computationally interesting properties, the obtained models will be analyzed for identifying possible weaknesses. In this sense, it can be challenging for researchers and practitioners investigating how the ASM models can help the interoperability and the standardization of Grid systems achieving optimal performance and reduction of costs.

Acknowledgement

This work has been partially funded by the Italian Ministry of Education, University and Research (MIUR), within the Piano Operativo Nazionale - PON02_00563_3489339.

The authors are very grateful to the anonymous reviewers for their constructive remarks and comments.

References

[1] I. Foster, C. Kesselman, and S. Tuecke, “The anatomy of the Grid: Enabling scalable virtual organizations”, *International Journal of High Performance Computing Application*, vol. 15, no.3, pp. 200-222, 2001.

[2] I. Foster, “What is the Grid? A Three Point Checklist”, *Global Grid Forum*, Available: http://www.globus.org/alliance/publications/papers.php, 2002.

[3] Globus Alliance – Globus Toolkit – www.globus.org

[4] I. Foster, I., Kishimoto, H., Savva, A., Berry, D., Djaoui, A., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., Reich, J.V. Reich, “The Open Grid Services Architecture, Version 1.5” , GFD-I.080, *Open Grid Forum*, Available: www.ogf.org/documents/GFD.80.pdf, 2006.

[5] http://glite.cern.ch/

[6] A. Bianchi, L. Manelli and S. Pizzutilo, “A Distributed Abstract State Machine for Grid Systems: A Preliminary Study”, in P. Iványi and B.H.V. Topping (Eds.) *Proceedings of the Second International Conference on Parallel, Distributed, Grid And Cloud Computing For Engineering, Civil-Comp Press*, Ajaccio, France, Paper 84, April 2011

[7] E. Börger, R. Stärk, *Abstract State Machine*, Springer, 2003.

[8] www.coreasm.org

[9] R. Farahbod, V. Gervasi, and U. Glaesser, "CoreASM: An extensible ASM execution engine", *Fundamenta Informaticae*, vol. 77, no.1-2, pp. 71-103, 2007.

[10] X. Zhao, B. Wang, L. Xu, “Grid Application Scheduling Model Based on Petri Net with Changeable Structure”, *Proceedings of the 6th International Conference on Grid and Cooperative Computing*, Los Alamitos, CA, pp.733-736, 2007.

[11] H. Storrle, J. Hausmann, “Towards a formal semantics of UML 2.0 activities”, in *Software Engineering*, Lecture Notes in Informatics vol. P-64, P. Liggesmeyer, K. Pohl, M. Goedicke, Eds., pp. 117-128, 2005.

[12] S. Sarstedt, and W. Guttmann, "An ASM Semantics of Token Flow in UML 2 Activity Diagrams", *Proceedings of the 6th International Andrei Ershov memorial conference on Perspectives of systems informatics*, I. Virbitskaite and A. Voronkov Eds., LNCS 4378, pp. 349-362, 2007.

[13] W. Reisig, "The Expressive Power of Abstract State Machines", *Computing and Informatics*, vol. 22, no.3-4, pp. 1-10, 2003.

[14] U. Glässer, Y. Gurevich, M. Veanes, “Abstract Communication Model for Distributed Systems”, *IEEE Transactions on Software Engineering*, vol.30, no.7, pp. 458-472, 2004.

[15] J. Lemcke, and A. Friesen, “Composing Web-service-like abstract state machines (ASMs)”, *Proceedings of the IEEE Congress on Services*, Salt Lake City, Utah, pp. 262–269, July 2007

[16] D. Lamch, R. Wyrzykowski, “Specification, analysis and testing of Grid environments using Abstract State Machines”, *Proceedings of the International Conference on Parallel Computing in Electrical Engineering*, Bialystok, Poland, pp. 116-120, September 2006.

[17] Z.Nemeth, V. Sunderam, “Characterizing Grids: Attributes, Definitions and Formalism”, *Journal of Grid Computing*, Vol. 1, no.1, pp. 9-23, 2003.

[18] Z.Nemeth, V. Sunderam, “A Formal Framework for Defining Grid Systems”, *Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid*, Berlin, pp. 202-211, May 2002.

[19] M.Parashar, J.C. Browne, "Conceptual and implementation models for the Grid", *Proceedings of the IEEE*, vol.93, no 3, pp.653-668, 2005.

[20] D. Zou, W. Qiang, Z. Shi, "A Formal General Framework and Service Access Model for Service Grid", *Proceedings of the 10th IEEE International Conference on Engineering of Complex Computer Systems*, Shanghai, China, pp. 349-356, June 2005.

[21] A. Blass, Y. Gurevich, "Abstract State Machines Capture Parallel Algorithms", *ACM Transactions on Computational Logic*, Vol. 4 no. 4, pp.578-651, 2003.

[22] R. Farahbod, and U. Glasser, "The CoreASM modeling framework", *Software – Practice and Experience*, 41, no.2, pp. 167–178, 2011.

[23] G.M.P. O'Hare, N.R. Jennings, *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, 1996.

[24] J. Yu, R. Buyya, "A taxonomy of scientific workflow management systems for grid computing", *ACM SIGMOD Record*, Vol. 34, no.3, pp. 44-49, 2005.

[25] J. Montes, A. Sánchez, J.J. Valdés, M.S. Pérez, P. Herrero, "Finding order in Chaos: A behavior model of the whole grid", *Concurrency and Computation: Practice & Experience*, Vol.22 no. 11, pp.1386-1415, 2010.

[26] D. Gannon, R. Bramley, G. Fox, S. Smallen, A. Rossi, R. Ananthakrishnan, F. Bertrand, K. Chiu, M. Farrellee, M. Govindaraju, S. Krishnan, L. Ramakrishnan, Y. Simmhan, A. Slominski, Y. Ma, C. Olariu, and N. Rey-Cenvaz, "Programming the Grid: Distributed Software Components, P2P and Grid Web Services for Scientific Applications", *Cluster Computing*, Vol.5, no3, pp.325-336, 2002.

[27] A.N. Haidar, P. V. Coveney, A.E. Abdallah, P. Y. A. Ryan, B. Beckles, J. M. Brooke and M.A.S. Jones "Formal Modelling of a Usable Identity Management Solution for Virtual Organisations", *Proceedings of the 2nd Workshop on Formal Aspects of Virtual Organisations*, Electronic Proceedings in Theoretical Computer Science - EPTCS 16, pp. 41-50, 2010.

[28] J. Woodcock, and J. Davies, *Using Z Specification, Refinement, and Proof*. C.A.R Hoare series editor, Prentice Hall International, 1996.

[29] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall, 1985.

[30] F. Neubauer, A. Hoheisel, J. Geiler, "Workflow-based Grid applications", *Future Generation Computer Systems*, Vol.22, no.1-2, pp.6–15, 2006.

[31] Z. Guan, F. Hernandez, P. Bangalore, J. Gray, A. Skjellum, V. Velusamy, Y. Liu, "Grid-Flow: a Grid-enabled scientific workflow system with a Petri-net-based interface", *Concurrency and Computation: Practice and Experience*, Vol. 18, no. 10, pp. 1115–1140, 2006.

[32] W.D. Liu, J.X. Song, C. Lin, "Modeling and Analysis of Grid Computing Application Based Price Timed Petri Net", *Acta Electronica Sinica*, 2005-08.

[33] Y. Gurevich, B. Rossman, W. Schulte, "Semantic essence of AsmL", *Theoretical Computer Science*, Vol.343, no.3, pp.370 – 412, 2005.

[34] Y. Gurevich, "Sequential Abstract State Machines capture Sequential Algorithms", *ACM Transactions on Computational Logic*, Vol.1, no.1, pp. 77-111, 2000.

[35] R. Milner, *Communicating and Mobile Systems: the π-calculus*, Cambridge University Press, 1999.

[36] J. Zhou, G. Zeng, "Describing and reasoning on the composition of grid services using pi-calculus", *Proceedings of the 6th IEEE International Conference on Computer and Information Technology*, Seoul, Korea, pp.48-53, September 2006.

[37] R. Eshuis, R. Wieringa, "Comparing Petri Net and Activity Diagram Variants for Workflow Modelling – A Quest for Reactive Petri Nets", *Petri Net Technology for Communication Based Systems*, LNCS vol.2472, Springer, 2003, pp 321-351.

[38] R. Farahbod, V. Gervasi, U. Glässer, "Executable formal specifications of complex distributed systems with CoreASM", *Science of Computer Programming*, 2012.

[39] P. Andreetto, S. Andreozzi, A. Ghiselli, M. Marzolla, V. Venturi, L. Zangrando, "Standards-based Job Management in Grid Systems", *Journal of Grid Computing*, Vol.8, no.1, pp.19-45, 2010.

[40] W. van der Aalst, C. Bratosin, N. Sidorova, and N. Trcka, "A Reference model for Grid Architectures and its Validation", *Concurrency and Computation: Practice and Experience*, Vol.22, no.11, pp. 1365-1385, 2010.

[41] J. Atlas, M. Swany, K.S. Decker, "Flexible Grid Workflows Using TAEMS", Proceedings of the Workshop on Exploring Planning and Scheduling for Web Services, Grid and Autonomic Computing, at AAAI05, Pittsburgh, Pennsylvania, pp. 24-31, July2005.

An Enterprise Digital Right Management Scheme with Anonymous Trust for Mobile Devices

Jen-Ho Yang
Department of Multimedia and Mobile Commerce, Kainan University,
No. 1, Kannan Road, Luzhu, Taoyuan County, 33857, Taiwan
E-mail: jenhoyang@mail.knu.edu.tw

Hung-Ming Sun
Department of Information Management, Kainan University,
No. 1, Kannan Road, Luzhu, Taoyuan County, 33857, Taiwan
E-mail: sunhm@mail.knu.edu.tw

Ping-Liang Chen
Department of Multimedia and Mobile Commerce, Kainan University,
No. 1, Kannan Road, Luzhu, Taoyuan County, 33857, Taiwan
E-mail: pingliang@mail.knu.edu.tw

Keywords: enterprise digital right management, authentication, user privacy, anonymity, mobile device

Received: August 3, 2012

In recent years, various enterprise digital right management (E-DRM) schemes have been proposed to protect and manage access rights of digital contents for the enterprise applications. However, we find that the previous E-DRM schemes do not protect the user privacy while mobile users access digital contents. In addition, the previous E-DRM schemes have high computation and communication loads. Besides, these schemes do not provide usage tracking for the digital content, and thus the digital right may be abused by malicious users. To solve the above problems, we propose a new E-DRM scheme with anonymous trust for mobile devices in this paper. The proposed scheme has low computation and communication loads, and it provides the user anonymity and usage tracking. Therefore, the proposed scheme is more efficient and practical than the related works for E-DRM applications.

Povzetek: Razvita je nova shema E-DRM za določanje zaupanja v mobilnih napravah.

1 Introduction

The Digital Right Management (DRM) scheme is a digital technique that protects and manages the access rights of digital contents. It can prevent the confidential information of a digital content from unauthorized usages by illegal users. Generally, there are four roles in the DRM scheme: a content provider (author), a consumer (client), a clearing house, and a distributor [1].

The content provider creates the digital content and encrypts it using some proper cryptosystems, such as RSA [2], ElGamal [3], and ECC [4]. Then, the content provider sends the encrypted content to the distributor (e.g., web server or online shop) for online distribution. Note that some researches combine the distributor with the content provider. Next, the content provider sends the usage rules to the clearing house, such as the copy permit, the pay-per-view, and the usage fee, to specify how to use the digital content. Note that the clearing house is responsible for issuing the digital license and handling the financial transactions for the content provider, the distributor, and the consumer.

Assume that the consumer downloads the digital content from the web server. To access the encrypted content, the consumer requests the clearing house to issue a valid license, which contains the decryption key, usage rules, and descriptions of the digital content. Then, the clearing house performs the user authentication mechanisms [5-11] to verify the identity of the consumer. Then, the clearing house can charge the consumer account for the digital content. After the consumer has paid the money, the clearing house sends the license to the consumer. Finally, the consumer has the access rights to use the digital content. Figure 1 illustrates the above DRM infrastructure as follows.

Due to the digital content can be easily obtained and distributed via the Internet, the DRM scheme becomes a popular research topic in recent years. Therefore, various DRM schemes [12-16] have been proposed to protect and manage the access right of a digital content. In 2005, Zhang et al. [12] proposed a license management scheme with anonymous trust for digital rights management (LMSAT) based on Elliptic Curve Cryptosystem (ECC). Zhang et al.'s scheme provides the user privacy protection, and their scheme allows the client access the content by using any permissive device. Thus, their

scheme provides a flexible license acquisition and usage tracking for the digital right management. However, Zhang et al.'s scheme has large computation loads because it utilizes the public-key cryptosystem [2-4]. Thus, their scheme is not suitable for the mobile device with low computation ability.

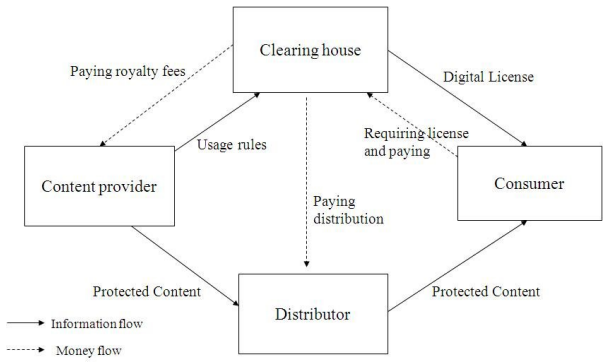


Figure 1: The DRM infrastructure [1].

On the other hand, the DRM scheme can be also applied to digital right protection for enterprise applications. In 2008, Chen [14] proposed a traceable enterprise digital right management (E-DRM) scheme based on one-way hash functions for mobile devices. However, Chang et al. [16] found that Chen's scheme is insecure, and thus they proposed an improved E-DRM scheme in 2010. Chang et al.'s scheme solved the security problem, and it has lower computation cost.

Unfortunately, we find that Chang et al.'s scheme has some flaws shown as follows. First, their scheme does not provide the anonymity so that the user's privacy may be revealed. Second, their scheme does not provide the usage trace of the digital content. Thus, the digital right may be ruined by any malicious user. Third, their scheme uses the certificate to authenticate the mobile user. This increases the communication time because the mobile user must apply the certificate from the certification authority (CA). Fourth, their scheme still has large computation costs because it has to compute the digital signature for authenticating messages.

To solve the above-mentioned flaws, we propose a new E-DRM authentication scheme with user anonymity for mobile devices in this paper. In the proposed scheme, the mobile user can be authenticated by using an anonymity identity so that the user privacy can be well-protected. Moreover, the proposed scheme provides the usage tracking of a digital content, and thus the digital right would not be abused by any malicious user. Instead of using the certificate and digital signature, we only use the one-way hash function and exclusive-or (XOR) operations to accomplish the user authentication so the computation costs can be greatly reduced. According to the above descriptions, the proposed scheme is more efficient and practical than the previously proposed E-DRM schemes, and it is more suitable for the digital right management in mobile environments.

2 Review of Chang et al.'s E-DRM scheme

In this section, we review Chang et al.'s scheme [16] and point out some drawbacks of their scheme. There are six roles in their scheme: the author of the digital content, the package server (PS), the content server (CS), the license server (LS), the authorization authority (AA), and the mobile user (MU). Table 1 shows the notations used in Chang et al.'s scheme.

T, τ	A timestamp and a time constant
$Cert$	The digital certificate of a mobile user
P_i	The *i*-th one-time password
$SEED$	The initial random seed number generated by the authorization authority
N_i	The *i*-th request random number generated by authorization authority
$IMEI$	A unique International Mobile Equipment Identification number of each mobile terminal
msg_{req}	The authorization request message of the mobile user
CID	The identity of a digital content
$DRM-AP_{type}$	The type of the DRM-Enable application
$V_X(\cdot)/S_X(\cdot)$	The verifying/signing function using *X*'s public/secret key
KEY_{CID}	The symmetric key for the digital content with *CID*
$E_{KEY_X}(\cdot)$ / $D_{KEY_X}(\cdot)$	The symmetric encryption / decryption function using a symmetric key KEY_X
$H(\cdot), F(\cdot)$	Two collision free one-way hash functions

Table 1: The notations used in Chang's E-DRM scheme

The package phase:

Step 1. The author produces the digital content and uploads it to PS.

Step 2. PS chooses a symmetric key KEY_{CID} to encrypt M by $C = E_{KEY_{CID}}(M)$. Then, PS generates the content header CH and uses its private key to produce two signatures $Sig_C = S_{PS}(C)$ and $Sig_{CH} = S_{PS}(CH)$. In addition, PS generates the E-DRM formatted file and sends it to CS.

Step 3. PS generates a signature $Sig_{KEY_{CID}} = S_{PS}(CID, KEY_{CID})$ and sends the messages $\{CID, KEY_{CID}, Sig_{KEY_{CID}}\}$ to LS.

Step 4. After receiving the E-DRM formatted file, CS uses PS's public key to verify whether $Sig_C = S_{PS}(C)$ and $Sig_{CH} = S_{PS}(CH)$ are valid or not. If they are valid, then CS stores the E-

DRM formatted file to its database and publishes the file to its public directory.

Step 5. After receiving {*CID*, KEY_{CID} , $Sig_{KEY_{CID}}$ }, LS uses PS's public key to verify $Sig_{KEY_{CID}} = S_{PS}(CID, KEY_{CID})$. If the signature is valid, then LS stores (*CID*, KEY_{CID}) to its database.

Step 6. To access the digital content, MU downloads the E-DRM file from the public directory of CS. According to *URL* of the content header, MU sends the registration request to AA for asking the access right.

Content Identity (*CID*)	Type of the DRM-enable Application ($DRM-AP_{type}$)	Identity of the Decryption Key (KEY_{CID})
Attributes	Signature of the Encrypted Content (Sig_C)	Authorization Authority (*URL*)
The Encrypted Digital Content(*C*)		

Figure 2: The E-DRM formatted file.

The registration phase:

Step 1. MU sends its *IMEI* and *Cert* to AA via a secure channel.

Step 2. AA checks MU by verifying *IMEI* and *Cert*. If MU is valid, then AA generates an initial random number N_1 = *SEED* and sends it to MU through a secure channel.

Step 3. AA stores *IMEI*, *Cert*, and *SEED* in its database and sends them to the LS through a secure channel.

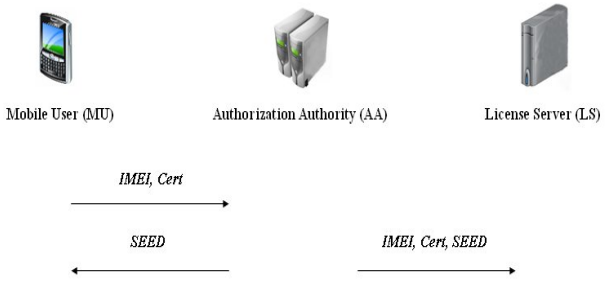


Figure 3: The registration Phase of Chang et al.'s scheme.

The authentication phase:

Step 1. When MU asks for *i*-th authentication from AA, it uses N_i to generate *i*-th one-time password $P_i = H^i(N_i \oplus IMEI \oplus Cert \| T \| CID)$, where $H^i(\cdot)$ denotes the message *x* performs one-way hash function for *i* times, and *T* is a timestamp. Then, MU sends the messages {*i*, msg_{req} , *T*, *CID*, P_i , *Cert*} to AA.

Step 2. AA checks whether *T* is correct or not. If *T* is smaller than τ, AA loads N_i and *IMEI* from its database and computes $H^i(N_i \oplus IMEI \oplus Cert \| T \| CID)$. If P_i and $H^i(N_i \oplus IMEI \oplus Cert \| T \| CID)$ are equal, then AA computes $H^{F(N_i)}(IMEI \| T \| N_i) \oplus N_{i+1}$ and $H(IMEI \| N_i \| N_{i+1})$. Then, AA sends the above messages to MU.

Step 3. AA generates $Sig_{AA} = S_{AA}(IMEI, CID, i, T, Cert, H^{F(N_i)}(IMEI \| T \| N_i) \oplus N_{i+1})$ and sends {*IMEI*, *CID*, *i*, *T*, *Cert*, $H^{F(N_i)}(IMEI \| T \| N_i) \oplus N_{i+1}$, Sig_{AA}} to MU.

Step 4. After MU obtains the messages sent from AA, it computes $N'_{i+1} = H^{F(N_i)}(IMEI \| T \| N_i) \oplus H^{F(N_i)}(IMEI \| T \| N_i) \oplus N_{i+1}$ and checks whether $H(IMEI \| N_i \| N'_{i+1})$ is equal to $H(IMEI \| N_i \| N_{i+1})$ or not. If they are equal, MU accepts N'_{i+1} and keeps it.

Step 5. LS uses the public key of AA to verify the digital signature by $V_{AA}(Sig_{AA}) = IMEI, CID, i, T, Cert, H^{F(N_i)}(IMEI \| T \| N_i) \oplus N_{i+1}$. If the equation holds, LS uses KEY_{CID} stored in its database to generate $H^{F(N_i)}(IMEI \oplus Cert \oplus T \oplus N_i) \oplus KEY_{CID}$ and $H(IMEI \| N_i \| KEY_{CID})$. Then, LS sends the above messages to MU. Finally, MU computes the equation $N'_{i+1} = H^{F(N_i)}(IMEI \| T \| N_i) \oplus H^{F(N_i)}(IMEI \| T \| N_i) \oplus N_{i+1}$ and stores it.

Step 6. To obtain the symmetric key, MU firstly computes the equation: = ( )(⊕ | |) ⊕ . Next, MU checks whether $H(IMEI \| N_i \| KEY'_{CID})$ is equal to $H(IMEI \| N_i \| KEY_{CID})$ or not. If they are equal, MU computes $M = D_{KEY'_{CID}}(C)$ to obtain the digital content.

According to the above description, we point out some flaws of Chang et al.'s scheme as follows. First, their scheme does not provide the user anonymity so that the user privacy may be revealed during the message transactions. Second, their scheme adopts the certificate and digital signature for user and message authentications. This causes large computation and communication loads. Third, their scheme does not consider the usage tracking of the digital content. Thus, the digital right may be abused by malicious users. To overcome the above flaws, we propose a new DRM scheme in next section.

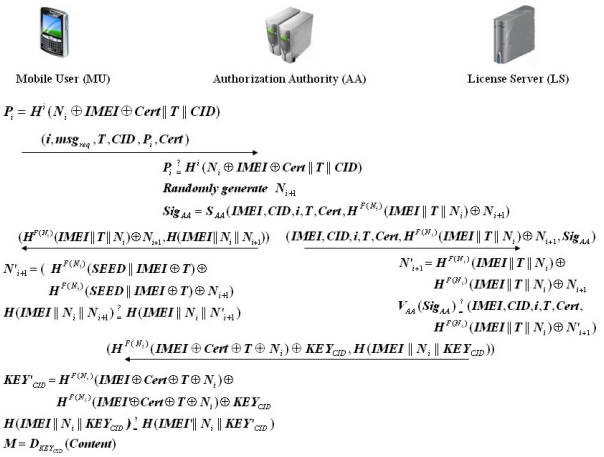


Figure 4: The authentication phase of Chang et al.'s scheme.

3 The proposed E-DRM scheme

There are three roles in the proposed scheme: the content provider (CP), the clearing house (CH), and the client device with the DRM agent (DA). The E-DRM model of the proposed scheme is similar as Figure 1. The difference of the proposed scheme and Chang et al.'s scheme is that the proposed scheme combines the functions of authorization authority and license server into a clearing house. This combination can greatly reduce the communication time, and that is why we do not adopt the model of Chang et al.'s scheme.

In the proposed scheme, DA is loaded in the Client's mobile device. And, DA is responsible for paying the digital content, acquiring the digital license from CH, authenticating the license and the content, decrypting the encrypted content, and reporting the usage to CH. The notations used in the proposed scheme are shown in Table 2.

Notations	Explanations
$H(.)$	A secure one-way hash function
SK	The session key
$E_{SK}(\cdot) / D_{SK}(\cdot)$	The symmetric encryption / decryption with the session key SK
X	The secret key of the content provider
$License$	License of a digital content
$Anonymity\_ID$	The anonymity identity of a client
$Content\_ID$	The identity of a digital content
$Usage\_Rules$	The usage rules of a digital content
$Usage\_Data$	The usage data of the digital content
$\|\|$	The concatenation of strings
SN	The sequence number of the license
$\oplus$	Exclusive-or operation
$Decryption\_Key$	The decryption key for the encrypted digital content
$Other\_Data$	The other information of the license

Table 2. The notations used in the proposed scheme

The registration phase:

In this phase, the client sends the registration information to the CP. Then, CP sends an anonymity identity and the authentication information to the client. Finally, the above information is stored in DA for the authentication. The steps of this phase are shown as follows.

Step 1. For the registration, the client sends his/her identity (ID) and password (PW) to CP in a secure channel.

Step 2. CP generates $Anonymity\_ID$ to compute the authentication information $Auth\_Info = H(Anonymity\_ID \oplus X) \oplus PW$ for the client. Then, CP sends $Anonymity\_ID$ and $Auth\_Info$ to the client.

Step 3. The client stores $Anonymity\_ID$ and $Auth\_Info$ in his/her device for later authentications.

The authentication and license acquisition phase:

In this phase, DA downloads the encrypted digital content from CP. To access the digital content, DA has to get the digital license from CH. In addition, CH is responsible for authenticating the client and sending the license to DA. Note that the client has to pay the money to CH for buying the license. However, we omit the payment steps because it is another research topic for DRM. The steps of this phase are shown as follows.

Step 1. The client computes $Auth\_Info \oplus PW = H(Anonymity\_ID \oplus X)$ by using PW. Then, the client (DA) chooses a random number R_{DA} to compute $S_{DA} = H(H(Anonymity\_ID \oplus X)) \oplus R_{DA}$ and $C_{DA} = H(R_{DA})$. Finally, DA sends $Anonymity\_ID$, S_{DA}, and C_{DA} to CH.

Step 2. CH computes $R_{DA}^{'} = S_{DA} \oplus H(H(Anonymity\_ID \oplus X))$ to check if C_{DA} is equal to $C_{DA}^{'} = H(R_{DA}^{'})$. If they are equal, then CH authenticates that DA is valid. Next, CH chooses random number R_{CH} to compute $S_{CH} = H(H(Anonymity\_ID \oplus X) \| R_{DA}) \oplus R_{CH}$ and the session key $SK = H(H(Anonymity\_ID \oplus X) \| R_{DA} \| R_{CH})$. Finally, CH computes $C_{CH} = H(R_{DA} \| R_{CH} \| SK)$ and sends S_{CH} and C_{CH} to DA.

Step 3. DA computes $R_{CH}^{'} = S_{CH} \oplus H(H(Anonymity\_ID \oplus X) \| R_{DA})$ and the session key $SK^{'} = H(H(Anonymity\_ID \oplus X) \| R_{DA} \| R_{CH}^{'})$. Then, DA computes $C_{CH}^{'} = H(R_{DA} \| R_{CH}^{'} \| SK^{'})$

to check if C_{CH} is equal to $C_{CH}^{'}$. If they are equal, then DA authenticates that CH and *SK* are both valid. Next, DA computes $E_{SK}(Anonymity\_ID \| Content\_ID \| Usage\_Rules \| SK)$ and sends it to CH.

Step 4. CH computes the digital license which contains the decryption key by the equation: $License = \{SN \| Content\_ID \| Usage\_Rules \| Decryption\_Key \| OtherData\}$. Then, CH computes $E_{SK}(License \| SK)$ and sends it to DA. Finally, DA can use *SK* to decrypt $E_{SK}(License \| SK)$ and obtain *License*. Thus, the client gets *Decryption_Key* from *License* and uses it to decrypt the encrypted digital content.

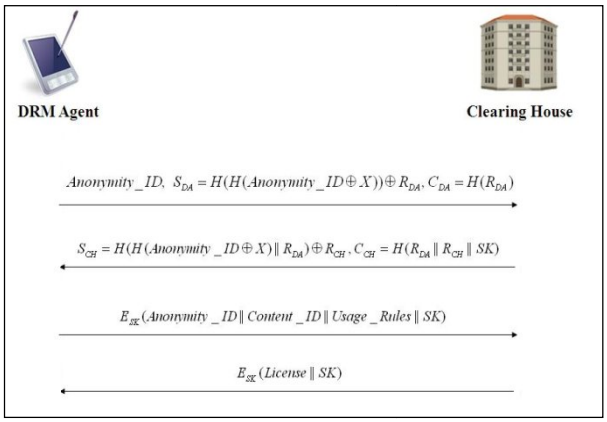


Figure 5: The authentication and license acquisition phase.

The usage tracking phase:

In this phase, CH receives the report of the content usage from DA. For fraud prevention, CH needs to authenticate the validity of DA. In addition, the usage information needs to be encrypted for protecting the user privacy. The steps of this phase are shown as follows.

Step 1. First, CH chooses a random number $\overline{R}_{CH}$ to compute $\overline{S}_{CH} = H(H(Anonymity\_ID \oplus X)) \oplus \overline{R}_{CH}$ and $\overline{C}_{CH} = H(\overline{R}_{CH})$. Then, CH sends $Anonymity\_ID$, $\overline{S}_{CH}$, and $\overline{C}_{CH}$ to DA.

Step 2. DA computes $\overline{R}_{CH}^{'} = \overline{S}_{CH} \oplus H(H(Anonymity\_ID \oplus X))$ to check if $\overline{C}_{DA}$ is equal to $\overline{C}_{CH}^{'} = H(\overline{R}_{CH}^{'})$. If they are equal, then DA authenticates that CH is valid. Next, DA chooses random number $\overline{R}_{DA}$ to compute $\overline{S}_{DA} = H(H(Anonymity\_ID \oplus X) \| \overline{R}_{CH}) \oplus \overline{R}_{DA}$ and the session key $\overline{SK} = H(H(Anonymity\_ID \oplus X) \| \overline{R}_{CH} \| \overline{R}_{DA})$. Next, DA computes $\overline{C}_{DA} = H(\overline{R}_{CH} \| \overline{R}_{DA} \| \overline{SK})$ and sends $\overline{S}_{DA}$ and $\overline{C}_{DA}$ to CH.

Step 3. CH computes $\overline{R}_{DA}^{'} = \overline{S}_{DA} \oplus H(H(Anonymity\_ID \oplus X) \| \overline{R}_{CH})$ and the session key $\overline{SK}^{'} = H(H(Anonymity\_ID \oplus X) \| \overline{R}_{CH} \| \overline{R}_{DA}^{'})$. Then, CH computes $\overline{C}_{DA}^{'} = H(\overline{R}_{CH} \| \overline{R}_{DA}^{'} \| \overline{SK}^{'})$ to check if $\overline{C}_{DA}$ is equal to $\overline{C}_{DA}^{'}$. If they are equal, then CH ensures that DA and $\overline{SK}$ are both valid. Next, CH computes $E_{\overline{SK}}(Anonymity\_ID \| Content\_ID \| SN \| \overline{SK})$ and sends it to DA.

Step 4. DA computes $E_{\overline{SK}}(Usage\_Data \| \overline{SK})$ and sends it to CH. Finally, CH can decrypt $E_{\overline{SK}}(Usage\_Data \| \overline{SK})$ to trace the content usage.

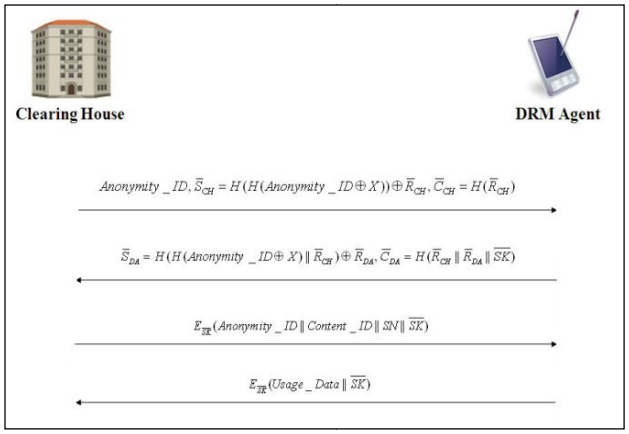


Figure 6: The usage tracking phase.

According to the above descriptions, the proposed scheme is designed by one-way hash functions and XOR operations. Thus, the computation cost of the mobile device can be greatly reduced. In addition, the proposed scheme provides the usage tracking and user anonymity so the digital right and the user privacy can be well-protected. Besides, we simplify the communication model and eliminate the use of the certificate. Therefore, the communication cost of the proposed scheme is much lower than that of Chang et al.'s scheme.

4 Discussions

In this section, we perform some possible attacks on the proposed scheme to show the security analyses as follows.

Man-in-the-middle attack:

Assume that an attacker wants to get the *Decryption_Key* from CH, and he/she impersonates DA to share the session key *SK* with CH. First, the attacker intercepts all messages sent form DA to CH. Then, the attacker impersonates DA to generate the forged

$S_{DA}^{"} = H(H(Anonymity\_ID \oplus X^{"})) \oplus R_{DA}^{"}$ and $C_{DA}^{"} = H(R_{DA}^{"})$. Next, the attacker sends $Anonymity\_ID$, $S_{DA}^{"}$, and $C_{DA}^{"}$ to CH. However, the attacker cannot pass the authentication because $S_{DA}^{"} \neq S_{DA}$. Thus, CH can find that $Anonymity\_ID$, $S_{DA}^{"}$, and $C_{DA}^{"}$ are sent from an attacker. Similarly, an attacker cannot impersonate CH because he does not know the real $H(Anonymity\_ID \oplus X)$. Therefore, the man-in-the-middle attack is infeasible for the proposed scheme.

Outsider attack:

Assume that an attacker wants to get *SK*, and then he/she wiretaps the communications between DA and CH. Thus, the attacker can obtain S_{DA}, C_{DA}, S_{CH}, and C_{CH}. Then, the attacker wants to use the information to compute $SK = H(H(Anonymity\_ID \oplus X) \| R_{DA} \| R_{CH})$. Unfortunately, the attacker needs to know $H(Anonymity\_ID \oplus X)$, R_{DA}, and R_{CH} from S_{DA}, C_{DA}, S_{CH}, and C_{CH} to obtain *SK*. Thus, the attack is impossible because $H(Anonymity\_ID \oplus X)$, R_{DA}, and R_{CH} are protected by the one-way hash function. Therefore, the outsider attack is infeasible for the proposed scheme.

Replay attack:

Assume that an attacker collects the messages once being transferred between DA and CH. To get the digital license or the session key, the attacker may pretend to be DA or CH by resending the pre-collected messages between DA and CH. However, this attack cannot work because all messages are generated and changed according to the random numbers in the proposed scheme. Thus, the messages are different in each time so that the replay attack is infeasible for the proposed scheme.

Stolen-verifier attack:

Assume that the client's device is lost or stolen by an attacker. The attacker may try to use this device to access the digital content as a legal client. However, this attack is impossible because the authentication information $H(Anonymity\_ID \oplus X)$ is protected by user's password *PW*. And, the attacker does not know the correct *PW* to compute $Auth\_Info \oplus PW$. Therefore, the proposed scheme is still secure even if the client's device is lost or stolen.

Impersonating attack:

Assume that an attacker wants to impersonate a legal user to access the digital content. Then, he/she may generate a forged $H(Anonymity\_ID \oplus X)$" and R_{DA}" to compute $S_{DA}" = H(H(Anonymity\_ID \oplus X)") \oplus R_{DA}"$ and sends $Anonymity\_ID$, $S_{DA}"$, and $C_{DA}"$ to CH, where $C_{DA}" = H(R_{DA}")$. However, this attack does not work because the CH will use $H(Anonymity\_ID \oplus X)$ to compute $R_{DA}^{'} = S_{DA}" \oplus H(H(Anonymity\_ID \oplus X))$. Finally, CH will find that $C_{DA}"$ is not equal to $C_{DA}^{'} = H(R_{DA}^{'})$. Therefore, the impersonating attack is impossible for the proposed scheme.

Performance analyses:

In Table 3, we show some comparisons among the proposed scheme, Zhang et al.'s scheme [12], and Chang et al.'s scheme [16]. According to Table 3, only the proposed scheme has no stolen-verifier attack. Unlike the other schemes, the proposed scheme can solve the security problem when the mobile devise is lost or stolen. Moreover, the proposed scheme provides the user anonymity and usage tracking.

Table 3 also shows the computation costs in the user's sides of these three schemes. The computation costs of the asymmetric encryption, symmetric encryption, one-way hash function and exclusive-or operation are denoted as A, S, H, and X, respectively. According to [18], the measurement of the above computation costs can be denoted as A>>>S>>>H>>>X, where "A>>>S" means that A is much larger than S. According to Table 3, the computation cost of the proposed scheme is lower than those of the other schemes. In conclusion, the proposed scheme is more efficient and practical than the other schemes.

Meth. / Comp.	[12]	[16]	Ours
User anonymity	Yes	No	Yes
Usage tracking	Yes	No	Yes
Stolen-verifier attack	Yes	Yes	No
Computation cost	3A+2S +3H	1S+(2|$F(N_i)$| +i+2)H+7X	1S+4H+3 X

Table 3. Comparisons of the related works

5 Conclusion

In this paper, we propose an efficient and practical E-DRM scheme with anonymity trust for mobile devices. According to our analysis, the proposed scheme has low computation cost so it is suitable for mobile devices. In addition, it protects the user's privacy by using the anonymous identity for the user authentication. Thus, the proposed scheme allows users access their digital contents by any permissive mobile devices. Besides, the proposed scheme provides usage tracking to make sure that the access is not abused by malicious users. Compared with the related works, the proposed scheme is more efficient and practical in the E-DRM applications for mobile devices. Based on the proposed scheme, we will investigate the usage charge for E-DRM applications to make our research more complete in the future.

Acknowledgement

This work was supported in part by National Science Council under the grants NSC 100-2221-E-424-006.

References

[1] Q. Liu, S. N. Reihaneh, and N. P. Sheppard, "Digital rights management for content distribution," *Proceedings of Australasian Information Security Workshop 2003 (AISW2003), Conferences in Research and Practice in Information Technology*, Adelaide, Australia., Vol. 21, 2003.

[2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, Vol. 21, pp. 120-126, 1978.

[3] T. ElGamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Transactions on Information Theory*, Vol. 31, pp. 469-472, 1985.

[4] N. Koblitz, "Elliptic curve cryptosystem", *Mathematics of Computation*, Vol. 48, pp. 203-209, 1987.

[5] T. Kwon and J. Song, "Efficient key exchange and authentication protocols protecting weak secrets," *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E81-A, pp. 156-163, 1998.

[6] T. Kwon and J. Song, "Authenticated key exchange protocols resistant to password guessing attacks," *IEEE Proceedings Communications*, Vol. 145, pp. 304-308, 1998.

[7] M. Sandirigama, A. Shimizu, and M. T. Noda, "Simple and secure password authentication protocol (SAS)," *IEICE Transactions on Communications*, Vol. E83-B, pp. 1363-1365, 2000.

[8] M. S. Hwang, C. C. Lee, and Y. L. Tang, "A simple remote user authentication protocol," *Mathematical and Computer Modelling*, Vol. 36, pp. 103-107, 2002.

[9] H. Y. Chien and J. K. Jan, "Robust and simple authentication protocol," *Computer Journal*, Vol. 46, pp. 193-201, 2003.

[10] E. Bresson, O. Chevassut, A. Essiari, and D. Pointcheval, "Mutual authentication and group key agreement for low-power mobile devices," *Computer Communications*, Vol. 27, pp. 1730-1737, 2004.

[11] H. M. Sun and H. T. Yeh, "Password-based authentication and key distribution protocols with perfect forward secrecy," *Journal of Computer and System Sciences*, Vol. 72, pp. 1002-1011, 2006.

[12] J. Zhang, B. Li, L. Zhao, and S. Q. Yang, "License management scheme with anonymous trust for digital rights management," *Proceedings of* 2005 IEEE International Conference on Multimedia and Expo, Amsterdam, Netherlands, pp. 257-260, July 2005.

[13] C. C. Lin and P. H. Chiang, "A mobile trading scheme for digital content based on digital rights", *Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications (ISDA 2008)*, Kaohsiung City, Taiwan, Vol. 3, pp. 451-456, Nov. 2008.

[14] C. L. Chen, "A secure and traceable E-DRM system based on mobile device," *Expert Systems with Applications*, Vol. 35, No. 3, pp. 878-886, Oct. 2008.

[15] C. C. Lin, S. C. Wu, P. H. Chiang, and C. C. Chen, "Enterprise-oriented digital rights management mechanism: eDRM," *Proceedings of International Conference on Availability, Reliability and Security*, ares, pp. 923-928, 2009.

[16] C. C. Chang, J. H. Yang, and, D. W. Wang, "An efficient and reliable E-DRM scheme for mobile environments," *Expert Systems with Applications*, Vol. 37, No. 9, pp. 6168-6176, Sep. 2010.

[17] D. Hankerson, A. Menezes, and S. Vanstone, Guide to elliptic curve cryptography, Springer-Verlag, New York, USA, 2004.

[18] B. Schneier, Applied Cryptography: Protocols, Algorithms, and Source Code in C, Second Edition, John Wiley & Sons, Inc., New York, USA, 1996.

A Benchmarking Algorithm to Determine the Sequence of Stable Data Gathering Trees for Wireless Mobile Sensor Networks

Natarajan Meghanathan
Associate Professor, Department of Computer Science, Jackson State University
Jackson, MS, USA
E-mail: natarajan.meghanathan@jsums.edu

Philip Mumford
Senior Electronics Engineer, Sensors Directorate, Air Force Research Lab/RYWC
Wright-Patterson Air Force Base, OH, USA

Keywords: stability, data gathering tree, mobile sensor networks, tree lifetime, network lifetime, coverage loss time, simulations

Received: September 15, 2012

We propose a benchmarking algorithm to determine the sequence of longest-living stable data gathering trees for wireless mobile sensor networks whose topology changes dynamically with time due to the random movement of the sensor nodes. Referred to as the Max.Stability-DG algorithm, the algorithm assumes the availability of complete knowledge of future topology changes and is based on the following greedy principle coupled with the idea of graph intersections: Whenever a new data gathering tree is required at time instant t corresponding to a round of data aggregation, choose the longest-living data gathering tree from time t. The above strategy is repeated for subsequent rounds over the duration of the lifetime of the sensor network to obtain the sequence of longest-living stable data gathering trees spanning all the live sensor nodes in the network such that the number of tree discoveries is the global minimum. Thus, the number of tree discoveries incurred with the Max.Stability-DG algorithm will serve as the lower bound for the number of discoveries for any network-wide communication topology (like spanning trees and connected dominating sets) determined through any other algorithm for data gathering in mobile sensor networks under identical operating conditions. In addition to theoretically proving the correctness of the Max.Stability-DG algorithm, we also conduct exhaustive simulations to evaluate the performance of the Max.Stability-DG trees and compare to that of the minimum-distance spanning tree based data gathering trees with respect to metrics such as tree lifetime, delay per round, node lifetime, network lifetime and coverage loss time, under both sufficient-energy and energy-constrained scenarios.

Povzetek: Predstavljen je izvirni referenčni algoritem za iskanje najbolj obstojnih dreves v brezžičnih senzorskih omrežjih.

1 Introduction

A wireless sensor network is a network of several smart sensor nodes that can gather data about the ambient environment as well as intelligently process them before propagating to a control center called the sink, which is typically located far away from the field being monitored and used to remotely administer the sensor network. Even though widely used for data gathering in several real-time applications, wireless sensor networks are mostly deployed for static environments, wherein the mobility of the sensor nodes, the users and the monitored phenomenon are all totally ignored. A wireless mobile sensor network (WMSN) is the next logical evolutionary step for sensor networks in which mobility needs to be handled in all its forms. With the widespread growth of embedded systems and ubiquitous computing technologies, a mobile sensor network could be envisioned as a homogeneous or heterogeneous network of sensor-equipped computers, mobile phones and vehicles, generally referred to as nodes (having one or more sensors like a camera sensor, microphone, GPS sensor, etc) [10]. The nodes of a WMSN often move in an arbitrary fashion, independent of each other. Some of the applications [9] of WMSNs could be traffic monitoring, route planning, civil infrastructure monitoring (say, attaching vibration sensors to cars and monitoring the conditions of roads/pot holes), geo-imaging, mobile target tracking [33] and etc. WMSNs can be used to monitor and collect data over a much larger geographical area with less number of sensor nodes compared to static sensor networks. With mobility, the entire area could be covered with fewer sensors/nodes over a period of time

Like their static counterparts, the mobile sensor nodes are likely to be constrained with limited battery charge, memory and processing capability as well as operate under a limited transmission range. Two sensor

nodes that are outside the transmission range of each other cannot communicate directly. The bandwidth of a WMSN is also expected to be as constrained as that of a static sensor network. Due to all of the above resource and operating constraints, it will not be a viable solution to require every sensor node to directly transmit their data to the sink over a longer distance. Also, if several signals are transmitted at the same time over a longer distance, it could lead to lot of interference and collisions. Thus, there is a need for employing energy-efficient data gathering algorithms that can effectively combine the data collected at these sensor nodes and send only the aggregated data (that is a representative of the entire network) to the sink.

Over the past few years, the sensor network research community has proposed a number of data gathering algorithms to effectively combine the data collected at these sensor nodes through a properly constructed communication topology and send only the aggregated data (that is a representative of the entire network) to the sink. However, a majority of these data gathering algorithms are meant for static sensor networks (i.e., static sensor nodes) with either a static (e.g., [8][12]) or mobile (e.g., [26][29]) sink. Tree-based data gathering is considered to be the most energy-efficient [16] in terms of the number of link transmissions; however, almost all of the tree-based data gathering algorithms have been proposed for static sensor networks without taking the mobility of the sensor nodes into consideration. In the presence of node mobility, the network topology changes dynamically with time – leading to frequent tree reconfigurations. Thus, mobility brings in an extra dimension of constraint to a WMSN and we need algorithms that can determine stable long-living data gathering trees that do not require frequent reconfigurations. To the best of our knowledge, we have not come across any work on stable data gathering trees for mobile sensor networks.

In this research, we address the issue of finding a sequence of longest-living stable data gathering trees for mobile sensor networks such that the number of tree discoveries is the global minimum. We present a simple but powerful polynomial-time greedy algorithm, referred to as the Max.Stability-DG algorithm, to determine the sequence of longest-living stable data gathering trees. Given the complete knowledge of the future topology changes, the Max.Stability-DG algorithm operates based on the following greedy principle: Whenever a data gathering tree is required at time instant t, choose the longest-living data gathering tree from t. The above strategy is repeated over the duration of the data gathering session. The sequence of such longest-living data gathering trees is called the *Stable-Mobile-DG-Tree*. The worst-case run-time complexity of the Max.Stability-DG tree algorithm is $O(n^2T\log n)$ and $O(n^3T\log n)$ when operated under sufficient-energy and energy-constrained scenarios respectively, where n is the number of nodes in the network and T is the total number of rounds of data gathering; $O(n^2\log n)$ is the worst-case run-time complexity of the minimum-weight spanning tree algorithm (we use Prim's algorithm [5]) used to determine the underlying spanning trees from which the data gathering trees are derived.

The rest of the paper is organized as follows: Section 2 presents the system model and the terminology used in this research as well as a high-level overview of the working of the proposed Max.Stability-DG algorithm and highlights its key contributions. Section 3 presents related work on data gathering in mobile sensor networks. Section 4 describes in detail the working of the Max.Stability-DG algorithm, analyzes its run-time complexity for both sufficient-energy and energy-constrained scenarios, and provides a formal proof of correctness of the algorithm. We also present an algorithm to determine a minimum-distance spanning tree based data gathering (MST-DG) tree that has been observed (in previous research) to be the most energy-efficient approach [16] for data gathering in static sensor networks. We also present an example to illustrate the working of the Max.Stability-DG and MST-DG algorithms. Section 5 presents an exhaustive simulation study evaluating the performance of the Max.Stability-DG trees under diverse conditions of network dynamicity (node mobility and number of static nodes), network density (transmission range) and energy level at the nodes (sufficient-energy and energy-constrained scenarios). The performance metrics evaluated are the tree lifetime, delay per round, energy lost per node, energy lost per round, fairness of node usage, node lifetime, network lifetime, coverage loss time and fraction of coverage loss. We compare the performance of the Max.Stability-DG trees with that of the MST-DG trees. Section 6 presents the conclusions along with a summary of the simulation results. Section 7 discusses future work. For the rest of the paper, the terms 'node' and 'vertex', 'edge' and 'link', 'data aggregation' and 'data gathering' will be used interchangeably. They mean the same.

2 System model, terminology and overview

2.1 System model

The system model adopted in this research is as follows: (i) Each sensor node is assumed to operate with an identical and fixed *transmission range*. (ii) For the purpose of calculating the coverage loss, we also use the *sensing range* of a sensor node, considered in this research, as half the transmission range of the node. Basically, a sensor node can monitor and collect data at locations within the radius of its sensing range and transmit them to nodes within the radius of its transmission range. It has been proven in the literature [32] that the transmission range per node has to be at least twice the sensing range of the nodes to ensure that coverage implies connectivity. (iii) A data gathering tree is obtained by conducting a Breadth First Search (BFS) [5] of the spanning tree, starting from a root node that serves as the leader node for the tree. (iv) The leader node of a data gathering tree remains the same until the

tree exists and is randomly chosen each time a new tree needs to be determined. (v) Data gathering proceeds in rounds. During a round of data gathering, data gets aggregated starting from the leaf nodes of the tree and propagates all the way to the leader node. An intermediate node in the tree collects the aggregated data from its immediate child nodes and further aggregates with its own data before forwarding to its immediate parent node in the tree.

2.2 Terminology

We use the notions of static graphs and mobile graphs (adapted from [7]) to capture the sequence of topological changes in the network and determine a stable data gathering tree that spans over several time instants. A ***static graph*** is a snapshot of the network at any particular time instant and is modeled as a unit disk graph [11] wherein there exists a link between any two nodes if and only if the physical distance between the two end nodes of the link is less than or equal to the transmission range. The weight of an edge on a static graph is the Euclidean distance between the two end nodes of the edge. The Euclidean distance for a link $i - j$ between two nodes i and j, currently at (X_i, Y_i) and (X_j, Y_j) is given by: $\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2}$.

A ***mobile graph*** $G(i, j)$, where $1 \leq i \leq j \leq T$, where T is the total number of rounds of the data gathering session corresponding to the network lifetime, is defined as $G_i \cap G_{i+1} \cap \ldots G_j$. Thus, a mobile graph is a logical graph that captures the presence or absence of edges in the individual static graphs. In this research work, we sample the network topology periodically for every round of data gathering to obtain the sequence of static graphs. The weight of an edge in the mobile graph $G(i, j)$ is the geometric mean of the weights of the edge in the individual static graphs spanning $G_i, \ldots, G_j$. Since there exist an edge in a mobile graph if and only if the edge exists in the corresponding individual static graphs, the geometric mean of these Euclidean distances would also be within the transmission range of the two end nodes for the entire duration spanned by the mobile graph. Note that at any time, a mobile graph includes only ***live sensor nodes***, nodes that have positive available energy.

A ***static spanning tree*** is a minimum-weight spanning tree determined on a static graph. Since we use the Euclidean distance between the constituent nodes of an edge as the link weight, the minimum-weight spanning tree determined on a static graph will be a minimum-distance spanning tree for which the sum of the edge weights will be the minimum. A ***static data gathering tree*** is a rooted form of its corresponding static spanning tree with the root node being the leader node chosen for the round corresponding to the time instant represented by the static spanning tree. A ***mobile spanning tree*** is a minimum-weight spanning tree determined on a mobile graph whose edge weights are the geometric mean of the corresponding edge weights in the constituent static graphs. A ***mobile data gathering tree*** is a rooted mobile spanning tree whose root node is the leader node chosen at the beginning time instant of the corresponding mobile graph. The leader node of a mobile data gathering tree remains the same until the mobile graph gets disconnected due to node mobility or a node failure occurs, whichever happens first.

2.3 Overview of the maximum stability based data gathering algorithm and key contributions

A high-level overview of the working of the Max.Stability-DG algorithm is as follows: To determine a stable data gathering at time instant t_i ($1 \leq i \leq T$, the total number of rounds of the data gathering session), we determine the mobile graph $G(i, j)$, where $i \leq j$ such that there exists a spanning tree of the sensor nodes in $G(i, j)$ and not in $G(i, j+1)$. We transform such a longest-living spanning tree existing in each of the static graphs of the mobile graph $G(i, j)$ to a data gathering tree by simply running a breadth-first search (BFS) algorithm [5] starting from an arbitrarily chosen root node (also called the leader node). The data gathering tree rooted at the leader node is used for all the rounds from time instants t_i to t_j, which is considered as the lifetime of the spanning tree. The above procedure is repeated until the end of the data gathering session or the network lifetime, as appropriate. Any spanning tree algorithm can be used to determine the spanning tree on the mobile graph. In this research, we use the Prim's $O(n^2 * \log n)$ algorithm to determine a minimum-weight spanning tree on the mobile graph (of n nodes) whose edge weights are modeled as the geometric mean of the edges in the constituent static graphs.

A ***key assumption*** in this research is that the entire sequence of network topology changes is known beforehand at the time of running the Max.Stability-DG algorithm. This is required to generate the mobile graph spanning several static graphs, each representing snapshots of the network topology at time instants corresponding to successive rounds of data gathering, on which a stable long-living data gathering tree will be determined. The above assumption may not be practical for distributed systems of sensor networks. However, note that our goal in this research is not to develop a distributed algorithm for data gathering; but, to develop a ***benchmarking algorithm*** that can give us the sequence of long-living data gathering trees (over the duration of the data gathering session) whose ***lifetime will be the upper bound for the data gathering trees*** obtained using any other algorithm developed for this problem in the area of mobile sensor networks. The sequence of such stable longest-living data gathering trees determined using the Max.Stability-DG algorithm will involve the minimum number of discoveries. Thus, the number of data gathering tree discoveries incurred with the Max.Stability-DG algorithm ***will form the lower bound for the number of data gathering tree discoveries*** incurred with any other algorithm for mobile sensor networks.

The proposed algorithm is very generic in nature and it can be used to determine a sequence of stable

communication topologies of any type (for example, a connected dominating set [17], a chain [12], a cluster [8] etc) as long as there is an underlying algorithm to determine that communication topology on a given graph. In this research, we focus only on spanning tree as the communication topology for data gathering. Moreover, since the Max.Stability-DG trees are spanning-tree based and a spanning tree exists in a network if and only if the underlying network is connected, the stability of network-wide communication topologies (like a connected dominating set [17] that spans all the nodes) determined by any algorithm can be evaluated by comparing their lifetime with that obtained for the Max.Stability-DG trees under identical operating conditions. Henceforth, the relative stability of data gathering trees or any network-wide communication topology for mobile sensor networks, determined from any existing or newly proposed algorithm (very few of which is currently available in the literature, as reviewed in Section 3), either centralized or distributed, can be evaluated in comparison with the mobile data gathering trees obtained by running the Max.Stability-DG algorithm, developed in this research, under the same conditions in which the existing or prospective data gathering algorithm is run.

3 Related work on data gathering in wireless mobile sensor networks

The research on mobile sensor networks started with the deployment of mobile sink nodes on a network of static sensor nodes. A common approach of data gathering in such environments is to employ a mobile data collecting agent (e.g., [30][31][34]) that goes around the network in the shortest possible path towards the location from which the desired data is perceived to originate. In [35], the authors propose a distributed algorithm to optimize both coverage control and mobile collection using a Bayesian occupancy grid mapping technique that recursively updates the locations of potential data sources. In [18], the authors propose a 2-layer architecture comprising of mobile sinks and static sensor nodes for large scale wireless sensor networks. The top layer is a mobile ad hoc network of resource-rich sink nodes while the bottom-layer is a network of static resource-constrained sensor nodes. Each sink node is assigned a particular region to monitor and collect data. A sink node moves to the vicinity of the sensor nodes (within a few hops) to collect data. The collected data is exchanged with peer mobile sinks. A prototype implementation of the same is available in [19].

Very few topology-based data gathering algorithms have been proposed for mobile sensor networks where the sensor nodes actually move. Among these, most of the work is focused around the use of clusters wherein researchers have tried to extend the classical LEACH (Low Energy Adaptive Clustering Hierarchy) [8] algorithm for dynamically changing network topologies. Variants of LEACH for WMSNs that have been proposed in the literature include those that take into consideration the available energy level [2] and the mobility-level [23] of the nodes to decide on the choice of cluster heads; stability of the links between a regular node and its cluster head [6]; as well as set up a panel of cluster heads to facilitate cluster reconfiguration in the presence of node mobility [24]. In [13], the authors propose a distributed cluster-head based algorithm in which cluster-heads are elected based on node IDs (0 to C-1, C to 2C-1 …, to operate with C clusters at a time) or node locations (nodes that are closest to certain landmarks with in a WMSN serve as the cluster-heads). In [15], the authors investigate the use of a directed acyclic graph as the underlying communication topology of a sensor network field, modeled according to the theory of thermal fields, to form propagation paths such that the temperature of the nodes on the path increases as data progresses towards the sink, which is considered to be the warmest.

The only tree-based data gathering algorithm we have come across for WMSNs is a shortest path-based spanning tree algorithm [25] wherein each sensor node is constrained to have at most a certain number of child nodes. Based on the results from the literature of mobile ad hoc networks (e.g., [20][21]), minimum hop shortest paths and trees in mobile network topologies are quite unstable and need to be frequently reconfigured. We could not find any other related work on tree-based data gathering for wireless mobile sensor networks.

4 Data gathering algorithms based on maximum stability and minimum-distance spanning trees

4.1 Maximum stability spanning tree-based data gathering (max.stability-DG) algorithm

The Max.Stability-DG algorithm is based on a greedy look-ahead principle and the intersection strategy of static graphs. When a mobile data gathering tree is required at a sampling time instant t_i, the strategy is to find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots G_j$ such that there exists a spanning tree in $G(i, j)$ and no spanning tree exists in $G(i, j+1) = G_i \cap G_{i+1} \cap \ldots G_j \cap G_{j+1}$. We find such an epoch t_i, …, t_j as follows: Once a mobile graph $G(i, j)$ is constructed with the edges assigned the weights corresponding to the geometric mean of the weights in the constituent static graphs G_i, G_{i+1}, …, G_j, we run the Prim's minimum-weight spanning tree algorithm on the mobile graph $G(i, j)$. If $G(i, j)$ is connected, we will be able to find a spanning tree in it. We repeat the above procedure until we reach a mobile graph $G(i, j+1)$ in which no spanning tree exists and there existed a spanning tree in $G(i, j)$. It implies that a spanning tree basically existed in each of the static graphs G_i, G_{i+1}, ..., G_j and we refer to it as the mobile spanning tree for the time instants t_i, …, t_j. To obtain the corresponding mobile data gathering tree, we choose an arbitrary root node for this mobile spanning tree and run the Breadth First Search (BFS) algorithm on it starting

from the root node. The direction of the edges in the spanning tree and the parent-child relationships are set as we traverse its vertices using BFS. The resulting mobile data gathering tree with the chosen root node (as the leader node) is used for every round of data gathering spanning time instants t_i, …, t_j. We then set $i = j+1$ and repeat the above procedure to find a mobile spanning tree and its corresponding mobile data gathering tree that exists for the maximum amount of time since t_{j+1}. A sequence of such maximum lifetime (i.e., longest-living) mobile data gathering trees over the timescale T corresponding to the number of rounds of a data gathering session is referred to as the ***Stable Mobile Data Gathering Tree***. Figure 1 presents the pseudo code of the Max.Stability-DG algorithm that takes as input the sequence of static graphs spanning the entire duration of the data gathering session.

Input: Sequence of static graphs G_1, G_2, ... G_T; Total number of rounds of the data gathering session – T
Output: *Stable-Mobile-DG-Tree*
Auxiliary Variables: i, j
Initialization: $i = 1$; $j = 1$; *Stable-Mobile-DG-Tree* = Φ

Begin *Max.Stability-DG Algorithm*

1 **while** ($i \leq T$) **do**

2 Find a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots \cap G_j$ such that there exists at least one spanning tree in $G(i, j)$ and {no spanning tree exists in $G(i, j+1)$ or $j = T$}

3 *Mobile-Spanning-Tree*(i, j) = **Prim's Alg** ($G(i, j)$)

4 *Root*(i, j) = Choose a node randomly in $G(i, j)$

5 *Mobile-DG-Tree*(i, j) = **Breadth First Search** (*Mobile-Spanning-Tree*(i, j), *Root*(i, j))

6 *Stable-Mobile-DG-Tree* = *Stable-Mobile-DG-Tree* U { *Mobile-DG-Tree*(i, j) }

7 **for** each time instant $t_k \in \{t_i, t_{i+1}, \ldots, t_j\}$ **do**
Use the *Mobile-DG-Tree*(i, j) in t_k

8 **if** node failure occurs at t_k **then**
$j = k - 1$
break
end if
end for

9 $i = j + 1$

10 **end while**

11 **return** *Stable-Mobile-DG-Tree*

End *Max.Stability-DG Algorithm*

Figure 1: Pseudo Code for the Maximum Stability-based Data Gathering Tree Algorithm

While operating the algorithm under energy-constrained scenarios, one or more sensor nodes may die due to exhaustion of battery charge even though the underlying spanning tree may topologically exist. For example, if we have determined a data gathering tree spanning across time instants t_i to t_j using the above approach, and we come across a time instant t_k ($i \leq k \leq j$) at which a node in the tree fails, we simply restart the Max.Stability-DG algorithm starting from time instant t_k considering only the live sensor nodes (i.e., the sensor nodes that have positive available energy) and determine the longest-living data gathering tree that spans all the live sensor nodes since t_k. The pseudo code of the Max.Stability-DG algorithm in Figure 1 handles node failures, when run under energy-constrained scenarios, through the ***if*** block segment in statement 8. If all nodes have sufficient-energy and there are no node failures, the algorithm does not execute statement 8.

4.2 Run-time complexity Analysis af the max.stability-DG algorithm

To expand a mobile graph $G(i, j) = G_i \cap G_{i+1} \cap \ldots G_j$ to $G(i, j+1)$, all we had to do is to check whether each of the edges in the mobile graph $G(i, j)$ existed at time instant t_{j+1}. This can be done in $O(n^2)$ time on a mobile graph of n nodes, as there can be at most $O(n^2)$ edges on a graph of n vertices. The overall complexity of the Max.Stability-DG algorithm is the sum of the time complexity to construct the mobile graphs, the time complexity to run the spanning tree algorithm on these mobile graphs and the time complexity to transform these spanning trees to data gathering trees using BFS.

Sufficient-energy Scenarios: When the network operates under sufficient-energy scenarios (i.e., no node failures), for a data gathering session comprising of T rounds, we will have to construct T mobile graphs, resulting in a time complexity of $O(n^2T)$ to construct the mobile graphs. On each of these T mobile graphs, we will have to run a spanning tree algorithm. If we use the $O(n^2 * \log n)$ Prim's algorithm to construct a spanning tree, the complexity to run the spanning tree algorithm on the T mobile graphs becomes $O(n^2 * \log n * T)$. A spanning tree on n vertices has n–1 edges. The time-complexity of running BFS on a spanning tree of n vertices with n–1 edges is $O(n)$ [5]. To run BFS on the $O(T)$ spanning trees, we incur $O(nT)$ time. Thus, the overall complexity of the Max.Stability-DG algorithm under sufficient-energy scenarios is $O(n^2T) + O(n^2T\log n) + O(nT) = O(n^2T\log n)$.

Energy-Constrained Scenarios: There can be at most n–1 node failures (on an n node network) that trigger the execution of statement 8 in the pseudo code of Figure 1 for the Max.Stability-DG algorithm. A node failure occurring at time instant t_k ($i \leq k \leq j$), while using a mobile data gathering tree that has been determined on a mobile graph for time instants t_i, …, t_j, would require us to construct a mobile graph starting from t_k and the number of mobile graphs that we have to construct and run the spanning tree algorithm increases by $j–k+1$. At the worst case, if there are n–1 node failures, the number of mobile graphs that we have to construct and run the spanning tree algorithm increases by $(T – 1) + (T – 2) +$

$(T - (n-1)) = (n-1)T - [1 + 2 + \ldots + (n-1)] = O(nT) + O(n^2)$. Under the sufficient-energy scenarios, we had to construct T mobile graphs and run the spanning tree algorithm on each of them. In the energy-constrained scenarios, we will have to construct at most $T + O(nT) + O(n^2)$ mobile graphs and run the spanning tree algorithm on each of them. The number of rounds of data gathering is generally far greater than the number of nodes in the network. For example, in our simulations, we use a value of T = 4000 rounds (4 rounds per second, for 1000 seconds) and n = 100 nodes. Thus, since $n << T$, we can say that $n^2 << nT$. Therefore, a total of $T + O(nT) + O(n^2) = T + O(nT) = O(nT)$ mobile graphs are constructed. The time complexity to construct these mobile graphs is $O(n^2 * nT) = O(n^3T)$. We run the $O(n^2 \log n)$ Prim's spanning tree algorithm on the $O(nT)$ mobile graphs, resulting in a time-complexity of $O(n^3T\log n)$ to determine the spanning trees. The time-complexity of running the $O(n)$-BFS algorithm on the $O(nT)$ spanning trees is $O(n^2T)$. Thus, the overall time-complexity of the Max.Stability-DG algorithm under the energy-constrained scenarios is $O(n^3T) + O(n^3T\log n) + O(n^2T) = O(n^3T\log n)$.

4.3 Minimum-distance spanning tree based data gathering algorithm

In Section 5 (Simulations), we compare the performance of the Max.Stability-DG trees with that of the minimum-distance spanning tree based data gathering (MST-DG) trees. The sequence of MST-DG trees for the duration of the data gathering session is generated as follows: If a MST-DG tree is not known for a particular round, we run the Prim's minimum-weight spanning tree algorithm on the static graph representing the snapshot of the network topology generated at the time instant corresponding to the round. Since the weights of the edges in a static graph represent the physical Euclidean distance between the constituent end nodes of the edges, the Prim's algorithm will return the minimum-distance spanning tree on the static graph. We then choose an arbitrary root node and run the Breadth First Search (BFS) algorithm starting from this node. The MST-DG tree is the rooted form of the minimum-distance spanning tree with the chosen root node as the leader node. We continue to use the MST-DG tree as long as it exists. The leader node of the MST-DG tree remains the same until the tree breaks due to node mobility or node failures. When the MST-DG tree ceases to exist for a round, we repeat the above procedure. This way, we generate a sequence of MST-DG trees, referred to as the ***MST Mobile Data Gathering Tree***. The MST-DG algorithm emulates the general strategy (referred to as Least Overhead Routing Approach, LORA [1]) of routing protocols and data gathering algorithms for ad hoc networks and sensor networks. That is, the algorithm chooses a data gathering tree that appears to be the best at the current time instant and continues to use it as long as it exists. In a recent work [16], the authors have observed the minimum-distance spanning tree-based data gathering trees to be the most energy-efficient communication topology for data gathering in static sensor networks.

To be fair to the Max.Stability-DG algorithm that is proposed and evaluated in this research, the MST-DG algorithm is also run in a centralized fashion with the assumption that the entire static graph information is available at the beginning of each round. The time-complexity of generating the sequence of MST-DG trees on a network of n nodes for a total of T rounds for the data gathering session is $O(n^2T\log n)$ for both the sufficient-energy and energy-constrained scenarios. The time-complexity of the MST-DG algorithm remains the same for both the sufficient-energy and energy-constrained scenarios; because, we do not need to backtrack on the sequence of static graphs upon node failure and repeat the algorithm more than once on a static graph.

4.4 Example

We run both the Max.Stability-DG and MST-DG algorithms on the same sequence of static graphs $G_1G_2G_3G_4G_5$ (shown in the first part of Figures 2 and 3), generated by sampling the network topology for every second. For simplicity in the representation, we do not use weights for the edges. In both Figures 2 and 3, one could assume that the spanning trees determined on the static graphs and mobile graphs at different instances of execution of the algorithms are the minimum-weight spanning trees on the corresponding graphs.

In Figure 2, we could find a connected mobile graph spanning G_1, G_2 and G_3; and could not find a connected mobile graph from G_1 through G_4. A spanning tree exists for a graph if and only if the graph is connected. We determine a spanning tree on $G_1 \cap G_2 \cap G_3$ and derive a data gathering tree rooted at an arbitrarily selected node (node 3). This stable data gathering tree is to be used for the rounds corresponding to time instants of the static graphs G_1, G_2 and G_3. Similarly, we continue with the subsequent two static graphs and find a data gathering tree (with an arbitrary root node – node 6) that exists in both G_4 and G_5. Thus, we require two tree discoveries for the sequence of static graphs $G_1G_2G_3G_4G_5$.

We apply the MST-DG algorithm on the same sequence of static graphs $G_1G_2G_3G_4G_5$ (Figure 3). Note that the MST-DG algorithm chooses a spanning tree that appears to be the best at the time of needing a new data gathering tree. With this locally optimal strategy, we observe that we need to use a total of four data gathering trees (one for G_1 and G_2; and one each for G_3, G_4 and G_5); thus, requiring four tree discoveries for the sequence of static graphs $G_1G_2G_3G_4G_5$. We observe similar trends on the lifetime of the MST-DG trees in the simulations. Such a behaviour is not just a characteristic of MST-DG trees. We conjecture that any non-stability based data gathering algorithm that does not take into consideration the mobility of the nodes and chooses a data gathering tree that appears to be locally optimal with respect to any other metric (like energy consumption, delay etc.) will end up determining data gathering trees that require to be frequently reconfigured in the presence of a dynamically changing topology, characteristic of mobile sensor networks.

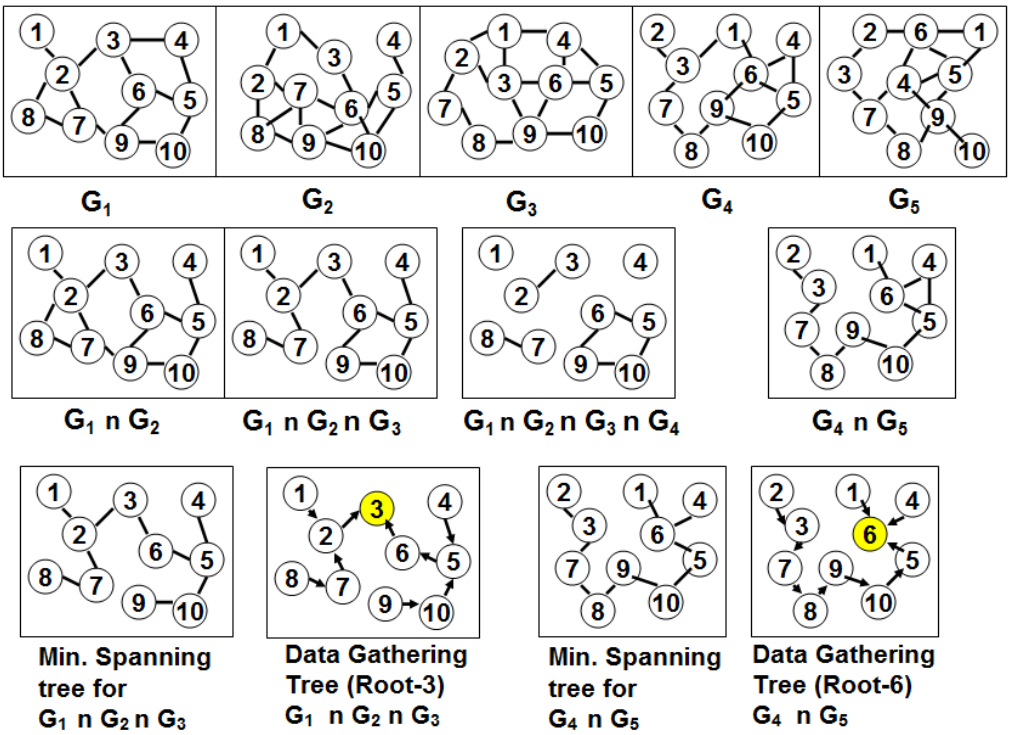


Figure 2: Example to Illustrate the Execution of the Maximum Stability Spanning Tree-based Data Gathering Tree Algorithm that uses the Globally Optimal Approach

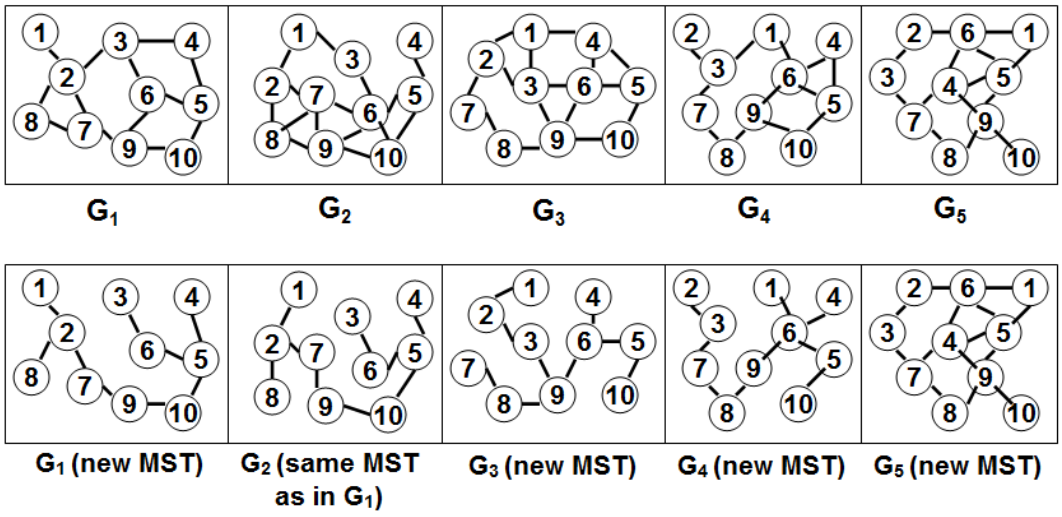


Figure 3: Example to Illustrate the Execution of the Minimum-distance Spanning Tree based Data Gathering Algorithm that uses the Locally Optimal Approach

4.5 Proof of correctness of the maximum stability-based data gathering algorithm

In this section, we prove that the Max.Stability-DG algorithm does determine the sequence of long-living stable mobile data gathering trees such that the number of tree discoveries is the global minimum (i.e. optimum). We use the approach of Proof by Contradiction. Let m be the number of data gathering tree discoveries incurred using the Max.Stability-DG algorithm on a sequence of static graphs G_1G_2 ... G_T. Let there be another algorithm (a hypothetical algorithm) that returns a sequence of mobile data gathering trees for the same sequence of static graphs such that the number of tree discoveries is $n < m$. If such an algorithm exists, then without loss of generality, there has to be one mobile data gathering tree, determined using this hypothetical algorithm, existing for the entire duration of a mobile graph $G(p, s)$; whereas, the Max.Stability-DG algorithm had to have at least one data gathering tree transition in $G(p, s)$. However, there cannot be such a data gathering tree that spanned through the entire mobile graph $G(p, s)$ and was not discovered by the Max.Stability-DG algorithm. Because, the Max.Stability-DG algorithm takes intersection of the

static graphs $G_p \cap G_{p+1} \cap \ldots G_s$ and runs a spanning tree algorithm on the intersection graph $G(p, s)$ – if at all a spanning tree existed in $G(p, s)$, then $G(p, s)$ would have to be connected. If the Max.Stability-DG algorithm could not determine a spanning tree/data gathering tree for the mobile graph $G(p, s)$, it implies the mobile graph $G(p, s)$ is not connected. It is not possible for any algorithm, including our hypothetical algorithm, to find a spanning tree/data gathering tree that covers all the vertices of a disconnected graph. Thus, the hypothetical algorithm would also had to have at least one tree transition in $G(p, s)$. The above proof holds good for any value of static graph indices p and s, where $1 \leq p \leq s \leq T$, and T is the total number of rounds corresponding to the duration of the data gathering session. Thus, the number of data gathering tree discoveries incurred with using the Max.Stability-DG algorithm is the global minimum.

Note that in the above proof, we have implicitly assumed that all the sensor nodes are alive for the entire duration of the data gathering session. In other words, we have proved that when operated under sufficient-energy scenarios, the Max.Stability-DG algorithm returns the stable sequence of data gathering trees such that the number of tree discoveries is the global minimum. It is not possible to theoretically prove the optimality of the Max.Stability-DG algorithm under energy-constrained scenarios. One can only validate the optimality of the lifetime of the Max.Stability-DG trees under energy-constrained scenarios through simulations, as we do in Section 5, wherein we observe the Max.Stability-DG trees to yield a significantly longer lifetime compared to the MST-DG trees under energy-constrained scenarios.

5 Simulations

In this section, we present an exhaustive simulation study on the performance of the Max.Stability-DG trees and compare them with that of the MST-DG trees under diverse conditions of network density and mobility. The simulations are conducted in a discrete-event simulator developed (in Java) by us exclusively for data gathering in mobile sensor networks. The MAC (medium access control) layer is assumed to be collision-free and considered an ideal channel without any interference. We opine that the use of any MAC-scheme proposed for energy-efficient and low-latency tree-based data gathering in sensor networks [14] can only be complementary to the performance of our benchmarking algorithm when implemented in a distributed context. Sensor nodes are assumed to be both TDMA (Time Division Multiple Access) and CDMA (Code Division Multiple Access)-enabled [28]. Every upstream node broadcasts a time schedule (for data gathering) to its immediate downstream nodes; a downstream node transmits its data to the upstream node according to this schedule. Such a TDMA-based communication between every upstream node and its immediate downstream child nodes can occur in parallel, with each upstream node using a unique CDMA code.

The network dimension is 100m x 100m. The number of nodes in the network is 100 and initially, the nodes are uniform-randomly distributed throughout the network. The sink is located at (50, 300), outside the network field. For a given simulation run, the transmission range per sensor node is fixed and is the same across all nodes. The network density is varied by varying the transmission range per sensor node from 20m to 50m, in increments of 5m. For brevity, we only present results obtained for transmission ranges per node of 25m and 30m (representative of moderate density, with connectivity of 97% and above), and for 40m (representative of high density, with 100% connectivity).

Simulations are conducted for two kinds of energy scenarios: One scenario wherein each node is provided with abundant supply of energy (50 J per node) and there are no node failures due to exhaustion of battery charge; the simulations in these ***sufficient-energy scenarios*** are conducted for 1000 seconds. The second scenario is an ***energy-constrained scenario*** in which each node is supplied with limited initial energy (2 J per node) and the simulations are conducted until the network of live sensor nodes gets disconnected due to the failures of one or more nodes.

The energy consumption model used is a first order radio model [22] that has been also used in several of the well-known previous work (e.g., [8][12]) in the literature. According to this model, the energy expended by a radio to run the transmitter or receiver circuitry is E_{elec} = 50 nJ/bit and $\in_{amp}$ = 100 pJ/bit/m^2 for the transmitter amplifier. The radios are turned off when a node wants to avoid receiving unintended transmissions. The energy lost in transmitting a k-bit message over a distance d is given by: $E_{TX}(k, d) = E_{elec} * k + \in_{amp} * k * d^2$. The energy lost to receive a k-bit message is: $E_{RX}(k) = E_{elec} * k$.

We conduct constant-bit rate data gathering at the rate of 4 rounds per second (one round for every 0.25 seconds). The size of the data packet is 2000 bits; the size of the control messages used for tree discoveries is assumed to be 400 bits. We assume that a tree discovery requires network-wide flooding of the 400-bit control messages such that each sensor node will broadcast the message exactly once in its neighborhood. As a result, each sensor node will lose energy to transmit the 400-bit message over its entire transmission range and receive the message from each of its neighbor nodes. In high density networks, the energy lost due to receipt of the redundant copies of the tree discovery control messages dominates the energy lost at a node for tree discovery. All of these mimic the energy loss observed for flooding-based tree discovery in ad hoc and sensor networks.

The node mobility model used is the well-known Random Waypoint mobility model [3] with the maximum node velocity being 3 m/s, 10 m/s and 20 m/s representing scenarios of low, moderate and high mobility respectively. According to this model, each node chooses a random target location to move with a velocity uniform-randomly chosen from [0,…, v_{max}], and after moving to the chosen destination location, the node continues to move by randomly choosing another new location and a new velocity. Each node continues to

move like this, independent of the other nodes and also independent of its mobility history, until the end of the simulation. For a given v_{max} value, we also vary the dynamicity of the network by conducting the simulations with a variable number of static nodes (out of the 100 nodes) in the network. The values for the number of static nodes used are: 0 (all nodes are mobile), 20, 50 and 80.

5.1 Performance metrics

We generated 200 mobility profiles of the network for a total duration of 6000 seconds, for every combination of the maximum node velocity and the number of static nodes. Every data point in the results presented in Figures 5 through 25 is averaged over these 200 mobility profiles. The tree lifetime and delay per round are measured for both the sufficient-energy and energy-constrained (appropriately prefixed as '*EC*' next to the names of the data gathering trees) scenarios. The trio of the energy consumption metrics – energy lost per round, energy lost per node and fairness of node usage are all measured only for the sufficient-energy scenarios in order to accurately capture the impact of the topological structure, network dynamicity and the stability of the data gathering trees on energy consumption. The node and network lifetimes as well as the fraction of coverage loss and coverage loss time are measured only for the energy-constrained scenarios.

The performance metrics measured in the simulations are:

(i) ***Tree Lifetime*** – the duration for which a data gathering tree existed, averaged over the entire simulation time period.

(ii) ***Delay per Round*** – measured in terms of the number of time slots needed per round of data aggregation at the intermediate nodes, all the way to the leader node of the data gathering tree, averaged across all the rounds of the simulation. A brief description of the algorithm used to compute the delay per round is given in Section 5.2 along with an illustration in Figure 4.

(iii) ***Energy Lost per Round*** – measured as the (sum of the energy lost due to the transmission and reception of data across all the links of a data gathering tree used for each round, the energy lost in transmitting the aggregated data from the leader node to the sink for each round plus the energy lost due to the network-wide flooding-based discovery of all the data gathering trees) divided by (the number of rounds of data gathering conducted on the network).

(iv) ***Energy Lost per Node*** – measured as the (sum of the energy lost at each node in the network due to transmission and reception of the data packets depending on their position in the data gathering trees used for the different rounds plus the energy lost due to broadcast transmission and reception of control messages in the neighborhood) divided by (the number of nodes in the network).

(v) ***Fairness of Node Usage*** – measured as the standard deviation (SD) of energy lost per node. The SD of energy lost per node should be ideally zero for maximum fairness of node usage. However, due to the stochastic nature of the network, nodes are not equally used. The lower the value for the SD of energy lost per node, the larger the fairness of node usage, and vice-versa.

(vi) ***Node Lifetime*** – measured as the time of first node failure due to exhaustion of battery charge.

(vii) ***Network Lifetime*** – measured as the time of disconnection of the network of live sensor nodes (i.e., the sensor nodes that have positive available battery charge), while the network would have stayed connected if all the nodes were alive at that time instant. So, before confirming whether an observed time instant is the network lifetime (at which the network of live sensor nodes is noticed to be disconnected), we test for connectivity of the underlying network if all the sensor nodes were alive.

We obtain the distribution of node failures as follows: The probability for '*x*' number of node failures (*x* from ranging from 1 to 100 as we have a total of 100 nodes in our network for all the simulations) for a given combination of the operating conditions (transmission range per node, maximum node velocity and number of static nodes) is measured as the number of mobility profile files that reported *x* number of node failures divided by 200, which is the total number of mobility profiles used for every combination of maximum node velocity and number of static nodes. Similarly, we keep track of the time at which '*x*' (*x* ranging from 1 to 100) number of node failures occurred in each of the 200 mobility profiles for a given combination of operating conditions and the values for the time of node failures reported in Figures 17, 18 and 19 are an average of these data collected over all the mobility profile files. We discuss the results for the distribution of the time and probability of node failures along with the discussion on node lifetime and network lifetime in Section 5.7.

(viii) ***Fraction of Coverage Loss and Coverage Loss Time***: If *f* is denoted as 'Fraction of Coverage Loss' (ranging from 0.01 to 1.0, measured in increments of 0.01), the coverage loss time is the time at which any *f* randomly chosen locations (X, Y co-ordinates) among 100 locations in the network is not within the sensing range of any node (explained in more detail below). Since the number of node failures increases monotonically with time and network coverage depends on the number of live nodes, our assumption in the calculations for network

coverage loss is that the fraction of coverage loss increases monotonically with time. We keep track of the largest fraction of coverage loss the network has incurred so far, and at the beginning of each round we check whether the network has incurred the next largest fraction of coverage loss, referred to as the *target fraction of coverage loss*. The first time instant during which we observe the network to have incurred the target coverage loss is recorded as the coverage loss time for the particular fraction of coverage loss, and from then on, we increment the target coverage loss by 0.01 and keep testing for the first occurrence of the new target fraction of coverage loss in the subsequent rounds. We repeat the above procedure until the network lifetime is encountered for the simulation with the individual data gathering algorithm.

At the beginning of each round, we check for network coverage as follows: We choose 100 random locations in the network and find out whether each of these locations is within the sensing range of at least one sensor node. We count the number of locations that are not within the sensing range of any node. If the fraction of the number of locations (actual number of locations that are not covered / total number of locations considered, which is 100) not within the sensing range of any node equals the target fraction of coverage loss, we record the time instant for that particular round of data gathering as the coverage loss time corresponding to the target fraction of coverage loss. We then increment the target fraction of coverage loss by 0.01 and repeat the above procedure to determine the coverage loss time corresponding to the new incremented value of the target fraction of coverage loss.

Each coverage loss time data point reported for particular fractions of coverage loss in Figures 23, 24 and 25 are the average values of the coverage loss times observed when the individual data gathering tree algorithms are run with the mobility profile files corresponding to a particular condition of network dynamicity (max. node velocity and number of static nodes) and transmission range per node. The probability for a particular fraction of coverage loss is computed as the ratio of the number of mobility profile files in which the corresponding fraction of coverage loss was observed divided by the total number of mobility profile files (200 mobility profile files for each operating condition).

5.2 Algorithm to compute the delay per round of data gathering

The delay incurred at a node is measured in terms of the number of time slots it takes to gather data from all of its immediate child nodes. The delay for the data gathering tree is one plus the delay incurred at the leader node (root node). We assume that it takes one time slot per child node to transfer data to its immediate predecessor node in the tree. However, a node cannot transfer the aggregated data to its parent node until it receives the data from its own child nodes. The delay calculations start from the bottom of the data gathering tree. The delay incurred at a leaf node is 0. To calculate the delay incurred at an intermediate node u, *Delay*(u), located at a particular level in the data gathering tree, we maintain a sorted list, *Child-Nodes*(u), of the delay associated with each of its immediate child nodes and use a temporary running variable *Temp-Delay*(u), initialized to zero, to explore the sorted list of the delays at the child nodes. For every child node $v \in$ *Child-Nodes*(u), *Temp-Delay*(u) = Maximum [*Temp-Delay*(u) + 1, *Delay*(v) + 1)], as we assume it takes one time slot for a child node to transfer its aggregated data to its immediate predecessor node in the tree. The delay associated with an intermediate node u, *Delay*(u), is the final value of the *Temp-Delay*(u) variable, after we iterate through the sorted list of the delays associated with the list *Child-Nodes*(u). The above procedure is repeated at all the intermediate nodes, from levels one less than the *Height* of the tree all the way to zero (i.e., the root node). We illustrate the working of the above explained procedure for delay computation on a data gathering tree through an example presented in Figure 4. The integer inside a circle indicates the node ID and the integer outside a circle indicates the delay for data aggregation at the node.

Nodes at Level	Nodes
0	12
1	7, 9
2	15, 2, 5, 14
3	3, 0, 6, 11, 13
4	10, 1, 4
5	8

Delay for the DG Tree
= 1 + Delay for the Root
= 1 + 6 = 7 time units

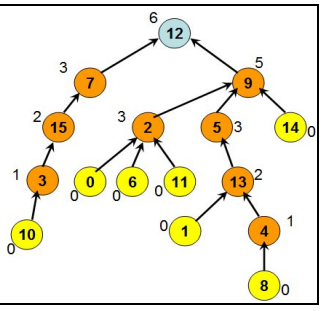
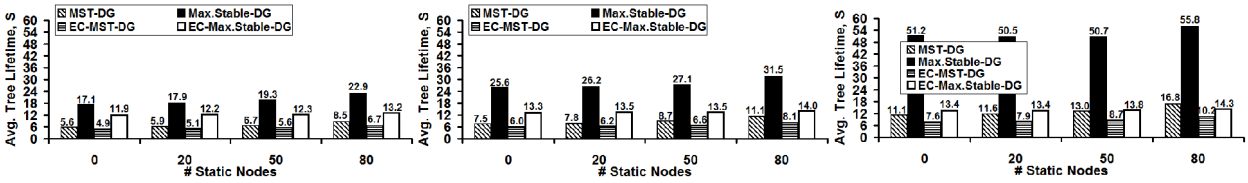


Figure 4: Example to Illustrate the Calculation of Delay per Round of Data Gathering.

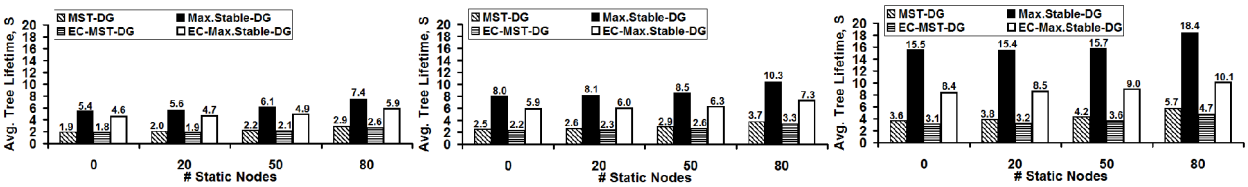
5.3 Tree lifetime

Among the three key operating parameters (maximum node velocity, number of static nodes and transmission range per node) of the simulations, we observe the stability of the data gathering trees to be highly influenced by the maximum node velocity (v_{max}) of the nodes. When operated under sufficient-energy scenarios, for a fixed number of static nodes and transmission range per node, we observe the lifetime incurred for both the Max.Stability-DG trees and MST-DG trees to proportionally decrease with a corresponding increase in the v_{max} values from 3 m/s to 10 m/s and further to 20 m/s. In the energy-constrained scenarios, even though a data gathering tree may topologically exist, the tree would require reconfiguration if one / more



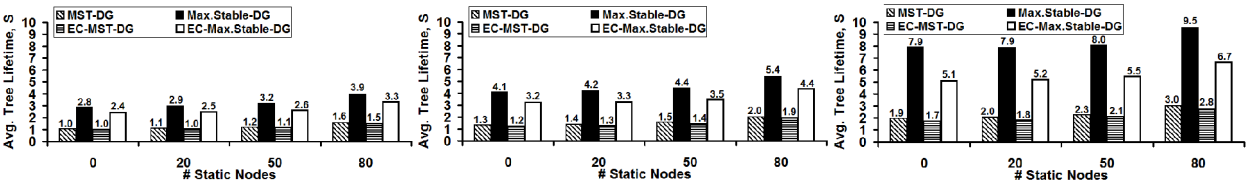
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 5: Average Tree Lifetime (Low Node Mobility: v_{max} = 3 m/s).



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 6: Average Tree Lifetime (Moderate Node Mobility: v_{max} = 10 m/s).



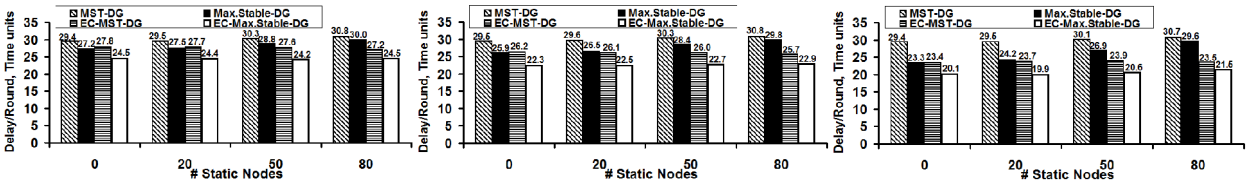
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 7: Average Tree Lifetime (High Node Mobility: v_{max} = 20 m/s).

nodes in the tree fail due to exhaustion of battery charge. Since a tree also needs to be reconfigured due to node mobility, the lifetime of the data gathering trees observed for energy-constrained scenarios is always less than or equal to that observed for sufficient-energy scenarios. In the case of both the Max.Stability-DG and MST-DG trees, for a fixed transmission range and # static nodes, we observe the largest difference between the tree lifetimes for the sufficient-energy scenarios vis-à-vis the energy-constrained scenarios to occur when the network is operated under low node mobility conditions (v_{max} = 3 m/s). This could be attributed to the significantly longer lifetime observed for the data gathering trees at low node mobility conditions when operated with sufficient-energy for the nodes.

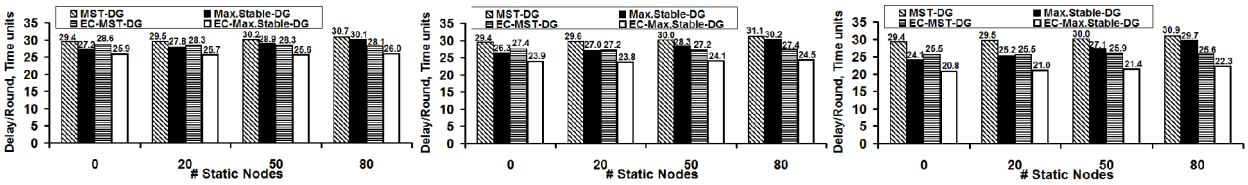
In low mobility scenarios (refer Figure 5), we also observe the difference in the tree lifetimes under sufficient-energy vs. energy-constrained scenarios to increase with increase in the transmission range per node. At higher transmission ranges, the links are more stable as nodes of a link have relatively higher freedom to move around (compared to operating at low and moderate transmission ranges) and still remain as neighbors. Hence, the data gathering trees are bound to be the most stable at low node mobility and larger transmission ranges per node. At these conditions – under sufficient-energy scenarios, we observe the Max.Stability-DG trees to sustain a lifetime that is larger than that of the MST-DG trees by a factor of about 3 to 4.5. However, under energy-constrained scenarios, when operated at low node velocity and larger transmission range per node, the Max.Stability-DG trees are only 50-75% more stable than that of the MST-DG Trees, even though the absolute magnitude of the tree lifetime incurred with both the trees is the maximum compared to the other operating conditions. On the contrary, larger differences between the lifetimes of the Max.Stability-DG trees and MST-DG trees under energy-constrained scenarios are observed when operated at moderate transmission ranges per node and the difference increases with increase in node mobility. This could be attributed to the significant energy savings sustained by the Max.Stability-DG algorithm with respect to tree discoveries under moderate and high node mobility levels (refer Figures 6 and 7). On the other hand, the MST-DG trees are quite unstable with increase in node mobility, resulting in frequent flooding-based tree discoveries that consume significant node energy. As a result, nodes on a Max.Stability-DG tree exist for a relatively much longer time compared to that of the MST-DG trees, contributing to the increasing difference in the lifetime of the two data gathering trees in energy-constrained scenarios when operated under moderate and high levels of node mobility.

With regards to the impact of the transmission range per node, the difference in the lifetime of the Max.Stability-DG trees and the MST-DG trees increases with increase in the transmission range per node, for a given level of node mobility. For a fixed v_{max} value, the lifetime of the Max.Stability-DG trees increases by a factor of 3 and above as we increase the transmission range from 25m to 40m; whereas the lifetime of the MST-DG trees increases only at most by a factor of 2.



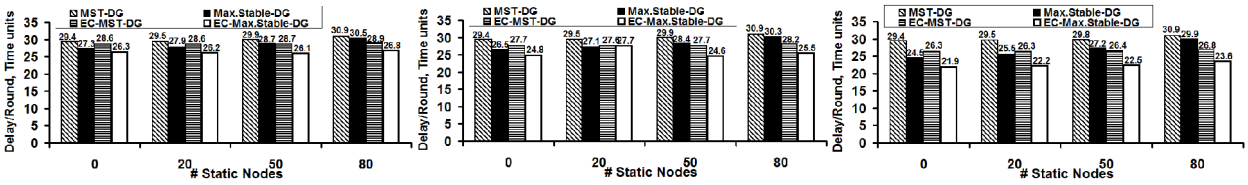
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 8: Average Delay per Round (Low Node Mobility: v_{max} = 3 m/s).



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 9: Average Delay per Round (Moderate Node Mobility: v_{max} = 10 m/s).



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 10: Average Delay per Round (High Node Mobility: v_{max} = 20 m/s).

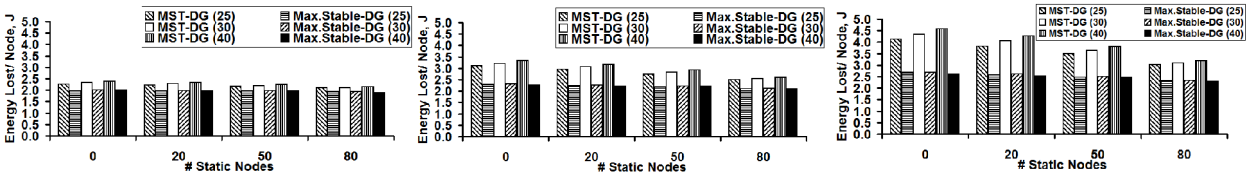
This could be again attributed to the optimal usage of the availability of stable links (facilitated by the larger transmission ranges per node) by the Max.Stability-DG algorithm through its look-ahead and graph intersection approach. However, as is the bane of the algorithms based on the local optimum approach, the MST-DG trees are formed with relatively less stable links even when operated with higher transmission ranges per node.

With regards to the impact of the number of static nodes, we observe that for both the sufficient-energy and energy-constrained scenarios, the lifetime of both the Max.Stability-DG trees and the MST-DG trees increases by about 50% when the number of static nodes is increased from 0 to 80 nodes. There is not much of a significant increase (only at most about 10-15% increase) in the lifetime of both the data gathering trees when we run the network with 20 and 50 static nodes instead of 0 nodes. This vindicates the impact of node mobility on the stability of the data gathering trees. Even if half of the nodes in the network are operated static, we observe the data gathering trees to have about the same vulnerability for a link failure vis-à-vis operating the network with all mobile nodes.

5.4 Delay per round

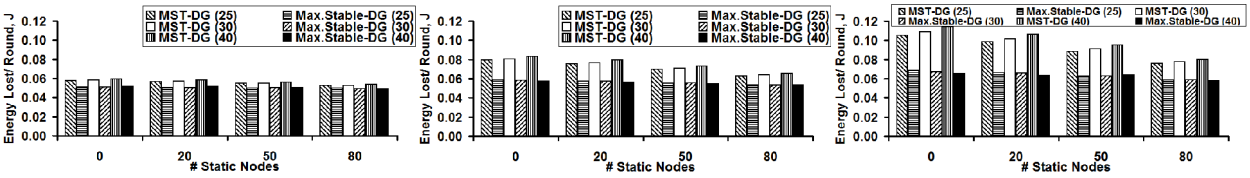
A minimum-distance based spanning tree tends to have relatively fewer leaf nodes, and as a result more nodes are likely to end up as intermediate nodes – leading to a much larger depth for the MST-DG trees. The MST-DG tree is also observed to be more unbalanced with respect to the distribution of the number of children per intermediate node as well as the distribution of the leaf nodes at different levels. Not all leaf nodes are located at the bottommost level of the tree. Due to all these structural complexities, the MST-DG trees have been observed to incur a much larger delay per round of data gathering. On the other hand, the Max.Stability-DG trees have been observed to be more shallow (i.e., lower depth) with more leaf nodes and the distribution of the number of child nodes per intermediate node is relatively more balanced. All of these factors contribute to a much lower delay per round of data gathering.

We observe the Max.Stability-DG trees to incur a lower delay per round of data gathering compared to that of the MST-DG trees under all operating conditions (the difference is as large as 25%). The delay per round is not much affected by the dynamicity of the network and is more impacted by the topological structure of the two spanning trees. We observe a relatively lower delay per round, especially for the MST-DG trees, at energy-constrained scenarios vis-à-vis sufficient-energy scenarios due to the decrease in the number of nodes and slightly better distribution of nodes when fewer in number. The delay per round for the Max.Stability-DG trees is influenced more by the transmission range per node in energy-constrained scenarios and by the number of static nodes in sufficient-energy scenarios. For a given transmission range per node and # static nodes, variations in the value of the maximum node velocity have the least impact on the delay per round for both the data gathering trees.



Low Mobility (v_{max} = 3 m/s) Moderate Mobility (v_{max} = 10 m/s) High Mobility (v_{max} = 20 m/s)

Figure 11: Average Energy Lost per Node.



Low Mobility (v_{max} = 3 m/s) Moderate Mobility (v_{max} = 10 m/s) High Mobility (v_{max} = 20 m/s)

Figure 12: Average Energy Lost per Round.

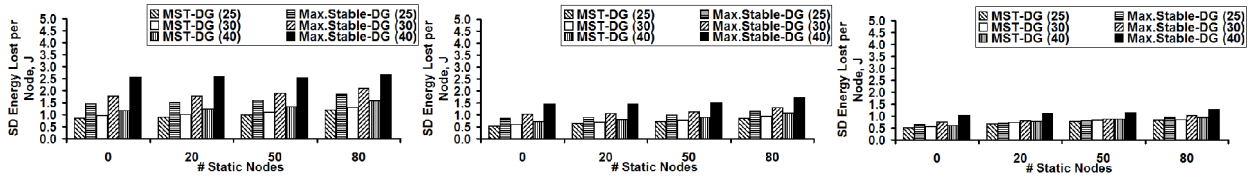
With node failures, the number of nodes in the data gathering trees decreases and this has a positive impact on the delay per round of data gathering. The decrease in the delay per round under energy-constrained scenarios ranges from 10-40%, with the larger percentage decrease in delay observed when the data gathering algorithms are operated with transmission range per node of 40m (observed for all node mobility levels) under energy-constrained scenarios compared to sufficient-energy scenarios. As a result, for a given energy scenario, the difference in the delay per round of the Max.Stability-DG trees and the MST-DG trees decreases with increase in the transmission range per node for a given node mobility.

For a given level of node mobility and transmission range per node, the delay per round for a data gathering tree, under both the sufficient-energy and energy-constrained scenarios, increases with increase in the number of static nodes. The increase is more predominantly observed for the Max.Stability-DG trees when operated at 80 static nodes under sufficient-energy scenarios. The Max.Stability-DG trees incurred about 10-25% lower delay than that of the MST-DG trees when all nodes are mobile. As the number of static nodes increases, the delay per round incurred with the Max.Stability-DG trees converges to that of the MST-DG trees, especially when operated under sufficient-energy scenarios. The relatively lower delay per round for the Max.Stability-DG trees under energy-constrained scenarios can be attributed to the decrease in the number of nodes coupled with the shallow topological structure of the data gathering tree.

5.5 Energy consumption

The energy lost per node and energy lost per round for the Max.Stability-DG trees are lower than that of the MST-DG trees for all the operating conditions. In this section, we combine the discussion for energy lost per node and energy lost per round as similar trends are observed for the two performance metrics (see Figures 11 and 12) with respect to the two data gathering trees under the different operating conditions. The maximum node velocity and the number of static nodes have been observed to have a significant impact on the energy lost per node and per round for the two data gathering trees. The lifetime of the MST-DG trees decreases significantly with increase in the maximum node velocity, requiring frequent network-wide flooding that consumes the energy level at the nodes. On the other hand, the Max.Stability-DG algorithm incurs significantly fewer tree discoveries and hence sustains a lower overall energy loss. The energy lost per node (and energy lost per round) for the MST-DG trees are about 10-20%, 15-45% and 30-75% more than that incurred for the Max.Stability-DG trees at conditions of low, moderate and high node mobility respectively. For a fixed v_{max} and transmission range per node, the difference in the energy lost per node (and energy lost per round) for the two data gathering trees decreases with increase in the number of static nodes. This is attributed to the relatively lower number of tree discoveries needed in the presence of static nodes and overall, the nodes incur a lower energy loss for the duration of the data gathering session. The decrease in the energy lost per node (and energy lost per round) with increase in the number of static nodes is significantly observed as we increase the maximum node velocity (about 5-10% decrease at v_{max} = 3 m/s and about 25%-35% decrease at v_{max} = 20 m/s).

For a given level of node mobility and number of static nodes, we observe the Max.Stability-DG trees to sustain almost similar values for the energy lost per node (and energy lost per round) as we increase the transmission range per node from 25m to 40m. Even though one can imagine that the nodes will lose more energy when operated under higher transmission range, the potential energy savings (obtained due to the reduced number of flooding-based tree discoveries attributed at larger transmission ranges per node) equally compensates for the increase in the transmission energy loss. Even the MST-DG trees have been observed to incur only about 5-10% increase in the energy lost per node (and energy lost per round) when operated at range



Low Mobility (v_{max} = 3 m/s) Moderate Mobility (v_{max} = 10 m/s) High Mobility (v_{max} = 20 m/s)

Figure 13: Fairness of Node Usage: Standard Deviation of Energy Lost per Node.

per node of 40m compared to 25m per node. Thus, among the three operating parameters – the transmission energy per node has the least impact on the energy lost per node and energy lost per round for the two data gathering trees.

5.6 Fairness of node usage

Ideally, all nodes are to be equally used. However, due to node mobility, node distribution and the varied roles for the nodes (leader node, intermediate node and leaf node) that change at different instants of the data gathering session depending on the topological stability of the data gathering tree used, the energy consumption across nodes is not uniform. As a result, certain nodes fail prematurely ahead of others.

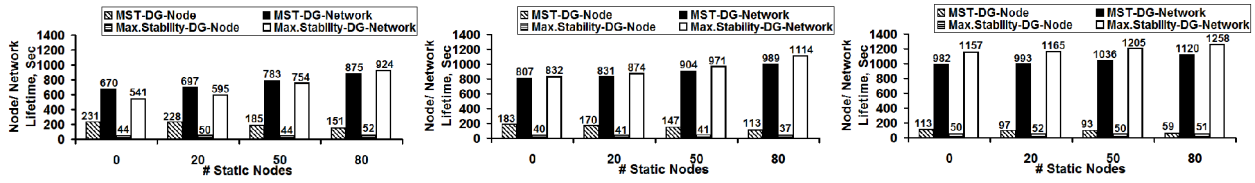
We observe the Max.Stability-DG trees to be unfair with respect to node usage. This could be attributed to the repeated use of certain nodes (the intermediate nodes and the leader node) of the stable data gathering tree for a longer time. This has a profound effect on the node lifetime (the time of first node failure) as observed in Section 5.7. However, the energy savings brought by reduced number of tree discoveries significantly compensates for the unfairness in node usage, leading to a larger network lifetime (the time the network of live nodes gets disconnected), as also observed in Section 5.7. Since the MST-DG trees are relatively more frequently reconfigured, we observe these data gathering trees to incur a lower standard deviation of energy lost per node under all simulation conditions. This plays a significant role in the MST-DG trees sustaining a relatively longer node lifetime (the time of first node failure). However, the relatively more equal, but higher energy lost per node for the MST-DG trees expedites node failures beyond the first node failure, eventually leading to a lower network lifetime than those incurred with the Max.Stability-DG trees.

For a given transmission range per node and number of static nodes, we observe the standard deviation of energy usage at the nodes to decrease with increase in maximum node velocity. Node mobility triggers more frequent tree reconfigurations and as a result, the chances of the role of the energy-consuming intermediate nodes and leader node gets rotated gets high, leading to increased fairness in node usage. On the other hand, the standard deviation of energy lost per node increases with increase in the transmission range per node (attributed to the increased stability at larger transmission ranges) as well as with increase in the number of static nodes (again attributed to the stability of the data gathering trees).

5.7 Node lifetime and network lifetime

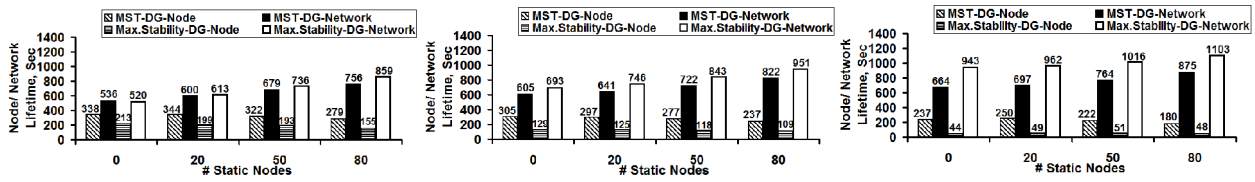
We observe a tradeoff between node lifetime and network lifetime for maximum stability vs. minimum-distance spanning tree based data gathering in mobile sensor networks. The MST-DG trees incur larger node lifetimes (the time of first node failure) for all the 48 operating combinations of maximum node velocity, number of static nodes and transmission range per node. The Max.Stability-DG trees incur larger network lifetime for most of the operating conditions. The lower node lifetime incurred with the Max.Stability-DG trees is attributed to the continued use of stable data gathering trees for a longer time and that too without changing the leader node. It would involve too much of message complexity and energy consumption to have the sensor nodes coordinate among themselves to choose a leader node for every round. Hence, we choose the leader node for a data gathering tree at the time of discovering it and let the leader node remain the same for the duration of the tree (i.e., until the tree fails). The same argument applies for the continued use of the intermediate nodes that receive aggregate data from one or more child nodes and transmit them to an upstream node in the tree. Due to the unfairness in node usage resulting from the overuse of certain nodes as intermediate nodes and leader node, the Max.Stability-DG trees have been observed to yield a lower node lifetime, especially under operating conditions (like low and moderate node mobility with moderate and larger transmission range per node) that facilitate greater stability. The node lifetime incurred with the Max.Stability-DG trees increases significantly with increase in the maximum node velocity, especially when operated in moderate transmission ranges per node.

The node lifetime incurred for the MST-DG trees can be larger than that of the Max.Stability-DG trees by as large as 400% at low and moderate levels of node mobility and by as large as 135% at higher levels of node mobility. For a given level of node mobility, the difference in the node lifetimes incurred for the MST-DG trees and Max.Stability-DG trees increases with increase in the transmission range per node (for a fixed number of static nodes) and either remain the same or slightly increase with increase in the number of static nodes (for a fixed transmission range per node). The MST-DG trees too suffer a decrease in node lifetime with increase in transmission range per node; but, at a lower scale – due to the relative instability of the trees. At larger transmission ranges per node, the data gathering trees are bound to be more stable, and the negative impact of this



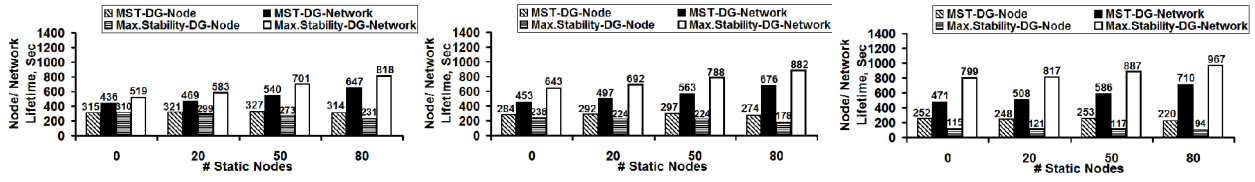
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 14: Average Node and Network Lifetime (Low Node Mobility: v_{max} = 3 m/s).



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 15: Average Node and Network Lifetime (Moderate Node Mobility: v_{max} = 10 m/s).



Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

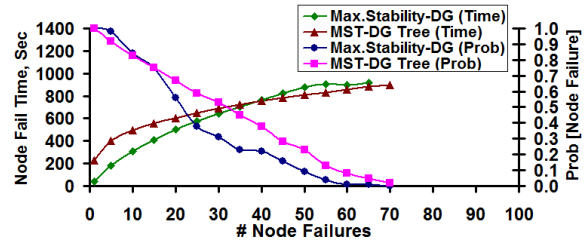
Figure 16: Average Node and Network Lifetime (High Node Mobility: v_{max} = 20 m/s).

on node lifetime is significantly felt in the case of the Max.Stability-DG trees. For a given transmission range per node, the negative impact associated with the use of static nodes on node lifetime is increasingly observed at v_{max} values of 3 m/s and 10 m/s. At v_{max} = 20 m/s, since the network topology changes dynamically, even the use of 80 static nodes is not likely to overuse certain nodes and result in their premature failures. The node lifetime incurred with MST-DG trees is more impacted with the use of static nodes at low node mobility scenarios (Figure 14) and the node lifetime incurred with the Max.Stability-DG trees is more impacted with the use of static nodes at moderate and higher node mobility scenarios (Figures 15 and 16).

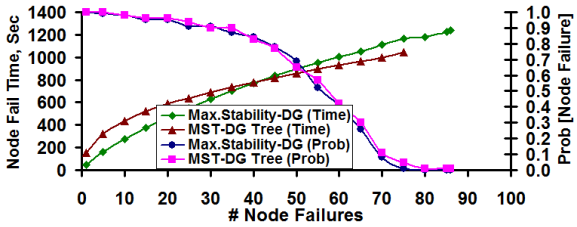
The Max.Stability-DG trees compensate for the premature failures of certain nodes by incurring a lower energy loss per round and energy loss per node due to lower tree discoveries and shorter tree height with more even distribution of the number of child nodes per intermediate node. As the dynamicity of the network increases, the data gathering trees become less stable, and this helps to rotate the roles of the intermediate nodes and leader node among the nodes to increase the fairness of node usage. All of these save significantly more energy at the remaining nodes that withstand the initial set of failures. As a result, we observe the Max.Stability-DG trees to observe a significantly longer network lifetime compared to that of the MST-DG trees.The difference in the network lifetime incurred for the Max.Stability-DG trees and that of the MST-DG trees increases with increase in the maximum node velocity and transmission range per node. For a given v_{max} and transmission range per node, the number of static nodes does not make a significant impact on the difference in the network lifetime incurred with the two data gathering trees, especially at moderate transmission ranges per node of 25 and 30m.

With respect the impact of the operating parameters on the absolute magnitude of the network lifetime, we observe the network lifetime incurred with the two data gathering trees increases with increase in the number of static nodes for a given value of v_{max} and transmission range per node. For a given level of node mobility, the network lifetime increases with increase in transmission range per node; however, for the MST-DG trees, the rate of increase decreases with increase in the maximum node velocity. This could be attributed to the relative instability of the MST-DG trees at high node mobility levels, requiring frequent tree reconfigurations. During a network-wide flooding, all nodes in the network tend to lose energy, almost equally. The Max.Stability-DG trees maintain a steady increase in the network lifetime with increase in transmission range per node for all levels of node mobility. For a given transmission range per node and number of static nodes, the network lifetime incurred for the two data gathering trees decreases with increase in the maximum node velocity, especially for the MST-DG trees due to their instability. This could be attributed to the energy loss incurred due to frequent tree discoveries.

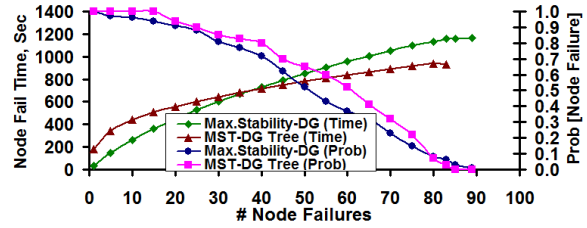
For a given transmission range per node, with the absolute values of the node lifetime increasing with increase in the maximum node velocity and the network lifetime decreasing with increase in the maximum node



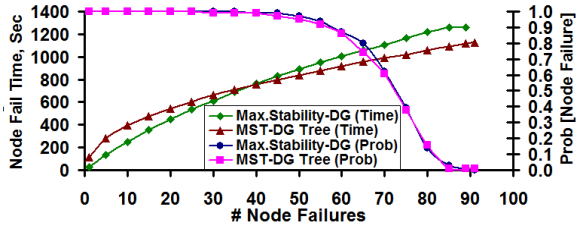
Transmission Range: 25 m, 0 static nodes



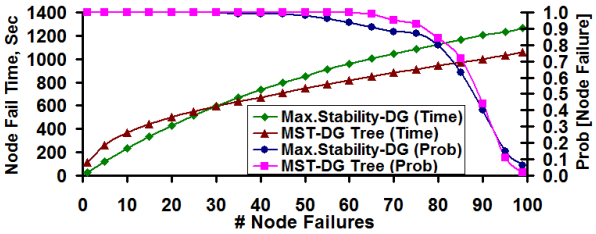
Transmission Range: 25 m, 80 static nodes



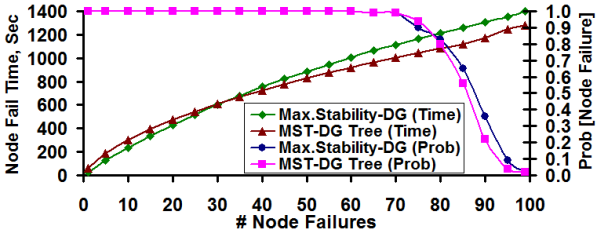
Transmission Range: 30 m, 0 static nodes



Transmission Range: 30 m, 80 static nodes

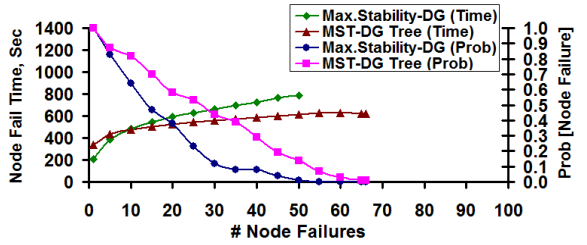


Transmission Range: 40 m, 0 static nodes

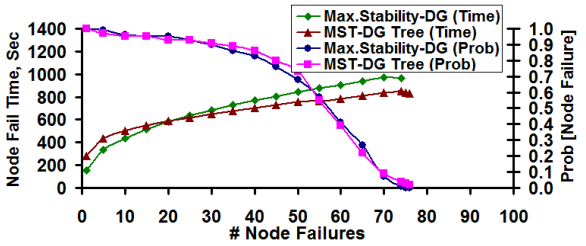


Transmission Range: 40 m, 80 static nodes

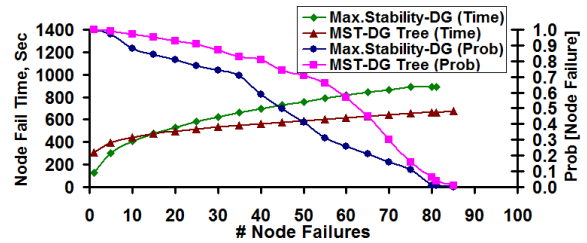
Figure 17: Distribution of Node Failure Times and Probability of Node Failures [v_{max} = 3 m/s].



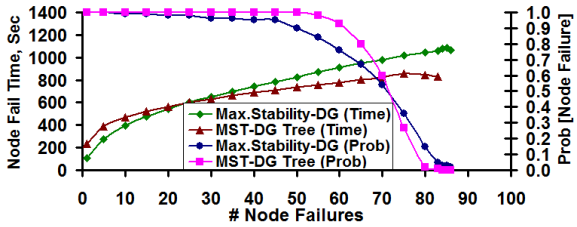
Transmission Range: 25 m, 0 static nodes



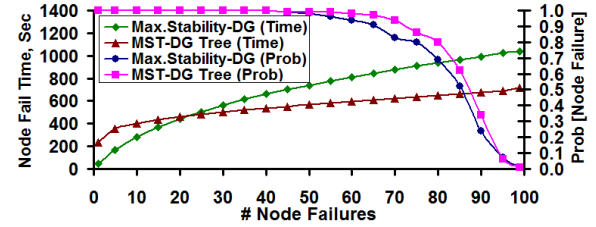
Transmission Range: 25 m, 80 static nodes



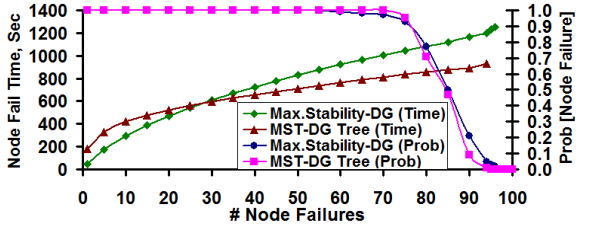
Transmission Range: 30 m, 0 static nodes



Transmission Range: 30 m, 80 static nodes

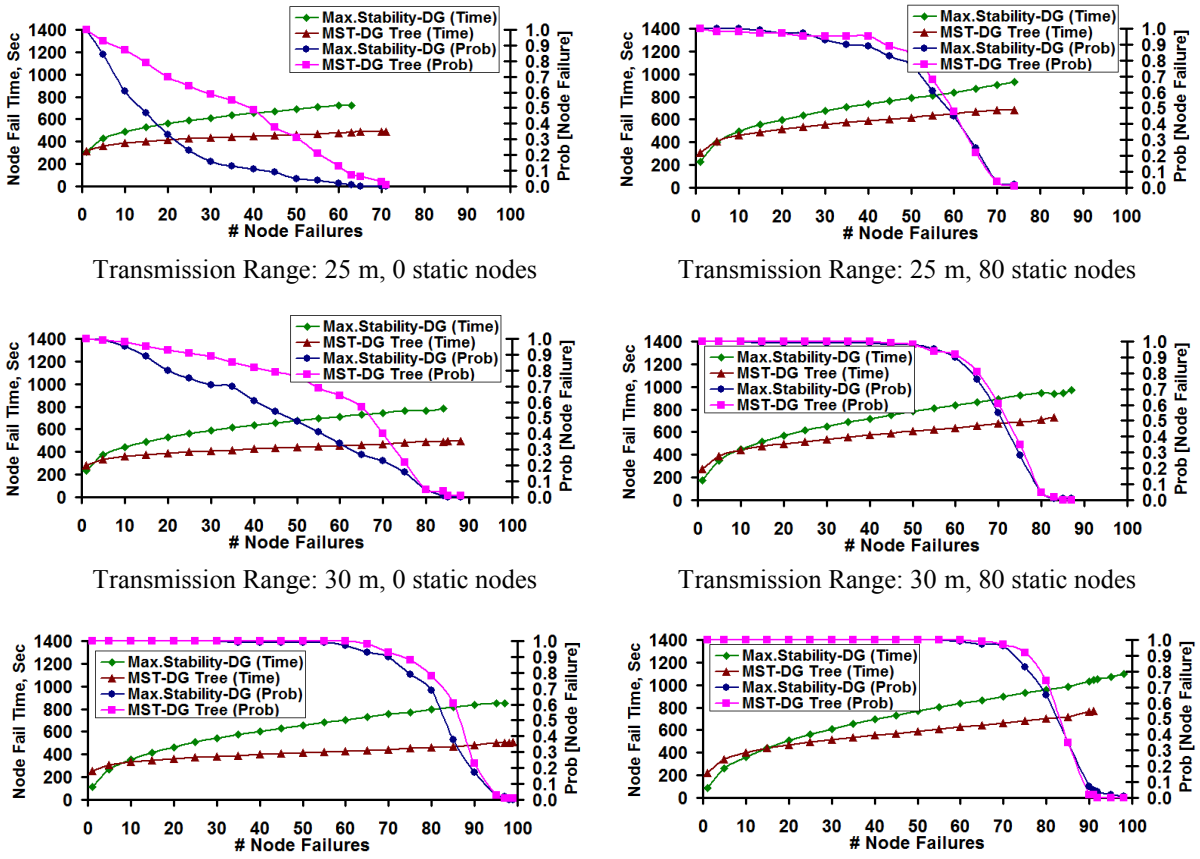


Transmission Range: 40 m, 0 static nodes



Transmission Range: 40 m, 80 static nodes

Figure 18: Distribution of Node Failure Times and Probability of Node Failures [v_{max} = 10 m/s].



Transmission Range: 25 m, 0 static nodes

Transmission Range: 25 m, 80 static nodes

Transmission Range: 30 m, 0 static nodes

Transmission Range: 30 m, 80 static nodes

Transmission Range: 40 m, 0 static nodes

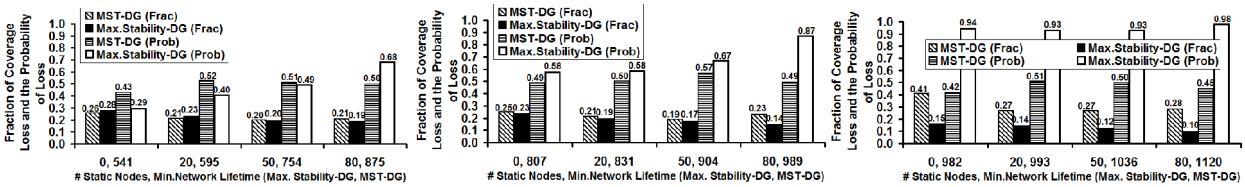
Transmission Range: 40 m, 80 static nodes

Figure 19: Distribution of Node Failure Times and Probability of Node Failures [v_{max} = 20 m/s].

velocity, we observe the maximum increase in the absolute time of node failures to occur at low node mobility. This vindicates the impact of network-wide flooding based tree discoveries on energy consumption at the nodes. Since all nodes are likely to lose the same amount of energy with flooding, the more we conduct flooding, the larger is the network-wide energy consumption. As a result, node failures tend to occur more frequently when we conduct frequent flooding. Thus, even though operating the network at moderate and high levels of node mobility helps us to extend the time of first node failure, the subsequent node failures occur too soon after the first node failure. This could be justified with the observation of flat curves for the MST-DG trees with respect to the distribution of node failure times (in Figures 17, 18 and 19). The distribution of node failure times is relatively steeper for the Max.Stability-DG trees. The unfair usage of nodes in the initial stages does help the Max.Stability-DG trees to prolong the network lifetime. Aided with node mobility, it is possible for certain energy-rich nodes (that might have been leaf nodes in an earlier data gathering tree) to keep the network connected for a longer time by serving as intermediate nodes, and the energy-deficient nodes serve as leaf nodes during the later rounds of data gathering.

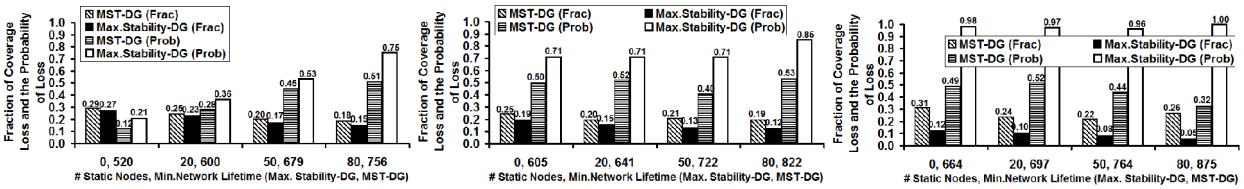
The impact of mobility in prolonging node failure lifetimes could also be explained by the lower probability of node failure observed for the Max.Stability-DG trees in comparison to the MST-DG trees when there are 0 static nodes (the plots to the left in Figures 17, 18 and 19). At 80 static nodes, the probability of node failures for the two data gathering trees is about the same and is higher than that observed when all nodes are mobile. This could be attributed to the repeated overuse of certain nodes as intermediate nodes and leader node on relatively more stable data gathering trees. Thus, with the use of static nodes, even though the absolute magnitude of the network lifetime can be marginally increased (by about 10-70%; the increase is larger at moderate transmission range per node and larger values of v_{max}), the probability of node failures to occur also increases.

In terms of the percentage difference in the values for the network lifetime and node lifetime incurred with the two data gathering trees, we observe the Max.Stability-DG trees to incur a significantly prolonged network lifetime, beyond the time of first node failure. For a given transmission range per node and maximum node velocity, we observe the difference between the node lifetime and network lifetime for the Max.Stability-DG trees to increase significantly with increase in the number of static nodes. This could be attributed to the reduction in the number of flooding-based tree discoveries. For a given level of node mobility, we observe the difference in the node lifetime and network lifetime for the Max.Stability-DG trees to increase with increase in the transmission range per node. This could be again attributed to the decrease in the number of network-wide flooding based tree discoveries when used



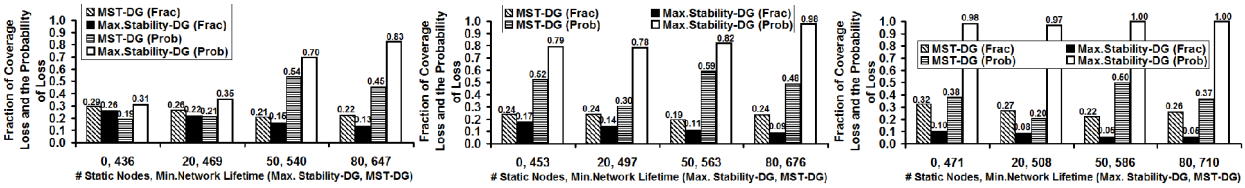
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 20: Fraction of Coverage Loss and Associated Probability (Low Node Mobility: v_{max} = 3 m/s)



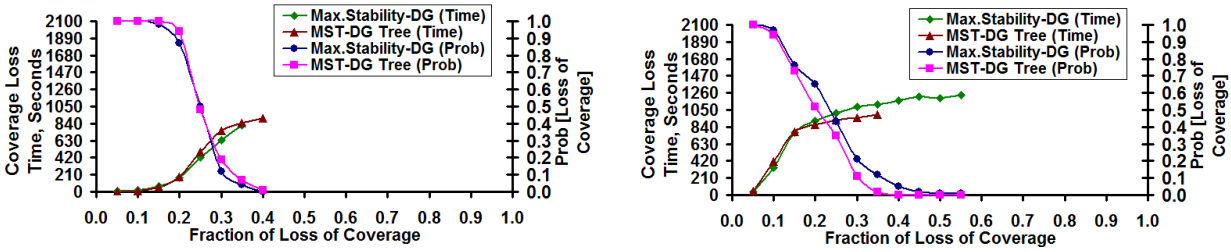
Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

Figure 21: Fraction of Coverage Loss and Associated Probability (Moderate Node Mobility: v_{max}=10 m/s)

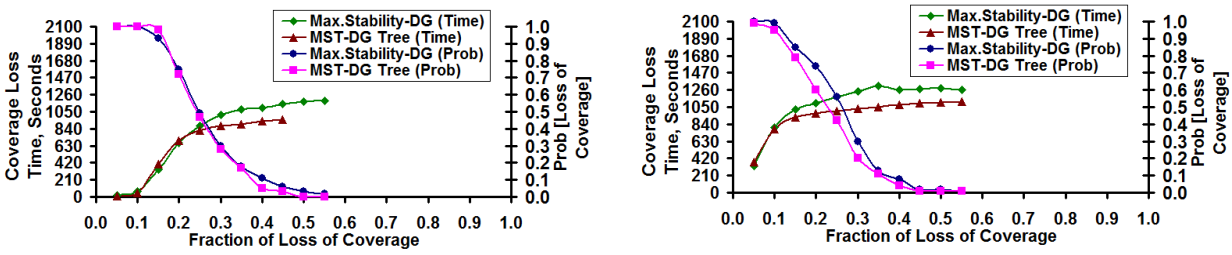


Transmission Range = 25 m Transmission Range = 30 m Transmission Range = 40 m

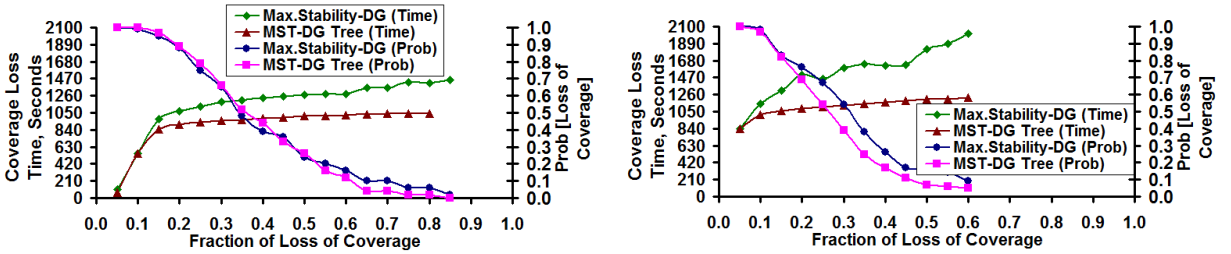
Figure 22: Fraction of Coverage Loss and Associated Probability (High Node Mobility: v_{max} = 20 m/s).



Transmission Range: 25 m, 0 static nodes Transmission Range: 25 m, 80 static nodes



Transmission Range: 30 m, 0 static nodes Transmission Range: 30 m, 80 static nodes



Transmission Range: 40 m, 0 static nodes Transmission Range: 40 m, 80 static nodes

Figure 23: Coverage Loss Time and the Probability of Coverage Loss [Low Mobility: v_{max} = 3 m/s].

at larger transmission ranges per node. Relatively, the MST-DG trees incur a very minimal increase in the network lifetime compared to the node lifetime, especially when operated at higher levels of node mobility.

One can also observe from Figures 17, 18 and 19 that the number of node failures that require for the node failure time incurred with the Max.Stability-DG trees to exceed that of the node failure time incurred with the MST-DG trees decreases with increase in maximum node mobility. This could be attributed to the premature very early node failure occurring for the Max.Stability-DG trees when operated under low node mobility scenarios, with the time of first node failure for the MST-DG tree being as large as 400% more than the time of first node failure for the Max.Stability-DG tree. On the other hand, at high levels of node mobility, the time of first node failure incurred with the MST-DG trees is at most 100% larger than that of the Max.Stability-DG trees. Hence, the node failure times incurred with the Max.Stability-DG trees could quickly exceed that of the MST-DG trees at higher levels of node mobility. At the same time, the probability for node failures to occur (that was relatively low at moderate transmission ranges per node, low and moderate levels of node mobility) with the Max.Stability-DG trees converges to that of the MST-DG trees when operated at higher levels of node mobility as well as with larger transmission ranges per node. For a given v_{max} value and transmission range per node, we also observe that the number of node failures required for the failure times incurred with the Max.Stability-DG trees to exceed that of the MST-DG trees increases with increase in the number of static nodes.

5.8 Coverage loss at a common timeline

In this section, we compare the loss of coverage incurred with both the Max.Stability-DG and MST-DG trees with respect to a *common timeline, chosen to be the minimum of the network lifetime obtained for the two data gathering trees* under every operating condition of transmission range per node, maximum node velocity and the number of static nodes. Given the nature of the results obtained for the network lifetime under different operating conditions, the minimum of the network lifetime for the two data gathering trees ended up mostly being the network lifetime observed for the MST-DG trees. For this value of network lifetime, we measured the fraction of coverage loss in the network incurred for each of the two data gathering trees, as well as measured the probability with which the corresponding fraction of coverage loss is observed.

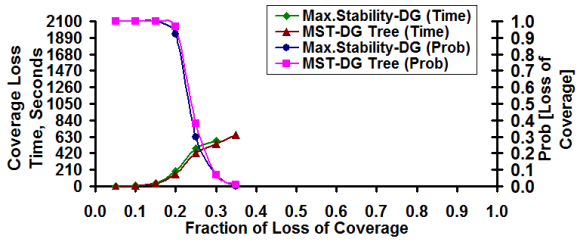
Under the above measurement model, we observe the Max.Stability-DG trees incur lower values of the fractions of coverage loss at the minimum of the network lifetime incurred for the two data gathering trees for most of all the 48 combinations of the operating conditions of maximum node velocity, number of static nodes and transmission range per node (see Figures 20, 21 and 22). However, the fraction of coverage loss observed for the Max.Stability-DG trees is bound to occur with a higher probability than that of the coverage loss to be incurred by using the MST-DG trees. The difference in the fraction of coverage loss incurred for the Max.Stability-DG trees vis-à-vis could be as large as 0.18-0.21, observed at transmission range per node of 40m and 80 static nodes, under all levels of node mobility. The only three combinations of operating conditions for which the Max.Stability-DG trees sustain a larger value for the fraction of coverage loss (that too, only by 0.02) are at a transmission range per node of 25m - v_{max} = 3 m/s, 0 and 20 static nodes; and v_{max} = 10 m/s, 0 static nodes.

In the case of the Max.Stability-DG trees, for a fixed v_{max} value, we observe the fraction of loss of coverage to decrease with increase in transmission range per node from 25m to 40m, of course with a higher probability. The significant decrease in the loss of coverage (as low as 0.05) at higher transmission range per node of 40m could also be attributed to the increase in the network lifetime, and also due to the reason that we measure the loss of coverage at a time value (corresponding to the network lifetime of the MST-DG trees), which is lower than the network lifetime of the Max.Stability-DG trees. For fixed v_{max} and transmission range, as we increase the number of static nodes, the fraction of coverage loss decreases significantly for Max.Stability-DG trees by about 0.05 to 0.1; whereas, the fraction of coverage loss for the MST-DG trees suffers a very minimal decrease or remains the same. For a fixed # static nodes and transmission range per node, node velocity has minimal impact on coverage loss for the MST-DG trees.

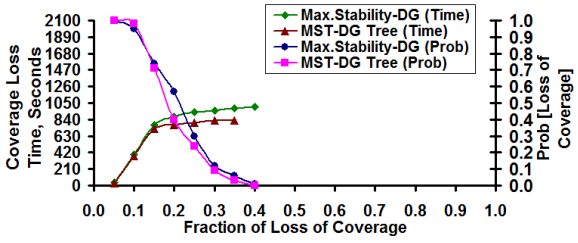
5.9 Distribution of coverage loss

In Figures 23, 24 and 25, we illustrate the distribution of the time (referred to as the coverage loss time) at which particular fractions of coverage loss occurs in the network when run with the Max.Stability-DG and MST-DG trees (until the network lifetime of the individual data gathering tree). The Max.Stability-DG trees incur larger values of coverage loss time for moderate and higher values of the fractions of coverage loss (generally above 0.15 or 0.2), under most of the combinations of the operating conditions of maximum node velocity, 0 and 80 static nodes and transmission range per node. For quantitative comparison purposes, we base our discussion in this section on the coverage loss time observed when the fraction of coverage loss is 0.3. For most of the combinations of operating conditions, we observe the coverage loss times incurred with the Max.Stability-DG and MST-DG trees to flatten out (i.e., not appreciably increase) starting from this fraction of coverage loss.

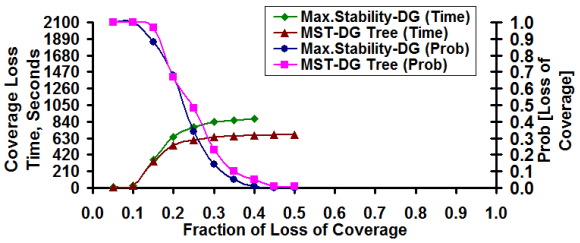
In terms of the percentage difference in the coverage loss time incurred at a fraction of coverage loss of 0.3, we observe the coverage loss time incurred with the Max.Stability-DG trees to be about 15-40%, 15-45% and 30-70% greater than the coverage loss time incurred with the MST-DG trees at low, moderate and high levels of node mobility respectively. For fixed transmission range



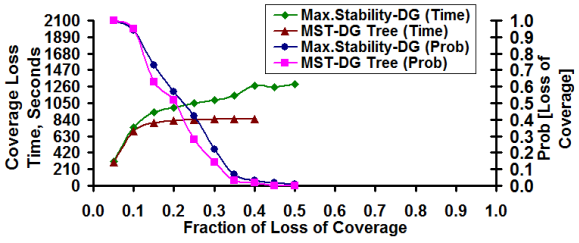
Transmission Range: 25 m, 0 static nodes



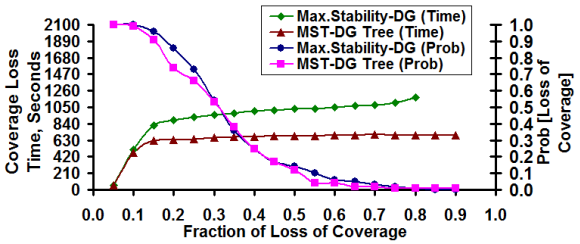
Transmission Range: 25 m, 80 static nodes



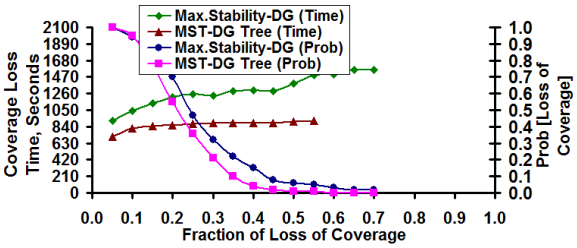
Transmission Range: 30 m, 0 static nodes



Transmission Range: 30 m, 80 static nodes

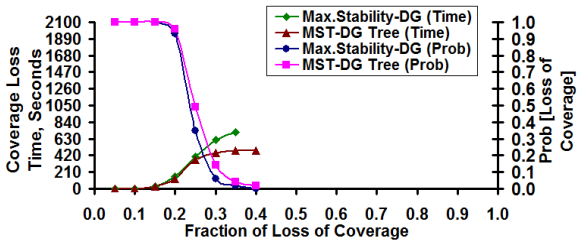


Transmission Range: 40 m, 0 static nodes

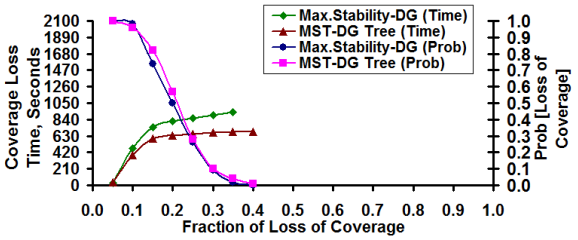


Transmission Range: 40 m, 80 static nodes

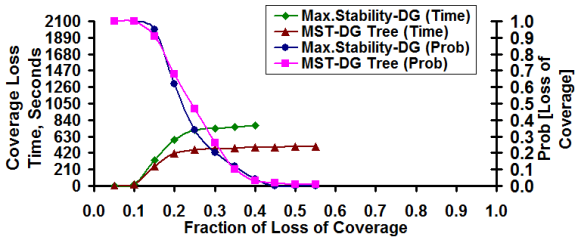
Figure 24: Coverage Loss Time and the Probability of Coverage Loss [Moderate Mobility: v_{max} = 10 m/s].



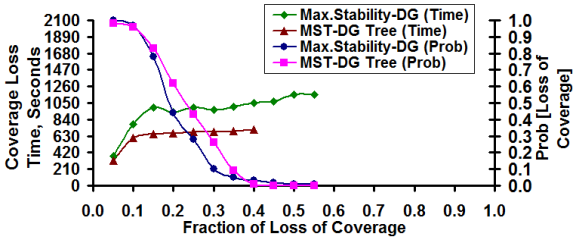
Transmission Range: 25 m, 0 static nodes



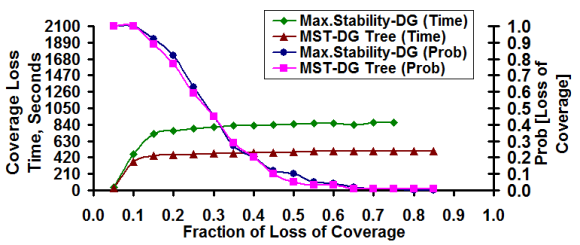
Transmission Range: 25 m, 80 static nodes



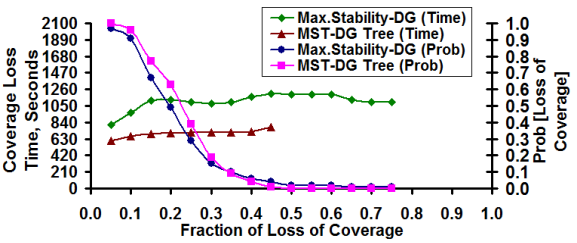
Transmission Range: 30 m, 0 static nodes



Transmission Range: 30 m, 80 static nodes



Transmission Range: 40 m, 0 static nodes



Transmission Range: 40 m, 80 static nodes

Figure 25: Coverage Loss Time and the Probability of Coverage Loss [High Mobility: v_{max} = 20 m/s].

per node and number of static nodes, the absolute magnitude for the coverage loss time incurred for both the data gathering trees decreases with increase in the v_{max} value.

For a given level of node mobility, the coverage loss time incurred with the Max.Stability-DG trees almost doubles, if not more, as we increase the transmission range per node from 25m to 40m and the number of static nodes from 0 to 80. This could be attributed to the significant energy savings obtained as a result of the need for very few network-wide flooding tree discoveries with the use of the Max.Stability-DG algorithm when operated at larger transmission ranges per node and/or more static nodes. We observe significant gains in the coverage loss time when the number of static nodes is also simultaneously increased with increase in the transmission range per node. In fact, at moderate and high levels of node mobility, the coverage loss time incurred when we run the network at transmission range per node of 25m and increase the number of static nodes from 0 to 80 is greater than or equal to the coverage loss time incurred when we run the network with 0 static nodes and increase the transmission range per node from 25m to 40m. In the case of MST-DG trees, the percentage increase in the coverage loss time with increase in the number of static nodes vis-à-vis increase in the transmission range per node is more obvious.

For both the data gathering trees, especially in the case of MST-DG trees, the potential energy savings obtained with respect to reduction in the number of network-wide flooding discoveries is much more when we operate at a moderate transmission range per node and increase the number of static nodes from 0 to 80 rather than operating at a larger transmission range per node with 0 static nodes. It is to be noted that larger the transmission range, the larger is the energy lost in transmission, and also larger is the energy lost due to receipt of the control messages from a larger number of neighbor nodes. For both the data gathering trees, we observe the increase in coverage loss time with the use of more static nodes vis-à-vis a larger transmission range per node to occur with a relatively lower probability of coverage loss.

6 Conclusions

The high-level contribution of this research is the design and development of a benchmarking algorithm (Max.Stability-DG algorithm) to obtain the upper bounds for the maximum lifetime that can be incurred with data gathering trees for mobile sensor networks. Given the entire sequence of topology changes over the duration of the data gathering session as input, the Max.Stability-DG algorithm returns the sequence of longest-living stable data gathering trees such that the number of tree discoveries is the global minimum. The run-time complexity of the algorithm has been observed to be $O(n^2T\log n)$ and $O(n^3T\log n)$ when operated under sufficient-energy and energy-constrained scenarios respectively, where n is the number of nodes in the network and T is the duration of the data gathering session. Since the Max.Stability-DG trees are spanning tree-based and a spanning tree exists in a network if and only if the network is connected, the stability of a spanning tree or any network-wide communication topology (like a connected dominating set) discovered by an existing or prospective data gathering algorithm can be evaluated by comparing its lifetime with that obtained for the Max.Stability-DG trees. With a polynomial-time complexity and a much broader scope of application, as described above, the Max.Stability-DG algorithm has all the characteristics to become a global standard for evaluating the stability of communication topologies for data gathering in mobile sensor networks.

We have shown that under sufficient-energy scenarios, the number of spanning tree discoveries incurred with the Max.Stability-DG algorithm is the theoretical minimum for any network-wide communication topology used for data gathering. In addition to the theoretical evaluation and proof of correctness, we have also shown through extensive simulations that the Max.Stability-DG trees are significantly more stable than the MST-DG trees under both sufficient-energy and energy-constrained scenarios. We evaluate the performance of the data gathering trees obtained with the Max.Stability-DG algorithm under diverse conditions of network dynamicity (varied by changing the maximum node velocity and number of static nodes) and network density (varied by changing the transmission range per node). Due to its nature to use a long-living data gathering tree as long as it exists, we observe the Max.Stability-DG algorithm to incur a lower time for the first node failure. However, the tradeoff between stability and fairness of node usage ceases to exist beyond the first few node failures; the reduced number of network-wide flooding discoveries coupled with the shallow structure and even distribution of nodes across the intermediate nodes (which also contribute to a lower delay per round) contribute to a longer lifetime for the remaining nodes in the network and significantly prolong the network lifetime as well as the coverage loss time. On the contrary, the MST-DG trees that incur a larger time for the first node failure are observed to incur a significantly lower network lifetime and lower coverage loss time for a given fraction of loss of coverage (and correspondingly incur a larger fraction of coverage loss at any time), owing to frequent network-wide flooding-based tree discoveries that expedite the node failures after the first node failure. We did not come across such a comprehensive analysis for node failure times, network lifetime, coverage loss times and fraction of coverage loss in any prior work in the literature.

Table 1 summarizes the overall performance gains obtained with the Max.Stability-DG tree vis-à-vis the MST-DG trees under both the sufficient-energy and energy-constrained scenarios, as applicable. Table 2 ranks the three operating parameters in the decreasing order of influence on the performance of the two data gathering trees. As can be seen from Table 2, the nature of influence of the operating parameters on the performance of the two data gathering trees is more or less the same.

Performance Metric	Better Data Gathering Tree	Range of Performance Gain Compared to the other Data Gathering Tree	
		Sufficient-Energy Scenario	***Energy-Constrained Scenario***
Tree Lifetime	Max.Stability-DG tree	150% to 360% larger	40% to 200% larger
Delay per Round	Max.Stability-DG tree	4% to 25% lower	7% to 18% lower
Energy Lost per Node	Max.Stability-DG tree	7% to 45% lower	Not applicable
Energy Lost per Round	Max.Stability-DG tree	7% to 45% lower	Not applicable
Fairness of Node Usage	MST-DG tree	10% to 50% better	Not applicable
Node Lifetime	MST-DG tree	Not applicable	10% to 420% larger
Network Lifetime	Max.Stability-DG tree	Not applicable	5% to 60% larger
Coverage Loss Time	Max.Stability-DG tree	Not applicable	15% to 70% larger
Fraction of Coverage Loss	Max.Stability-DG tree	Not applicable	2% to 25% lower

Table 1: Overall Performance Gains for the Maximum Stability Spanning Tree based Data Gathering (Max.Stability-DG) Tree and Minimum-distance Spanning Tree based Data Gathering (MST-DG) Tree.

Performance Metric	Ranking of the Operating Parameters in the Order of Influence [*1-Highest Influence*]					
	Max. Stability-based Data Gathering			***Min. distance-based Data Gathering***		
	Node Velocity	*Static Nodes*	*Transmission Range/ Node*	*Node Velocity*	*Static Nodes*	*Transmission Range/ Node*
Tree Lifetime	1	3	2	1	3	2
Delay per Round	3	2	1	3	2	1
Energy Lost / Node	1	2	3	1	2	3
Energy Lost / Round	1	2	3	1	2	3
Fairness of Node Use	1	3	2	1	3	2
Node Lifetime	1	3	2	1	3	2
Network Lifetime	1	2	3	1	2	3
Coverage Loss Time	1	2	3	1	2	3
Frac. Coverage Loss	3	2	1	2	3	1

Table 2: Influence of the Operating Parameters on the Performance of the Data Gathering Trees.

7 Future work

As part of future work, we plan to do the following:

(i) We will develop a distributed stability-based data gathering algorithm for mobile sensor networks based on the notion of the predicted link expiration time (LET) [27] – a link model successfully proposed for stable path routing in mobile ad hoc networks. From the lessons learnt in this research, we conjecture that a simple LET-based data gathering algorithm would discover stable trees at the cost of premature node failures (at least the first few node failures would occur much earlier, like we observed for the Max.Stability-DG trees). Hence, we plan to model the edge weight as a function of both the LET and the residual energy (available energy) of the end nodes of the link, so that we can balance the tradeoff between stability and unfairness in node usage. We can then compare the stability-node lifetime tradeoff of the LET-energy-DG trees with that of the Max.Stability-DG trees.

(ii) As stable data gathering trees are likely to be used for a longer time, the trustworthiness of the data aggregated at the intermediate nodes needs to be validated and maintained through proper trust-evaluation schemes. We plan to develop and integrate a trust-evaluation model as part of stable data aggregation in mobile sensor networks.

(iii) We plan to compare the stability of the Max.Stability-DG trees under several different node mobility models [4] vis-à-vis the Random waypoint model, the mobility model used in our simulations that has been widely used in the ad hoc network literature.

Acknowledgment

This research was sponsored by the U. S. Air Force Office of Scientific Research (AFOSR) through the Summer Faculty Fellowship Program for the lead author (Natarajan Meghanathan) in June-July 2012. The research was conducted under the supervision of the co-author (Philip D. Mumford) at the U. S. Air Force Research Lab (AFRL), Wright-Patterson Air Force Base

(WPAFB) Dayton, OH. The public release approval number is 88ABw-2012-4935. The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[1] M. Abolhasan, T. Wysocki and E. Dutkiewicz, "A Review of Routing Protocols for Mobile Ad hoc Networks," *Ad hoc Networks*, vol. 2, no. 1, pp. 1-22, January 2004.

[2] T. Banerjee, B. Xie, J. H. Jun, and D. P. Agarwal, "LIMOC: Enhancing the Lifetime of a Sensor Network with Mobile Clusterheads," *Proceedings of the Vehicular Technology Conference Fall*, pp. 133-137, 2007.

[3] C. Bettstetter, H. Hartenstein and X. Perez-Costa, "Stochastic Properties of the Random-Way Point Mobility Model," *Wireless Networks*, vol. 10, no. 5, pp. 555 – 567, September 2004.

[4] T. Camp, J. Boleng and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communication and Mobile Computing*, vol. 2, no. 5, pp. 483-502, September 2002.

[5] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein, "Introduction to Algorithms," 3rd Edition, MIT Press, July 2009.

[6] S. Deng, J. Li and L. Shen, "Mobility-based Clustering Protocol for Wireless Sensor Networks with Mobile Nodes," *IET Wireless Sensor Systems*, vol. 1, no. 1, pp. 39-47, 2011.

[7] A. Farago and V. R. Syrotiuk, "MERIT: A Scalable Approach for Protocol Assessment," *Mobile Networks and Applications*, Vol. 8, No. 5, pp. 567 – 577, October 2003.

[8] W. Heinzelman, A. Chandrakasan and H. Balakarishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks," *Proceedings of the Hawaaian International Conference on Systems Science*, January 2000.

[9] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan and S. Madden, "CarTel: A Distributed Mobile Sensor Computing System," *Proceedings of the 4th International Conference on Embedded Networked Sensor Systems*, November 2006.

[10] A. Kansal, M. Goraczko and F. Zhao, "Building a Sensor Network of Mobile Phones," *Proceedings of International Symposium on Information Processing in Sensor Networks*, pp. 547-548, April 2007.

[11] F. Kuhn, T. Moscibroda and R. Wattenhofer, "Unit Disk Graph Approximation," *Proceedings of the Workshop on Foundations of Mobile Computing*, pp. 17-23, October 2004.

[12] S. Lindsey, C. Raghavendra and K. M. Sivalingam, "Data Gathering Algorithms in Sensor Networks using Energy Metrics," *IEEE Transactions on Parallel and Distributed Systems*, 13(9):924-935, September 2002.

[13] C-M. Liu, C-H. Lee and L-C. Wang, "Distributed Clustering Algorithms for Data Gathering in Wireless Mobile Sensor Networks," *Journal of Parallel and Distributed Computing*, vol. 67, no. 11, pp. 1187-1200, November 2007.

[14] G. Lu, B. Krishnamachari and C. S. Raghavendra, "An Adaptive Energy-Efficient and Low-Latency MAC for Tree-based Data Gathering in Sensor Networks," *Wireless Communications and Mobile Computing*, vol. 7, no. 7, pp. 863-875, September 2007.

[15] M. Macuha, M. Tariq and T. Sato, "Data Collection Method for Mobile Sensor Networks based on the Theory of Thermal Fields," *Sensors*, vol. 11, no. 7, pp. 7188-7203, July 2011.

[16] N. Meghanathan, "A Comprehensive Review and Performance Analysis of Data Gathering Algorithms for Wireless Sensor Networks," *International Journal of Interdisciplinary Telecommunications and Networking* (IJITN), vol. 4, no. 2, pp. 1-29, April-June 2012.

[17] N. Meghanathan, "A Data Gathering Algorithm based on Energy-aware Connected Dominating Sets to Minimize Energy Consumption and Maximize Node Lifetime in Wireless Sensor Networks," *International Journal of Interdisciplinary Telecommunications and Networking*, vol. 2, no. 3, pp. 1-17, July-September 2010.

[18] N. Meghanathan and G. W. Skelton, "A Two Layer Architecture of Mobile Sinks and Static Sensors," *Proceedings of the 15th International Conference on Advanced Computing and Communication*, pp. 249-254, Guwahati, India, December 2007.

[19] N. Meghanathan, S. Sharma and G. W. Skelton, "On Energy Efficient Dissemination in Wireless sensor Networks using Mobile Sinks," *Journal of Theoretical and Applied Information Technology*, vol. 19, no. 2, pp. 79-91, September 2010.

[20] N. Meghanathan, "Performance Comparison of Minimum Hop and Minimum Edge Based Multicast Routing Under Different Mobility Models for Mobile Ad Hoc Networks," *International Journal of Wireless and Mobile Networks*, vol. 3, no. 3, pp. 1-14, June 2011.

[21] N. Meghanathan and A. Farago, "On the Stability of Paths, Steiner Trees and Connected Dominating Sets in Mobile Ad Hoc Networks," *Ad Hoc Networks*, vol. 6, no. 5, pp. 744 - 769, July 2008.

[22] T. S. Rappaport, "Wireless Communications: Principles and Practice," 2nd edition, Prentice Hall, January 2002.

[23] G. Santhosh Kumar, M. V. Vinu Paul and K. Jacob Poulose, "Mobility Metric based LEACH-Mobile Protocol," *Proceedings of the 16th International Confernece on Advanced Computing and Communications*, pp. 248-253, December 2008.

[24] H. K. D. Sarma, A. Kar and R. Mall, "Energy Efficient and Reliable Routing for Mobile Wireless Sensor Networks," *Proceedings of the 6th IEEE International Conference on Distributed Computing in Sensor Systems Workshops*, June 2010.

[25] M. Singh, M. Sethi, N. Lal and S. Poonia, "A Tree Based Routing Protocol for Mobile Sensor Networks (MSNs)," *International Journal on Computer Science and Engineering*, vol. 2, no. 1S, pp. 55-60, 2010.

[26] A. Srinivasan and J. Wu, "TRACK: A Novel Connected Dominating Set based Sink Mobility Model for WSNs," *Proceedings of the 17th International Conference on Computer Communications and Networks*, August 2008.

[27] W. Su and M. Gerla, "IPv6 Flow Handoff in Ad hoc Wireless Networks using Mobility Prediction," *Proceedings of the IEEE Global Telecommunications Conference*, pp. 271-275, December 1999.

[28] A. J. Viterbi, "CDMA: Principles of Spread Spectrum Communication," 1st edition, Prentice Hall, April 1995.

[29] N. Vlajic and D. Stevanovic, "Sink Mobility in Wireless Sensor Networks: When Theory meets Reality," *Proceedings of the IEEE Sarnoff Symposium*, March-April 2009.

[30] W. Wu, H. Beng Lim and K-L. Tan, "Query-driven Data Collection and Data Forwarding in Intermittently Connected Mobile Sensor Networks," *Proceedings of the 7th International Workshop on Data Management for Sensor Networks*, pp. 20-25, 2010.

[31] G. Xing, T. Wang, W. Jia and M. Li, "Rendezvous Design Algorithms for Wireless Sensor Networks with a Mobile Base Station," *Proceedings of the 9th ACM International Symposium on Mobile Ad hoc Networking and Computing*, pp. 231-240, 2008.

[32] H. Zhang and J. C. Hou, "Maintaining Sensing Coverage and Connectivity in Large Sensor Networks," *Wireless Ad hoc and Sensor Networks*: *An International Journal*, vol. 1, no. 1-2, pp. 89-123, January 2005.

[33] [33] W. Zhang and G. Cao, "DCTC: Dynamic Convoy Tree-based Collaboration for Target Tracking in Sensor Networks," *IEEE Transactions on Wireless Communications*, vol. 3, no. 5, pp. 1689-1701, September 2004.

[34] M. Zhao and Y. Yang, "Bounded Relay Hop Mobile Data Gathering in Wireless Sensor Networks," *Proceedings of the 6th IEEE International Conference on Mobile Ad hoc and Sensor Systems*, pp. 373-382, October 2009.

[35] M. Zhong and C. G. Cassandras, "Distributed Coverage Control and Data Collection with Mobile Sensor Networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2445-2455, October 2011.

Decision-Making in Determining the Level of Knowledge of Students in the Learning Process under Uncertainty

Shahnaz N. Shahbazova
Department of Information Technology and Programming
Azerbaijan Technical University
25 H.Javid Ave., Baku, AZ1073 Azerbaijan
E-mail: shahbazova@gmail.com

Keywords: intellectual learning and testing system, algorithm of decision-making, fuzzy logic, knowledge base, control the student's knowledge

Received: August 22, 2012

In this paper, intellectual technology is applied with the device of fuzzy logic as the basis of a developed system of learning and testing. This system is used to automate the process of teaching and control of knowledge of students in higher education institutions. The quality of the intellectual learn learning system directly depends on the accuracy of the definition of the student's current level of knowledge, as well as the choice of a learning strategy. It can be a transition to teaching new material, review of old material, or the learning completion.

Povzetek: Predstavljena je metoda za določanje znanj študentov v verjetnostih domenah .

1 Introduction

Application of modern intellectual information systems creates the new environment into which modern methods of teaching are easily integrated, as well as flexibility and individuality that were unachievable in traditional methods of learning.

An advantage to nontraditional ways of teaching is that the application of network technologies can simultaneously train a considerable quantity of students while maintaining an individualized educational process and testing. It allows superior mastery of course material in comparison with a traditional method of learning because it can determine the real level of knowledge of teaching materials of each given student.

Another advantage is the flexibility of the system. The student can choose from a collection of materials reflecting the experience and talent of the best teachers from every corner of the world. This system allows the student to choose from various learning courses the method and intensity of learning according to the student's own preferences and abilities.

Each modern higher education institution must create and integrate educational information systems into complex intellectual, information learning system network; it must develop universal automated methods of information control as well as an individualized approach to student learning [12].

As a practical realization of the intellectual information system, we solve the following problems:

- The problem of creating a unified system of storage methods of distributed informational materials in networks of complex configurations.
- The problem of creating a flexible system of learning and testing with universal quality monitoring of knowledge of students.

For each of these tasks, we propose solutions that have proven themselves in practice. The proposed model of structuring informational materials combines flexibility and efficiency and can be applied to any system of governance. Another problem solved in this article is the use of the algorithmic modeling decision-making system to control the student's knowledge; it is an original mechanism for simulating the techniques of the teacher in an individual test of the student. Analytical study of the teacher's teaching techniques makes it possible to find several main principles; these are used to increase the effectiveness of control procedures in the automated system of knowledge. It is used to develop a more detailed analysis of each incorrect answer given in a test to define not only the current assessment that the student has earned, but also to estimate the total knowledge the student possesses. This algorithm is an essential element developed for an automated system with minimal learning from an instructor.

2 Intellectual information system of learning and testing (IISLT)

The structure of an intellectual information system of learning and testing (IISLT) is represented in figure 1[5]. Let us consider the structure of IISLT in more detail. The following database groups are stored: informational

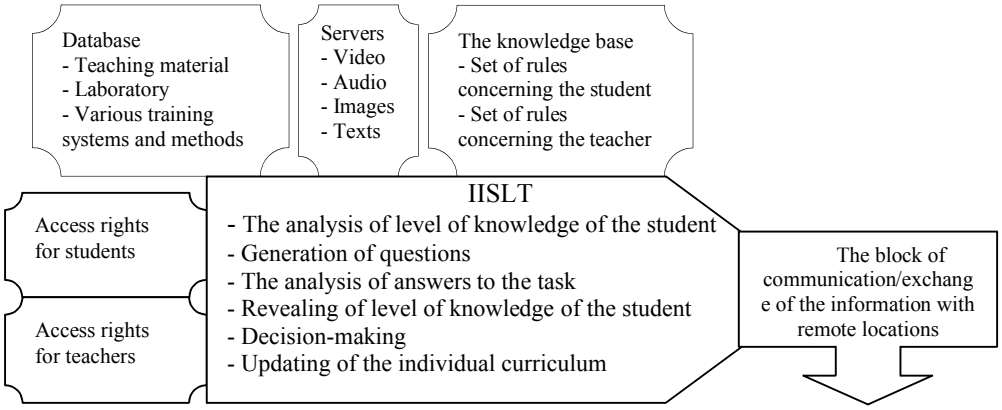


Figure 1: Structure of IISLT.

material, programs, teaching material, and details of the students.

Informational material can be presented in the form of electronic resources, for example, video and audio collections, laboratories of text materials, and unabridged editions of various dictionaries and encyclopedias. This large quantity of information is on different system servers and there is the possibility of the addition of new systems or edits of existing material.

Multimedia courses and search programs require multiple system resources and plays of files of various formats (video, audio, and images). The data can also be used for test and training programs.

The teaching material can be divided into three groups according to functional value:

- Preparatory material – contains a considerable quantity of varied teaching material on different themes of all areas of knowledge.
- Pedagogical materials – consist of courses of lectures, abstracts, and various questionnaires prepared by teachers for the practice. This material can be more purposeful and structured, reflecting different methods of teaching.
- Training material – is intended for intermediate or definitive testing of knowledge of the student. Various testing methods, for the purpose of definition of level of knowledge and abilities of the student, are in this category.

Information system resources are increased through cross-referencing; in this way, the new material is easily integrated into the information environment.

This method uses accompanying keywords as index elements. Keywords enable easy guidance in extensive information resources of a similar orientation and subjects, without limiting teachers to additional conventions required in registering learning materials.

Resources of the intellectual system are increased by addition or improvement of algorithms of decision-making and also expansion of the library of rules of the knowledge base realized on a mathematical apparatus of the fuzzy logic and neural networks.

One of the main system resources is the information folder of the student, which stores all data on the student's abilities, the schedule for training and testing, the statistic, and so on. In addition to this data, comments and remarks of teachers are stored in this folder.

3 The algorithm of decision-making

This mechanism of decision-making features the choice of level of question complexity set for the student on the basis of the result of the answer to the previous question [6,8].

The resolution of this problem depends on a considerable quantity of parameters, the majority of which are not known to the intellectual system (owing to complexity); however, a fairly accurate answer can be found by means of the mathematical apparatus of fuzzy logic [1,2].

The algorithm of decision-making is based on the results of the decisions to the following problems:

1. The preliminary analysis of knowledge of the student – is used for an estimation of level of student knowledge for decision-making of a choice of the first question (to the unprepared student a simple question is asked whereas the prepared student receives a more difficult question)[7];

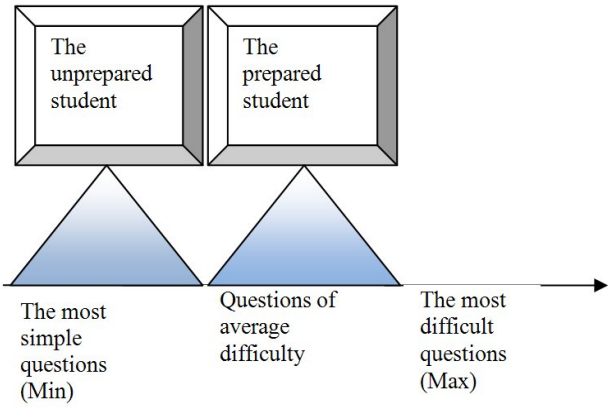


Figure 2: Strategy of a choice of the first question.

2. The student has answered the previous question correctly – in this case, the student is asked a question of increased difficulty

$$Q = (\text{Max } (A+) + (\text{Max or Max } (A-))) / 2 \pm 2\ \% \qquad (1)$$

where Q = the following question, (A +) = a right answer, (A-) = the wrong answer, and ±2 % are the limits for the following question.

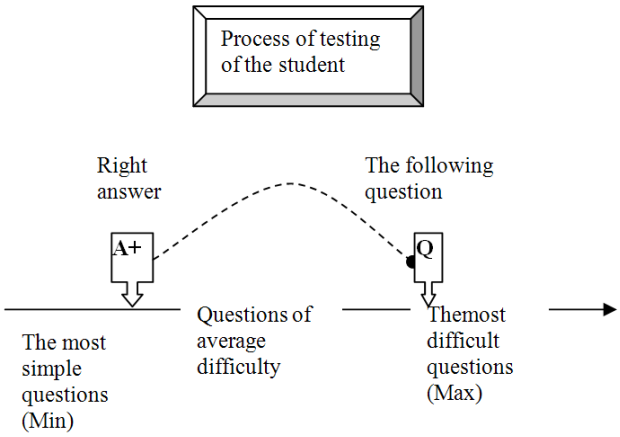


Figure 3: Strategy of a choice of a question resulting from a right answer.

3. The student has answered the previous question incorrectly – in this case, the student is asked a question of lowered difficulty.

$$Q = (\text{Max } (A-) + (\text{Min or Min } (A+))) / 2 \pm 2\ \%; \quad (2)$$

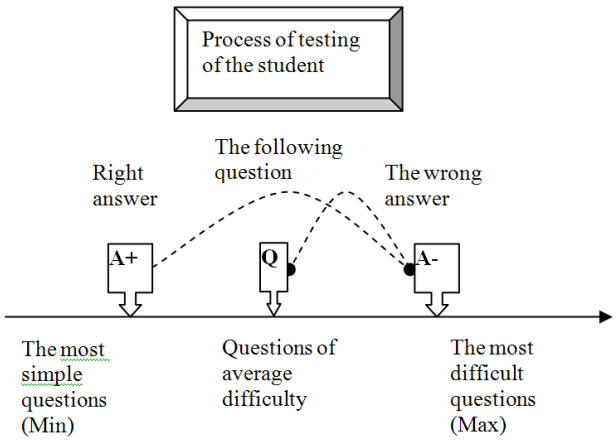


Figure 4: Strategy of a choice of a question resulting from a wrong answer.

4. Processing of results and decision-making on a definitive estimation or testing continuation – quantity of right answers multiplied by their complexity in relation to errors; the set of correct and wrong answers are input to the decision-making subroutine for estimation results. If there is a high probability of uncertainty, testing proceeds [11]

$$\{Z, P\} = f\,(\sum (A+) \times (V+), \sum (A-) \times (V-), (A0+), (A1+)..., (A0-), (A1-)...) \quad (3)$$

where Z = an estimation, P = uncertainty, f = the subroutine of decision-making,

(Ai +) = set of right answers, (Ai-) = set of wrong answers, (V +) = weight of right answers, and (V-) = weight of wrong answers.

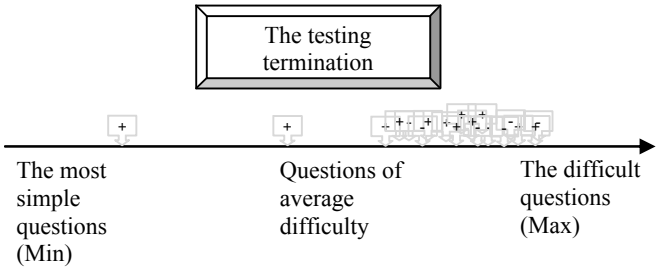


Figure 5: An example of distribution of answers upon termination of testing.

One of the functions of the decision-making block is a choice of the following question which, most likely, corresponds to the level of knowledge of the student [9]. Upon an incorrect answer, the data about the student is reevaluated and a less complicated question is chosen (2). Upon a right answer, the program chooses a more complicated question (1, 13).

As was previously mentioned, because of the considerable quantity of external parameters, these decisions are analyzed and executed by means of a mathematical apparatus of fuzzy logic. The question choice is carried out by performance of several calculations of a set of the fuzzy expressions [3,4]; the ultimate goal is transfer of the results to a decision-making subsystem.

This decision-making process enables the sequence of question choices to be individualized and at test termination to yield the most exact estimation of knowledge of the student [10].

Based on the previous results of the student (3) in the given subject, the system analyzes the test results and takes out an estimation which is brought in the private affair of the student. This estimate of overall test performance is presented in digital form (Figure 6).

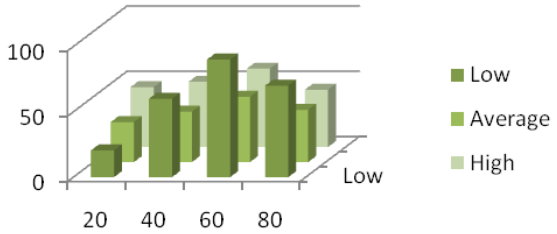


Figure 6: Graph of the generalized result of testing.

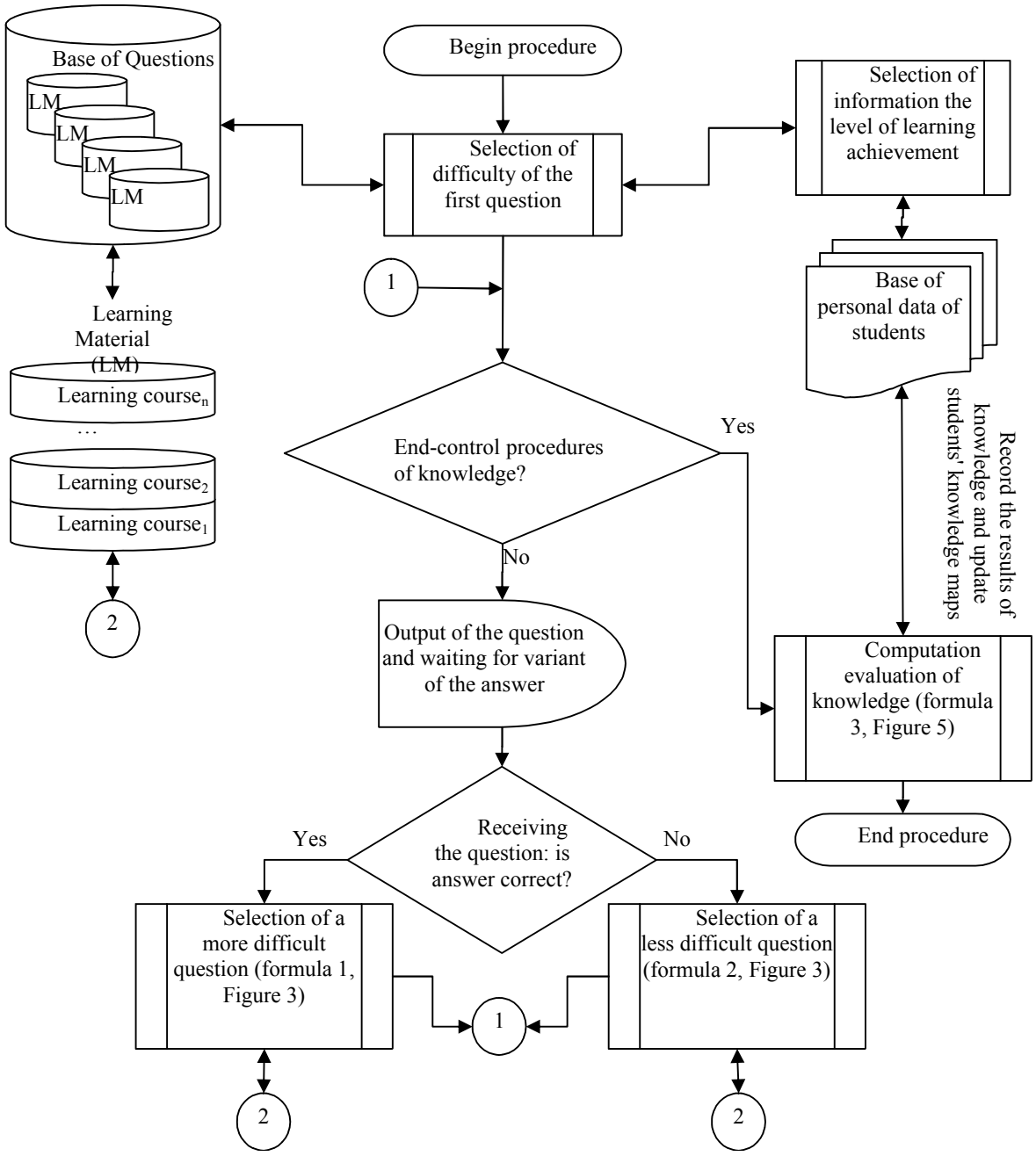


Figure 7: Block diagram of subprogram of control of knowledge.

The mechanism of selection of questions described above during the testing procedure can be represented as an algorithm displayed in the block diagram shown in Figure 7.

The implementation of this algorithm is extremely simple and is based on applying a set of rules of fuzzy logic, which are logical structures that control the process of control of knowledge.

$$\begin{aligned} &IF\ f_{input}(X_1)THEN\ f_{output}(Y_1)\\ &IF\ f_{input}(X_2)THEN\ f_{output}(Y_2)\\ &\ldots. \end{aligned} \tag{4}$$

where the incoming data according to X characterizing the coming of the student survey results, select the appropriate response of the current situation of systems which Y represents one of the four analyzed above, the answers of module knowledge control [14].

The decision of an appropriate question for the estimation for the executed test can be based on both the program and the teacher or their joint decision. The given algorithm of decision-making enables training or testing at any level of complexity, from the individual test to the graduation examination.

In the process, the students can not only determine their level of knowledge but also learn to analyze their own mistakes, which can affect the curriculum. By achieving sustainable results in the test on a particular subject, the student can go to the next section of the course.

Participating in the training will give the necessary time to complete assimilation and retention of the

material and then transition to new, more complex material. Each transition is accompanied by a small test evaluating retention of the content of the previous course.

4 Conclusion

This article discusses the mechanism for an intelligent method for the control of students' knowledge. This paper describes the intellectual system in which the advantages of the test system and the algorithm of performance of the teacher are combined to determine the knowledge of the trainee.

This developed algorithm of decision-making can test the knowledge level of the student using a minimal number of questions with a high level of reliability in comparison to a traditional testing method, the spent teacher or at the sample test task.

References

[1] M.Nikravesh, F.Aminzadeh, L.A.Zadeh, Soft Computing and Intelligent data analysis in oil exploration, 2003, Elsevier.

[2] M.Nikravesh, L.A.Zadeh, J.Kacprzyk, Soft Computing for Information Processing and Analysis, 2005, Springer.

[3] A.B.Barsky, Neural networks: recognition, management, decision-making, 2004, Moscow, «Finance and statistics».

[4] N.G.Jarushkina, Bases of the theory of fuzzy and hybrid systems, 2004, Moscow, «Finance and statistics».

[5] Shahbazova Sh., Freisleben B., A Network-Based Intellectual Information System for Learning and Testing, Fourth International Conference on Application of Fuzzy Systems and Soft Computing, Siegen, Germany, 2000, pp.308-313.

[6] Shahbazova Sh., Zeynalova S., Decision-Making In Definition Of Knowledge In The Conditions Of Uncertainty Of Educational Process, PCI-2010, Volume I, "Elm", 2010, pp.305-310.

[7] Zadeh L.A., Klir G.J., Yuan B., Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers by Lotfi A. Zadeh, 1996, pp.60-69.

[8] Zadeh L.A., The new approach to the analysis of difficulty systems and decision processes, Mathematics Today, Knowledge, 1974, pp. 23-37

[9] Zadeh L.A., A new approach to the analysis of difficulty systems and decision processes, Mathematics Today, Knowledge, 1974, pp. 23-37

[10] Bernshteyn L.S., Bojenyuk A.V., Fuzzy models of decision making: deduction, induction, analogy. Taganrog, Univ Tsure, 2001. pp. 78-99.

[11] Bouchon-Meunier B., Yager R.R., Fuzzy Logic and Soft Computing (Advances in Fuzzy Systems: Application and Theory), World Scientific, 1995, pp.84-93, 103-119.

[12] Gorbunova L.G., On the realization of the rating system in pedagogical high schools, Proceedings of 2 International Technical Conference "University Education", Penza, 1998, Part 1. – pp.105-106.

[13] Shahnaz N.Shahbazova, Application of Fuzzy Sets for Control of Student Knowledge, ISSN 1683-3511, Applied and Computational Mathematics, An International Journal, Volume 10, N 1, 2011. Special Issue on Fuzzy Set Theory and Applications, pp.195-208.

[14] Shahnaz N.Shahbazova, Application of Fuzzy Sets for Control of Student Knowledge, ISSN 1683-3511, Applied and Computational Mathematics, An International Journal, Volume 10, N 1, 2011, Special Issue on Fuzzy Set Theory and Applications, pp.195-208.

A Load Balancing Strategy for Replica Consistency Maintenance in Data Grid Systems

Senhadji Sarra, Kateb Amar and Belbachir Hafida
LSSD Laboratory, University of science and Technology, Oran, Algeria
E-mail: senhadji.sarah@gmail.com, kateb_amar@yahoo.fr and h_belbach@yahoo.fr

Keywords: consistency, replication, data grid, load balancing

Received: January 28, 2013

Abstract: In data grid environment, the management of shared data is one of the major scientific challenges. Data replication is one of the important techniques used in grid systems to increase the availability, scalability and fault tolerance. However, the update of a replica might bring a critical problem of replica consistency maintenance. Thus, maintaining the consistency of the replicas is not trivial because of the instability of the grid system where the nodes can join and leave at any time. In addition, according to Read/Write access frequency some nodes can be more uploaded than others and become a bottleneck. In order to handle these problems, we propose a model of consistency based on quorum protocol. The replica consistency performances are improved by using a dynamic load balancing strategy in a simulated grid environment.

Povzetek: Članek govori o ohranjanju konsistentnosti kopij podatkov v omrežnem računalništvu in predlaga nov model, ki temelji na protokolu sklepčnosti.

1 Introduction

Grids [7, 25] are a wide area computing system geographically dispersed that involves high computational and storage resources. The grid is considered as one of the promiscuous technologies for scientific applications like astronomy, bioinformatics and earth sciences. In this kind of dynamic and large scale environment, the management of massive data is still being one of the important open problems [14, 6]. Different techniques dedicated for intensive data management are used in several domains. Data replication is one of the most important techniques having attracted the researcher community attention.

The replication technique improves data availability, scalability and fault tolerance. The main research areas dealing with data replications are the replica placement and the replica consistency maintenance.

The replicas placement mechanism determines which data should be replicated? When to create new copy? And where the new replicas should be placed? For replica placement, many works and solutions have been proposed in the literature [17, 18, 22, 23]. The main objective of these solutions is to store copies (or replicas) of a data in different sites, so that data can be easily restored if one copy is lost. Also, by placing replicas closer to grid users, the access performances are significantly improved in term of response time, consumed bandwidth, etc.

However, the update of replica by any grid user might bring a critical problem of maintaining consistency among the other replicas of the grid. In fact, when a replica is modified, a copy must be propagated to the rest of replicas in order to get identical and consistent copies.

Moreover, according to the frequency access to the different grid nodes, some nodes are more uploaded then others. This irregular evolution engenders an unbalance and many resources can be unexploited. With an efficient load balancing strategy, the system can reduce the query response time and avoid failures due to overloaded nodes. That is why we propose a complementary solution between replica consistency and load balancing.

For this, we propose a consistency protocol based on quorum system [5]. The main idea of quorum systems is to involve a large collection of possible sets of nodes in the replica consistency management. Nodes holding replicas of the same data are represented logically into a tree structure, called Coterie. When a Read/ Write request is addressed to a grid node, a path from the root to the leaf tree, called quorum, is selected to achieve the replica consistency. Complementing to replica consistency, we define a load balancing strategy, based on elementary permutation between coterie's nodes in order to reduce the load and communication time of R/W request.

Thus, the main idea of our contribution consists of ameliorating the replica consistency performances through a dynamic load balancing strategy in term of consistency, load balancing and communication cost.

In the next section we give an overview of some existing works pertaining to replica consistency. Then we define our approach with the adopted dynamic load balancing strategy. The evaluation of our approach will be discussed in experiments section. Finally, we close this paper with some conclusions.

2 Related works

Various works have been done on the replica consistency domain in distributed systems, such as cluster, peer to peer and grid. Many consistency models exist in the literature [1] as Strong models and Weak models [9, 24]. Strong consistency models keep data consistent among all replicas simultaneously, which requires more resources and expensive protocols than other models. In weak consistency models the strong consistency protocols are relaxed in order to tolerate inconsistencies among replicas for a while to improve access performances. In consequence, replicas returned to a read request are not perfectly the latest updated value. For this reason, replica divergence must be controlled since maintaining replica freshness becomes more complex as divergence increases.

In [8] the authors address the problem of shared data in data grid systems. The consistency of replicated data is introduced by relaxed read which is an extension of the entry Consistency model [15]. Unlike the model of entry consistency, which ensures that data is current as at the acquisition of its lock, this new type of operation can be achieved without locking, in parallel with write operations. However, data freshness constraint is released and older versions, which however still be controlled, are accepted. The grid architecture considered in this work is composed of clients requesting the data, the data providers and two hierarchical levels: LDG (Local Data Grid) and GDG (Global Data Grid).Two types of copies are considered: local copy, hosted by the LDG and global copy, hosted by the GDG. When a client accesses the data, a request to acquire the synchronization object is addressed to the node hosting the local copy. If the node owns the synchronization object, the client is served. Otherwise, an acquisition request is sent to the node hosting the global copy.

To handle the problem of storage data in grids, the consistency model proposed in [4] improved the storage space and access time of replicated data. The authors of this work suggest a topology built hierarchically upon three types of nodes: Super Node SN, Master Node MN, and Child Node CN. The source of the replicated data is kept in the SN; this data can then be modified by users of the grid, called "original data". When the original data is added or modified, then it is automatically replicated to the Master Nodes MN. The replica of the Master Node MN is called Master Replica. At the node CN, the data is replicated from the Master Node MN according to two main factors: the file access frequency and the storage space capacity. The replicated data is called Child Replica. Replicas located at MN and CN are read only.

Another similar work to [4] was proposed by [3] by considering the bandwidth consumed until the Read/ Write operations. Most of existing replication works [2, 10] in data grid systems focuses on consistency management without taking care of the load imbalance of the grid nodes which can low significantly the replication performances.

Some of load balancing solutions have been proposed in the literature [13, 20].For example in [12] Quorum systems are used under a simulated environment [5]. A coterie represents a set of copies of the replicated data. A Quorum is defined as the minimum set of nodes owning a replica. Quorum protocols are characterized by two main properties which are properties of intersection and minimality [11]. Considering two quorums Q and Q' of a coterie C, the property $Q \cap Q' \neq \emptyset$ is called intersection property and the property $Q \not\subseteq Q'$ is called the minimality property. The authors of [12] treated the load balancing problem, by providing a coterie reconfiguration method, to improve the read / write accesses. The load of a quorum Q is the maximum load of the nodes of this Quorum and the load of a coterie is equivalent to the sum of loads of its quorums. The nodes are tree structured and a Quorum is obtained by taking all nodes of any path from the root to a leaf of the tree. Every read (or write) operation is performed on a Quorum of the coterie.

An elementary permutation of the coterie is performed to obtain a new less loaded coterie. For this, two parent's nodes are selected to be swapped in the tree (father and its son) when the son's load is less than the father's load. This has the effect of positioning the node with the lightest load above the busiest node. An extension of the atomic read / write service [16] is proposed, with multiple readers and multiple writers. Two phases are proposed: query and a propagation phase. During the request phase, a read-quorum is contacted and each node returns the recent version which is consequently propagated to all the nodes of the quorum. This has the advantage that obsolete copies are updated even during read operations.

As few works attempted to resolve the load balancing problem with replica consistency, our contribution is to propose a dynamic load balancing strategy to increase replica consistency performances in terms of load and communication cost.

3 Proposed approach

In the literature, few studies addressed the problem of load balancing to increase the replica access performances. In [12] load balancing is adopted by introducing dynamic node permutations, but regardless of the problem of communication cost generated during Read /Write access. In our work, nodes hosting replicas of the same data are represented into a binary tree, called Coterie. Indeed, in a R /W access, a Quorum designed from the root to a leaf node is selected to be red or written. An intermediate node can participate in different possible Quorums. These intermediate nodes can represent a critical point where the cost of communication may be degraded during exchange accesses. To handle this problem, when a Read/ write request is addressed, our load balancing strategy is invoked and the Coterie is restructured in order to reduce the load of the coterie and the communication cost of the quorum when the Read/Write request is achieved.

3.1 Grid and replicas model

The nodes of the grid are represented into coteries. A coterie that owns all replicas of the same data is structured in a binary tree. In order to improve data availability, for each coterie node we define n versions. Each version is characterized by three parameters < N, S, V>, representing respectively, the node that creates or modifies this version, the stamp which represents the moment of the creation or the update version of the replica and finally the value of the replica. An example is shown in figure 1.

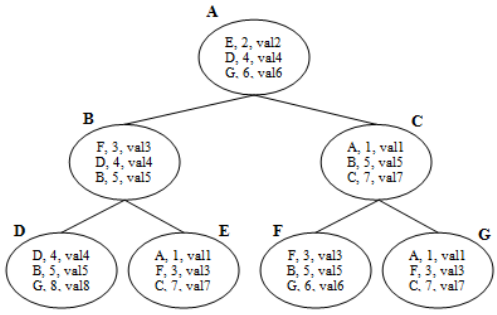


Figure 1: Example of a coterie with versions.

In figure 1, three versions are defined for each node of the coterie. For example, the node A owns the following versions **<E, 2, val2>, < D, 4, val4>, < G, 6, val6>**

3.2 Replica consistency

In order to achieve the consistency protocol, a replica is updated through a write protocol and requested through a read protocol. Before presenting the read/write protocol, we assume that a version can be locked or released. In addition, a node can take one of these three states: Free (F), Occupied (O) and Blocked (B). A node is free if all its versions are released. A node is occupied if it contains at least one released version. A node is blocked if all its versions are locked. The possible transitions from a state to another are illustrated in figure2.

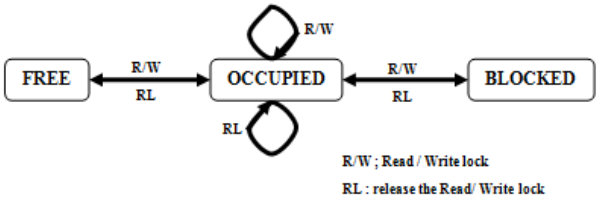


Figure 2: State of a node in a coterie.

Suppose that the initial state of the node is free (F). If a request (read/ write) is addressed to that node, the version chosen to perform the operation is locked and the node passes to occupied state (O). If this node receives another request then it keeps the same state even it still has released versions, otherwise it transits to the blocked state (B). If the node is in a blocked state and a version has been released, then the node returns to an occupied state. The node returns to a free state if all the locks of all versions are released.

3.2.1 Write protocol

*The node **N** requests the write on the data **D**.*

Choose the coterie corresponding to the data **D**.
If exist ***free*** nodes in the chosen coterie
Then
Begin
Select one of the ***free*** nodes that is near root node
If exist ***quorums*** containing the selected ***free*** node with no ***blocked*** nodes
Then
Begin
- Choose one of the existing ***quorums*** getting minimal ***occupied*** nodes and maximal ***free*** nodes.
- **Write** on the selected ***free*** node of the chosen ***quorum***. ***(See Write quorum algorithm)***

End
End If
End
Else If exist ***occupied*** nodes in the chosen coterie // an ***occupied*** node has got at least one released version
Then
Begin
Select one of the ***occupied*** nodes that is near root node
If exist ***quorums*** containing the selected ***occupied*** node with no ***blocked*** node
Then
Begin
- Choose one of the existing ***quorums*** getting minimal ***occupied*** nodes.
- **Write** on the selected ***occupied*** node of the chosen ***quorum***. ***(See Write quorum algorithm)***

End
End If
End
Else Write operation aborted.//since all nodes of the chosen coterie are ***blocked***
End If
End If

Write quorum algorithm

Write on the **oldest version** of the selected node (having the **smallest stamp**).
// propagate the written version to all nodes of the ***quorum***
For the other nodes of the chosen ***quorum*** **do**
If the latest version is locked in writing
Then abort the propagation
Else Write on the **oldest version** (having the **smallest stamp**).
End If
End For

3.2.2 Read protocol

*The node **N** requests the read on the data **D**.*

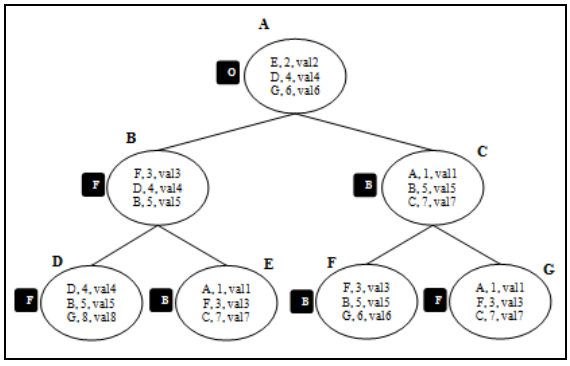
Choose the coterie corresponding to the data **D**.
If exist ***free*** nodes in the chosen coterie
Then
Begin
Select one of the ***free*** nodes that is near root node
If exist ***quorums*** containing the selected ***free*** node with no ***Blocked*** node
Then Begin
-Choose one of the existing ***quorums*** getting minimal ***occupied*** nodes and maximal ***free*** nodes

-**Read** on the selected ***free*** node of the chosen quorum. ***(See Read quorum algorithm)***
End
End If
End
Else
If exist ***occupied*** nodes in the chosen coterie
Then Begin
Select one of the ***occupied*** nodes that is near root node
If exist ***quorums*** containing the selected ***occupied*** node with no ***blocked*** node
Then Begin
- Choose one of the existing ***quorums*** getting minimal ***occupied*** nodes.
- **Read** on the ***occupied*** node of the chosen ***quorum***. ***(See Read quorum algorithm)***
End
End If
End
Else Read request aborted.// since all nodes of the chosen coterie are ***blocked***
End If
End If

Read quorum algorithm

Select the latest version of replicas (having the biggest **stamp**) of each node of the chosen ***quorum***.
Read the **selected last version.**
If there is divergence between replicas of each node of the ***quorum***
Then //propagate the selected last version to the ***quorum*** nodes.
For each node of the chosen ***quorum*** **do**
If the latest version is locked in writing
Then abort the propagation
Else Write on the **oldest version** of the selected node (having the **smallest stamp**).
End If
End For
End If

In the following examples, a coterie of a data D with 3 versions is represented with 7 nodes A, B, C, D, E, F and G having respectively the state occupied, free, blocked, free, blocked, blocked and free. An example of write and read algorithm on the coterie of data D is illustrated in the figure 3.



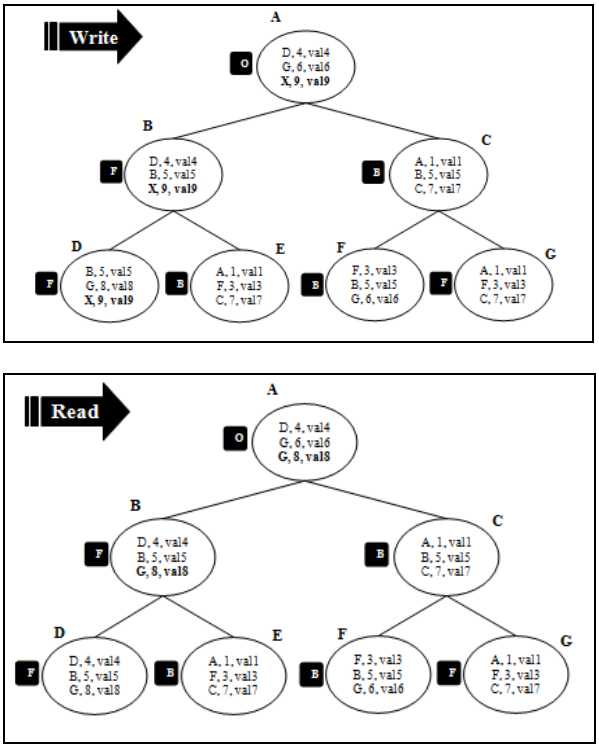


Figure 3: Write/ Read example.

Write: Suppose that at time t = 9, a Node X requests the write of the data D with the value val9. The corresponding coterie of the data D is contacted. There are three free nodes in the corresponding coterie: B, D and G but the node B is chosen because it is near the root node. Thus, two possible quorums can be designed from the node B: {A, B, D} and {A, B, E} and the quorum {A, B, D} is chosen to be written because it doesn't get blocked nodes. The oldest version (E, 2, val2) is locked to be written. At the end of the write, the lock is released and the new version (X, 9, val9) is propagated to the quorum nodes. In the propagation phase, the oldest version of each node is replaced by the new written version, for example in the node F the oldest version (D, 4, val4) is replaced by the new version (X, 9, val9).

Read: At time t = 9, a Node X requests the read of the data D. The coterie corresponding to the data D is contacted and the quorum {A, B, D} is designed. After, the latest version is chosen in the designed quorum. Among the nodes of the chosen quorum, the latest version is located in the node D (G, 8, val8). The latest version is locked to perform the reading operation. At the end of the read, the lock is released and the latest value is return to the request node. As there is a divergence between the latest versions of each node of the quorum, the latest version (G, 8, val8) must be propagated to the nodes that do not contain it. In the propagation phase, the oldest version of each node is consequently replaced by the latest version (G, 8, val8).

3.3 Load balancing strategy

Before we present our load balancing strategy, we precise how the load of a coterie is estimated.

In our approach, we define three load states of a node: **under loaded, medium loaded** and **up loaded**. In addition, we assign a numeric value to each state as follows:

Under loaded (1): the node is inactive or downloaded.
Medium loaded (2): the node is midway loaded.
Up loaded (3): the node is overloaded.

The load of a node, noted $\mathbf{L_{node}}$ is calculated by following the read/write access frequency. The node access frequency, noted $\mathbf{fa_{node}}$,is incremented at each read/ write operation and reinitialized periodically. For this, two thresholds: $\mathbf{fa_{min}}$ and $\mathbf{fa_{max}}$ are defined (The choice of the values of these thresholds will be defined in experimentations section).

$$L_{node} = 1, \text{ if } (fa_{node} < fa_{min})$$
$$L_{node} = 2, \text{ if } (fa_{min} \leq fa_{node} < fa_{max})$$
$$L_{node} = 3, \text{ if } (fa_{node} \geq fa_{max})$$

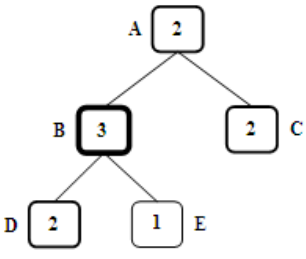
The load of a Quorum, noted $\mathbf{L_{quorum}}$, represent the maximum load of the nodes appertaining to this quorum.

$$L_{quorum} = Max_{node \in quorum}\{L_{node}\}$$

Finally, the load of a Coterie, noted $\mathbf{L_{coterie}}$, represents the sum of all quorums load of this coterie.

$$L_{coterie} = \sum_{quorum \in coterie} L_{quorum}$$

In the following example, we assume that the nodes {A, B, C, D and E} have got respectively the following load values {2, 3, 2, 2, 1}. The quorum load and the coterie load are demonstrated in the following example.



Load of nodes:

Nodes under loaded= {E}
Nodes medium loaded = {A, C, D}
Nodes up loaded = {B}

Load of quorums:

$L_{\{A, B, D\}}$ = max {2, 3, 2} = 3
$L_{\{A, B, E\}}$ = max {2, 3, 1} = 3
$L_{\{A, C\}}$ = max {2, 2} = 2

Load of coterie:

$L_{\{A, B, D\}} + L_{\{A, B, E\}} + L_{\{A, C\}}$ = 3+3+2= 8

Figure 4: Example of coterie load.

The load balancing is performed by following a dynamic reconfiguration of nodes of a coterie. The purpose of this reconfiguration is to reduce the load of quorums and the communication cost of Read / Write queries.

The communication cost represents the exchanged messages between two nodes of the grid. This cost can represent the bandwidth, the debit, the latency, etc. In our work we assume that the value of communication cost between any two nodes is provided with the file configuration of the system and it doesn't need to be calculated. As we suppose that the cost communication between grid nodes is brought by the configuration system file, the consumed communication cost of a Read/Write request can be evaluated as follow:

Write: When a node **i** writes a new value of the data D, a quorum Q of the corresponding coterie is contacted. We assume that the write cost and the size of the new value are negligible. The nodes appertaining to this quorum communicate with each other following a propagation phase then an acknowledgement phase.

$$com\_cost_{Write\_Qi} = (N-1) \times \left(\sum_{\substack{i \neq j \\ j \in Q}} prop\_cost_{i,j} + \sum_{\substack{i \neq j \\ j \in Q}} ack\_cost_{j,i} \right)$$

Where:
N: number of nodes appertaining to the quorum Q
$\boldsymbol{Com\_cost_{Write\_Qi}}$**:** communication cost of the write quorum initiated by the node i
$\boldsymbol{Prop\_cost_{i,j}}$: propagation cost of the new value from the node i to the node j
$\boldsymbol{Ack\_cost_{j,i}}$**:** acknowledgment cost from the node j to the node i

Read: When a read request is invoked by a node **i**, a quorum Q of the corresponding coterie is selected. Nodes appertaining to this quorum communicate to each other following a selection of latest version then the propagation of the selected last version then the acknowledgement phase.

$$com\_cost_{Read\_Qi} = (N-1) \times \left(\sum_{\substack{i \neq j \\ j \in Q}} slc_{cost_{j,i}} + \sum_{\substack{i \neq j \\ j \in Q}} prop_{cost_{i,j}} + \sum_{\substack{i \neq \\ j \in Q}} ack\_cost_{j,i} \right)$$

Where:
N: number of node appertaining to the quorum Q
$\boldsymbol{Com\_cost_{Write\_Qi}}$ **:** communication cost of the read quorum initiated by the node i
$\boldsymbol{Slc\_cost_{j,i}}$ **:** selection cost of the latest version to be red from the node j to the node i
$\boldsymbol{Prop\_cost_{i,j}}$ *:* propagation cost of the new value from the node i to the node j
$\boldsymbol{Ack\_cost_{j,i}}$**:** acknowledgment cost from the node j to the node i

Our load balancing strategy is achieved with a dynamic reconfiguration of nodes of the coterie from the root node to the leaf nodes. This reconfiguration is based on an elementary permutation between a parent node and its two son nodes so that the load of the father is greater than the load of its two nodes son getting a minimal communication cost. The main objective of this reconfiguration is to involve at least possible the overloaded nodes in the construction of quorums. In this way, we get overloaded nodes at the lowest level of the coterie (leaves level) and this without degrading the performance of consistency in terms of load and

communication cost. This reconfiguration is triggered at each Read/ Write request.

Load balancing strategy

Input: structured coterie, load nodes of each coterie.
Output: restructured coterie.

For each coterie **do**
 // seek the tree from the root to the leaf nodes
 For each parent node **do**
 // N_1, N_2 {child$_1$, child$_2$} and $N_1 \neq N_2$
 If (load $_{parent} \geq$ **max** (load $_{N1}$, load $_{N2}$)
 Then If ((cost $_{(N1, N2)} \leq$ cost $_{(parent, N2)}$) and
 (cost $_{(N1, N2)} >$ cost $_{(parent, N1)}$)
 Then Swap (Parent, N_1)
 Else If ((cost $_{(N1, N2)} \leq$ cost $_{(parent, N1)}$) and
 (cost $_{(N1, N2)} >$ cost $_{(parent, N2)}$)
 Then Swap (parent, N_2)
 End if
 End if
 Else If (load $_{parent} \geq$ load $_{N1}$) and
 (cost $_{(N1, N2)} \geq$ cost $_{(parent, N2)}$)
 Then Swap (Parent, N_1)
 Else if ((load $_{parent} >$ load $_{N2}$) and
 (cost $_{(N1, N2)} \geq$ cost $_{(parent, N1)}$)
 Then Swap (parent, N_2)
 End if
 End if
 End if
 End For
End For

The following example illustrates a coterie composed of nine nodes (A, B, C, D, E, F, G, H, I). Each node is represented by its load. The load of the coterie is calculated from the sum of the quorum's load. Thus, the load of the coterie before the reconfiguration is equal to 15.

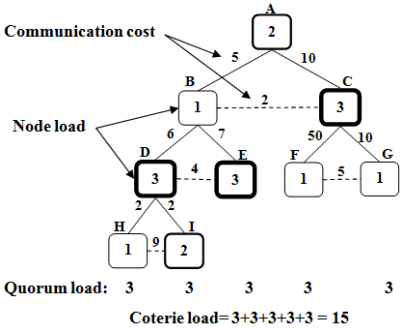


Figure 5: Load of the coterie before reconfiguration.

The reconfiguration of this coterie is performed from the root to a leaf node of the coterie, by considering three nodes: parent node and its two child nodes. Four cases are distinguished as follow:

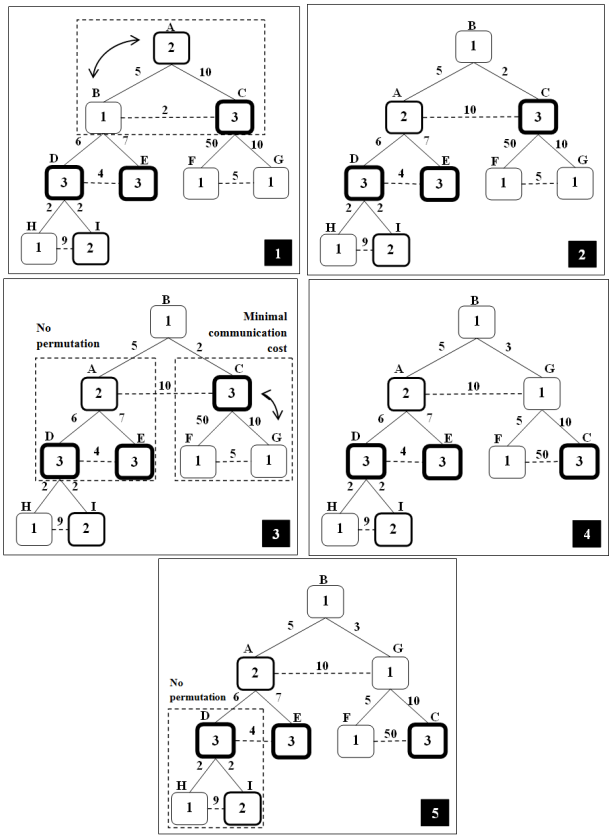


Figure 6: Reconfiguration of a coterie.

In Figure 6.1 as the parent node load (node A) is greater than the child node load (node B) the permutation between Node A (father) and Node B (son) which brings a minimal communication cost is illustrated in Figure 6.2.

In Figure 6.3 as the parent load (node A) is less than its two child loads (node D and E) then no permutation will be made. In Figure 6.3 the parent load (node C) is greater than its two child loads (F and G) then the child node G is permuted with the parent node C which brings a minimal communication cost, as shown in Figure 6.4.

In Figure 6.5 in spite the parent load (node D) is greater than the load of its two Childs (node H and I); no permutation is possible because this will increase the cost of communication instead of reducing it. The load of the coterie after reconfiguration became equal to 13.

In addition, to show how the cost of communication is optimized we suppose that a write request is addressed to the node F with the value V. So the communication cost of a write request is estimated as defined in section 3.3.

We conclude that the communication cost is reduced from 180 to 26 by using our load balancing strategy with taking care of a minimal communication cost.

We present the simulation results of our proposed approach in the next section.

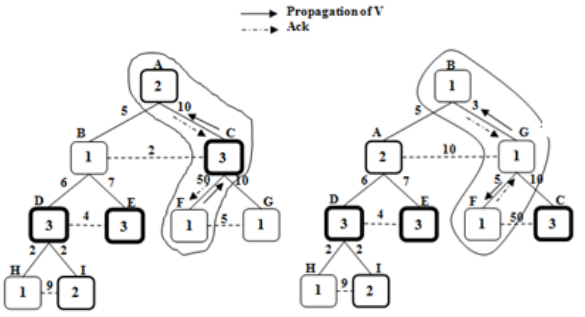


Figure 7: Communication cost before and after reconfiguration.

4 Experiments

We evaluate the performances of our consistency approach by using the grid simulator ***Gridsim toolkit 4.2*** [21] under the operating system ***Windows XP 7***.We defined different number of grid nodes. We replicate arbitrary **50 data** into **5 versions** over each node of the grid.

In order to estimate the load of a node, we define two thresholds $\mathbf{fa_{min}}$ and $\mathbf{fa_{max}}$ by following the number of nodes and the number of transactions (Read / Write requests). After many experiments, we conclude that the approximation of the $\mathbf{fa_{min}}$ that gives best results is calculated as follow:

If (**number_transactions >= number_nodes)**
Then $\mathbf{fa_{min}}$ =0
Else $\mathbf{fa_{min}}$= **number_transactions / number_nodes**.

To consider the uploaded state of a node, we approximate the $\mathbf{fa_{max}}$ to $\mathbf{fa_{min}}$**+3**.

# Data= 50, # Version=5		
# Experiment	**# Nodes**	**# Read/ Write Transactions**
experiment 1	**500**	**200, 500, 1000, 5000, 10000**
experiment 2	**1000**	**500, 1000, 5000, 10000, 50000**
experiment 3	**5000**	**1000, 5000, 10000, 50000, 100000**

Table 1: Simulation parameters.

In order to study the consistency results, we assume that a replica is consistent if its latest version corresponds to the latest written value. Thus, the consistency of a data D is calculated as below:

$$Consistency_D = \frac{number\ of\ nodes\ hosting\ consistent\ replicas\ of\ the\ data\ «\ D\ »}{total\ number\ of\ nodes\ hosting\ the\ replicas\ of\ the\ data\ «\ D\ »}$$

In figure 8, we note that the consistency is variable, this is explained by the fact that when there is a lot of write operations, the updates of replicas becomes more difficult and consequently inconsistencies occur. That is why we study the freshness of replicas.

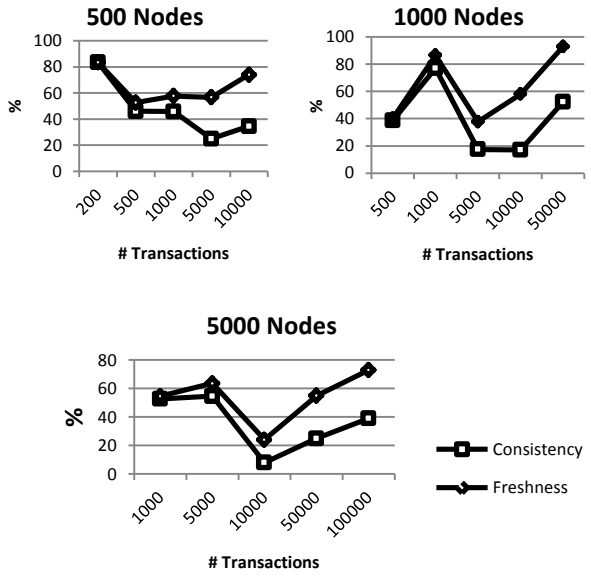


Figure 8: Consistency& Freshness.

Considering a set of replicas of the data **D** = {R_0, R_1… Rn} where: R_0 represents the first written replica and Rn the latest written replica. The freshness margin of a replica Ri is equal to n-i. A replica is assumed to be « fresh » if its freshness margin is lower than n/2. The freshness of a data ***D*** is calculated as below:

$$Freshness_D = \frac{\text{number of nodes hosting fresh replicas of the data « D »}}{\text{total number of nodes hosting the replicas of the data « D »}}$$

In fact, the experimentation results show that with the presence of inconsistent copies, the system still hold fresh replicas, as shown in figure 8.

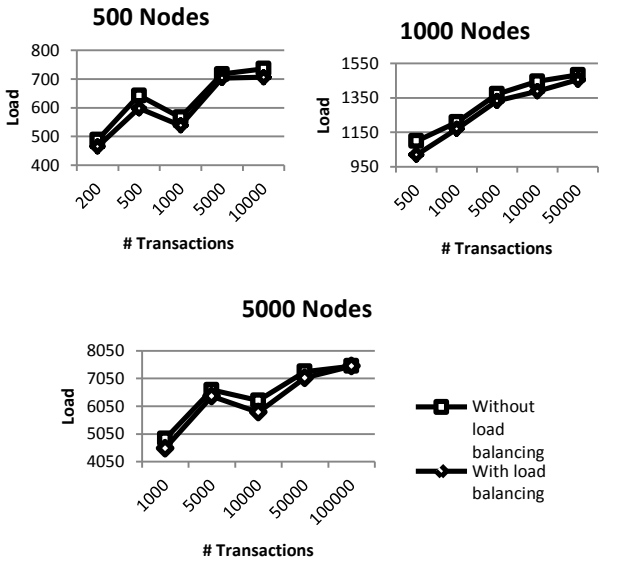


Figure 9: Load balancing results 1.

In figure 9, we note that when the number of transactions is significantly higher than the number of nodes, the average load of coteries increase, otherwise the load is variable. Also, the load of coteries is reduced when our load balancing strategy is used.

In the same time, the communication cost consumed for R/W accesses is significantly reduced as shown in figure 10.

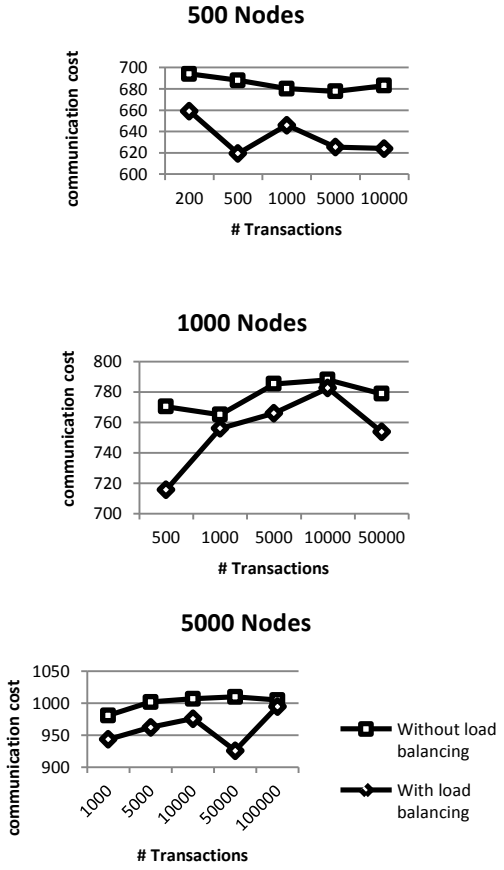


Figure 10: Load balancing results 2.

We conclude the main objectives of our work were gained in term of consistency, load balancing and communication cost.

The experiments results demonstrate that our approach guarantee the freshness when replicas are not strongly consistent. Also, when the load balancing results are studied, we note that the load of coteries and the communication cost of R/W queries are significantly reduced when the load balancing strategy is applied.

5 Conclusion

In this paper we presented our contribution to improve replica consistency trough a dynamic load balancing strategy. The use of a structured tree (coterie) allows a better logical organization of the grid nodes hosting a set of replicas. The definition of multiple versions of a replica can serve as many grid users as available versions and with a certain degree of similarity with the last update of the replicated data. Quorum structure ensures the existence at any time of the latest version of the replicated data. This is explained by the fact that the root node always has the latest version. Indeed, during a read / write operation, the latest version is always propagated to the quorum nodes designated for reading / writing. As a quorum is built from the root to a leaf of the tree, then the root node participates in any designed quorum. The consistency between replicas is not strong in our work in order to serve a maximum of read request. Despite this divergence between copies, the grid still holds fresh replicas.

Reconfiguration of the coterie provides better load balancing to increase access performance. The obtained results of our approach reveal that the consistency management of replicas is balanced dynamically following the state of each node of the coterie. The obtained results demonstrated the efficiency of our work in term of consistency, load balancing and communication cost of Read/ Write queries.

Finally, our work can present many perspectives; we cite the most interesting ones:

- Study the impact of different replica placement on the consistency performances
- Assure the fault tolerance of the root node of each coterie

References

[1] Alan D. Fekete and Krithi Ramamritham, « Consistency Models for Replicated Data», Replication Lecture Notes in Computer Science Volume 5959, 2010, pp 1-17

[2] Cécile Le Pape and Stéphane Gançarski, « Replica Refresh Strategies in a Database Cluster ». LIP6, LNCS 4395, pp. 679–691, 2007.

[3] Changqin Huang, Fuyin Xu, and Xiaoyong Hu,«Massive Data Oriented Replication Algorithms for Consistency Maintenance in Data Grids», Part I, LNCS 3991, pp. 838 – 841, 2006.

[4] Chao-Tung Yang Wen-Chi Tsai Tsui-Ting Chen Ching-Hsien Hsu, «A One-way File Replica Consistency Model in Data Grids» Tunghai University, Taiwan. IEEE Asia-Pacific Services Computing Conference 2007.

[5] Christian Storm, « Specification of Quorum Systems », Specification and Analytical Evaluation of Heterogeneous Dynamic Quorum-Based Data Replication Schemes, 2012, pp 81-153

[6] Esther Pacitti, Patrick Valduriez, Marta Mattoso, Grid Data Management: Open Problems and New Issues, Journal of Grid Computing, September 2007, Volume 5, Issue 3, pp 273-281

[7] Foster, I.: « What is the Grid ? A Three Point Checklist », Grid Today, 1(6), (2002).

[8] Gabriel Antoniu, Jean François Deverge and Sébastien Monnet,« How to bring together fault tolerance and data»,INRIA research report 2005.

[9] H. Guo and al. «Relaxed currency and consistency: How to say good enough in sql». H. Guo and al. In ACM SIGMOD int. conf., 2004.

[10] Hartmut Kaiser, Kathrin Kirsch, and Andre erzky, «Versioning and Consistency in Replica Systems», LNCS 4331, pp. 618–627, 2006.

[11] Hector Garcia-Molina and Daniel Barabara, «How to assign votes in a distributed system». Journal of the ACM, 32(4) :841–860, October 1985.
[12] Ivan Frain, Jean-Paul Bahsoun, Abdelaziz M'zoughi,« dynamic reconfiguration of a coterie tree-structured »,RNTL ViSaGe project, CDUR 2005
[13] James J. (Jong Hyuk) Park et al. «Data Consistency for Self-acting Load Balancing of Parallel File System», ITCS & STA 2012, LNEE 180, pp. 135–143, DOI: 10.1007/978-94-007-5082-1_18.
[14] Jinchuan Chen, Yueguo Chen, Xiaoyong Du, Cuiping Li, Jiaheng Lu, Suyun Zhao, Xuan Zhou, « Big data challenge: a data management perspective », Frontiers of Computer Science, April 2013, Volume 7, Issue 2, pp 157-164
[15] Liviu Iftode, Jaswinder Pal Singh, and Kai Li. «Scope consistency: A bridge between release consistency and entry consistency». In Proceedings of the 8th ACM Annual Symposium on Parallel Algorithms and Architectures (SPAA '96), pages 277.287, Padova, Italy, June 1996.
[16] N. A. Lynch and A. A. Shvartsman. «Robust emulation of shared memory using dynamic quorum-acknowledged broadcasts». In FTCS '97 : Proceedings of the 27th International Symposium on Fault-Tolerant Computing (FTCS '97). IEEE Computer Society,1997.
[17] N. Tziritas et al. «Implementing Replica Placements: Feasibility and Cost Minimization», in IPDPS 2007
[18] N. Tziritas et al. «Using Multicast Transfers in the Replica Migration Problem: Formulation and Scheduling Heuristics», Europar 2009.
[19] Peter Urban, Xavier Défago, and André Schiper. «Neko: A single environment to simulate and prototype distributed algorithms». In 15th Int'l Conference on Information Networking (ICOIN-15), pages 503–511, 2001.
[20] Quang Hieu Vu, Mihai Lupu, Beng Chin Ooi, «Load Balancing and Replication», Peer-to-Peer Computing, 2010, pp 127-156
[21] R. Buyya and M. Murshed., «GridSim: a toolkit for the modeling and simulation of Distributed resource management and scheduling for grid computing». Concurrency computat: pract. Exper., 2002
[22] S. Khan et al. « Robust CDN Replica Placement Techniques », IPDPS 2009
[23] S. Khan et al. «A Pure Nash Equilibrium-Based Game Theoretical Method for Data Replication across Multiple Servers», in TKDE 2009.
[24] Saito, Y., Shapiro, M.:«Optimistic replication. Comput. Surveys» 37(1), 42–81. 2005
[25] Uroš Čibej, Anthony Sulistio, Rajkumar Buyya, « Grid Computing», Parallel Computing, 2009, pp 117-145

Searching for Credible Relations in Machine Learning

Vedrana Vidulin
Department of Intelligent Systems, Jožef Stefan Institute, Jamova cesta 39, Ljubljana, Slovenia
E-mail: vedrana.vidulin@ijs.si; Web: http://dis.ijs.si/Vedrana

Thesis Summary

Keywords: interactive machine learning, credible relations, meaning

Received: June 8, 2013

This paper presents a summary of the doctoral dissertation of the author on the topic of searching for credible relations in machine learning.

Povzetek: Članek predstavlja povzetek doktorske disertacije avtorja, ki obravnava temo iskanja verodostojnih relacij v strojnem učenju.

1 Introduction

When machine learning (ML) and data mining (DM) methods construct models in complex domains, models can contain less-credible parts [2], which are statistically significant, but meaningless to the human analyst. For example, let us consider a decision tree model presented in Figure 1. The tree is constructed with the J48 algorithm in Weka [8] for a complex domain indicating which segments of research and development (R&D) sector have the highest impact on economic welfare of a country. Nodes in the tree represent segments of the R&D sector. Leaves in the tree represent economic welfare of the majority of countries that reached the specific leaf. Economic welfare can be: low, middle or high. In each leaf, the first number in brackets represents the number of countries that reached that leaf. The second number represents the number of countries in that leaf with the level of welfare different than the one represented by the leaf. The quantities are expressed in decimals to account for those countries with missing values for segments appearing in the tree. Note that the left subtree is omitted to simplify the example.

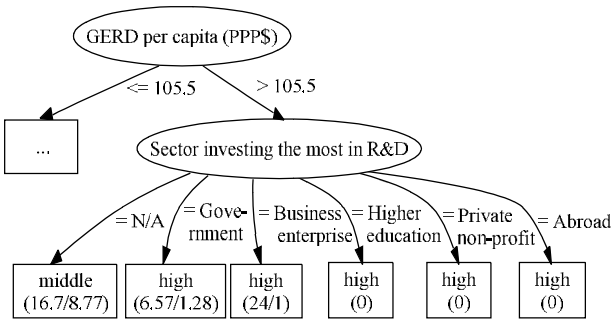


Figure 1: Decision tree constructed with J48 algorithm.

The tree represents two relations, one defining how the level of investment in R&D ("GERD per capita (PPP$)") is related to economic welfare and another defining how sector that invests the most in R&D is related to the welfare. The relation including sector is an example of a less-credible relation. It is statistically significant because the ML method included it in the tree. However, it is meaningless, since the single "middle" leaf represents the countries for which the sector is unknown ("N/A" value), while all other countries have high level of welfare. In other words, the subtree does not bring any additional information to the human analyst.

To eliminate less-credible relations from the models, both automatic and interactive approaches were suggested. Examples of the former are the pruning of decision trees [4] and the correction of a quality estimate to eliminate the random classification rules with optimistically high values of quality [3]. Typical examples of the latter provide improvements in the form of new training examples [1] or a list of attributes that would better describe the class [5]. The presented approaches aim to improve the model's predictive performance, while meaningless relations can remain a part of the model, as long as they positively influence the quality. The resulting models are not acceptable when the task is domain analysis.

Therefore, we proposed a novel method that constructs multiple models in an algorithmic way, enabling the human analyst to examine interesting relations from different angles and in different contexts, and based on additional evidence to conclude which relations are indeed credible.

2 Human-machine data mining

The interactive method for the construction of credible relations in complex domains, proposed in the thesis [6], is named Human-Machine Data Mining (HMDM). The basic idea of our method is to construct a large number of models to extract the credible relations, i.e., relations that are both meaningful and of high quality. The task is computationally very demanding, and for other than simple cases there is no possibility for humans to analyse

a meaningful share of all the hypothesized models on their own. However, the introduced combination of human understanding and raw computer power enables a smart examination of the parts of the huge search space with most credible relations. While ML and DM methods perform the search, humans examine and evaluate the results, make conclusions and redo the search in a way that seems to be the most promising based on the previous attempts. In this way, the humans guide the DM to search the subspaces with the most credible relations and finally the humans construct the overall conclusions from the various, most interesting solutions.

The HMDM defines a toolbox composed of semi-automated DM procedures and a set of scenarios for the human to guide the analysis towards credible relations. Furthermore, it defines a scheme for the extraction of credible relations from multiple models, which provides support to the human analyst in the process of constructing correct conclusions about the domain.

3 Evaluation

The proposed method was demonstrated in two complex domains that show how the higher education and the R&D sectors are related to economic welfare [7]. With the HMDM method we were able to extract relations well-established in the literature, which shows the capability of the method to find important relations in the domain. We also extracted new relations that were not previously addressed in the literature. In addition, we showed in a domain of automatic web genre identification that HMDM can be successfully used for learning predictive models in another domain.

A user study justified the HMDM method by showing that the users are frequently not able to detect meaningless relations by observing a single model constructed by a ML algorithm. However, by observing interesting variations, i.e., candidate solutions suggested by the HMDM method, the participants realized the weaknesses of the default model and created better domain models.

4 Conclusions

The thesis made several contributions to the area of ML and application areas of macroeconomics and automatic web genre identification. First, we proposed a new method Human-Machine Data Mining for extracting credible relations from data, based on interactive and iterative processes exploiting advantages of both human and artificial intelligence. Second, the quality measure corrected class probability estimate was adjusted to decision tree models. Third, interactive explanations of DM results were designed, conceived to facilitate the extraction of credible relations. Fourth, a computer program was developed to support the HMDM method. Fifth, for two real-life domains, showing how the higher education and R&D sector influence the economic welfare of a country, we extracted credible relations with the new method, confirming some well-known relations and providing some new ones. Finally, for the real-life domain of automatic web genre identification, we constructed credible models with the new method, which provide an insight into the role of content words in recognizing web genres.

References

[1] J.A. Fails, D.R. Olsen Jr. (2003) Interactive Machine Learning, In: *Proceedings of the 8th International Conference on Intelligent User Interfaces*, Miami, FL, pp. 39-45.

[2] D. Jensen, P. Cohen (2000) Multiple Comparisons in Induction Algorithms. *Machine Learning* 38(3):309-338.

[3] M. Možina, J. Demšar, J. Žabkar, I. Bratko (2006), Why is Rule Learning Optimistic and How to Correct It, In: *Machine Learning: ECML 2006*, Springer, Berlin, pp. 330-340.

[4] J.R. Quinlan (1993) C4.5: Programs for Machine Learning, Morgan Kaufmann, San Mateo, CA.

[5] S. Stumpf, V. Rajaram, L. Li, W. Wong, M. Burnett, T. Dietterich, E. Sullivan, J. Herlocker (2009) Interacting Meaningfully with Machine Learning Systems: Three Experiments, *International Journal of Human–Computer Studies* 67:639–662.

[6] V. Vidulin (2012) *Searching for Credible Relations in Machine Learning*, PhD Thesis, IPS Jožef Stefan, Ljubljana, Slovenia.

[7] V. Vidulin, M. Gams (2011) Impact of High-Level Knowledge on Economic Welfare through Interactive Data Mining. *Applied Artificial Intelligence* 25(4):267-291.

[8] I. Witten, E. Frank (2005) *Data Mining. Practical Machine Learning Tools and Techniques*, Morgan Kaufmann, San Francisco, CA.

JOŽEF STEFAN INSTITUTE

Jožef Stefan (1835-1893) was one of the most prominent physicists of the 19th century. Born to Slovene parents, he obtained his Ph.D. at Vienna University, where he was later Director of the Physics Institute, Vice-President of the Vienna Academy of Sciences and a member of several scientific institutions in Europe. Stefan explored many areas in hydrodynamics, optics, acoustics, electricity, magnetism and the kinetic theory of gases. Among other things, he originated the law that the total radiation from a black body is proportional to the 4th power of its absolute temperature, known as the Stefan–Boltzmann law.

The Jožef Stefan Institute (JSI) is the leading independent scientific research institution in Slovenia, covering a broad spectrum of fundamental and applied research in the fields of physics, chemistry and biochemistry, electronics and information science, nuclear science technology, energy research and environmental science.

The Jožef Stefan Institute (JSI) is a research organisation for pure and applied research in the natural sciences and technology. Both are closely interconnected in research departments composed of different task teams. Emphasis in basic research is given to the development and education of young scientists, while applied research and development serve for the transfer of advanced knowledge, contributing to the development of the national economy and society in general.

At present the Institute, with a total of about 900 staff, has 700 researchers, about 250 of whom are postgraduates, around 500 of whom have doctorates (Ph.D.), and around 200 of whom have permanent professorships or temporary teaching assignments at the Universities.

In view of its activities and status, the JSI plays the role of a national institute, complementing the role of the universities and bridging the gap between basic science and applications.

Research at the JSI includes the following major fields: physics; chemistry; electronics, informatics and computer sciences; biochemistry; ecology; reactor technology; applied mathematics. Most of the activities are more or less closely connected to information sciences, in particular computer sciences, artificial intelligence, language and speech technologies, computer-aided design, computer architectures, biocybernetics and robotics, computer automation and control, professional electronics, digital communications and networks, and applied mathematics.

The Institute is located in Ljubljana, the capital of the independent state of S**love**nia (or S♡nia). The capital today is considered a crossroad between East, West and Mediterranean Europe, offering excellent productive capabilities and solid business opportunities, with strong international connections. Ljubljana is connected to important centers such as Prague, Budapest, Vienna, Zagreb, Milan, Rome, Monaco, Nice, Bern and Munich, all within a radius of 600 km.

From the Jožef Stefan Institute, the Technology park "Ljubljana" has been proposed as part of the national strategy for technological development to foster synergies between research and industry, to promote joint ventures between university bodies, research institutes and innovative industry, to act as an incubator for high-tech initiatives and to accelerate the development cycle of innovative products.

Part of the Institute was reorganized into several high-tech units supported by and connected within the Technology park at the Jožef Stefan Institute, established as the beginning of a regional Technology park "Ljubljana". The project was developed at a particularly historical moment, characterized by the process of state reorganisation, privatisation and private initiative. The national Technology Park is a shareholding company hosting an independent venture-capital institution.

The promoters and operational entities of the project are the Republic of Slovenia, Ministry of Higher Education, Science and Technology and the Jožef Stefan Institute. The framework of the operation also includes the University of Ljubljana, the National Institute of Chemistry, the Institute for Electronics and Vacuum Technology and the Institute for Materials and Construction Research among others. In addition, the project is supported by the Ministry of the Economy, the National Chamber of Economy and the City of Ljubljana.

Jožef Stefan Institute
Jamova 39, 1000 Ljubljana, Slovenia
Tel.:+386 1 4773 900, Fax.:+386 1 251 93 85
WWW: http://www.ijs.si
E-mail: matjaz.gams@ijs.si
Public relations: Polona Strnad

INFORMATICA

AN INTERNATIONAL JOURNAL OF COMPUTING AND INFORMATICS

INVITATION, COOPERATION

Submissions and Refereeing

Please submit a manuscript at: http://www.informatica.si/Editors/PaperUpload.asp. At least two referees outside the author's country will examine it, and they are invited to make as many remarks as possible from typing errors to global philosophical disagreements. The chosen editor will send the author the obtained reviews. If the paper is accepted, the editor will also send an email to the managing editor. The executive board will inform the author that the paper has been accepted, and the author will send the paper to the managing editor. The paper will be published within one year of receipt of email with the text in Informatica MS Word format or Informatica LaTeX format and figures in .eps format. Style and examples of papers can be obtained from http://www.informatica.si. Opinions, news, calls for conferences, calls for papers, etc. should be sent directly to the managing editor.

QUESTIONNAIRE

☐ Send Informatica free of charge

☐ Yes, we subscribe

Please, complete the order form and send it to Dr. Drago Torkar, Informatica, Institut Jožef Stefan, Jamova 39, 1000 Ljubljana, Slovenia. E-mail: drago.torkar@ijs.si

Since 1977, Informatica has been a major Slovenian scientific journal of computing and informatics, including telecommunications, automation and other related areas. In its 16th year (more than nineteen years ago) it became truly international, although it still remains connected to Central Europe. The basic aim of Informatica is to impose intellectual values (science, engineering) in a distributed organisation.

Informatica is a journal primarily covering intelligent systems in the European computer science, informatics and cognitive community; scientific and educational as well as technical, commercial and industrial. Its basic aim is to enhance communications between different European structures on the basis of equal rights and international refereeing. It publishes scientific papers accepted by at least two referees outside the author's country. In addition, it contains information about conferences, opinions, critical examinations of existing publications and news. Finally, major practical achievements and innovations in the computer and information industry are presented through commercial publications as well as through independent evaluations.

Editing and refereeing are distributed. Each editor can conduct the refereeing process by appointing two new referees or referees from the Board of Referees or Editorial Board. Referees should not be from the author's country. If new referees are appointed, their names will appear in the Refereeing Board.

Informatica is free of charge for major scientific, educational and governmental institutions. Others should subscribe (see the last page of Informatica).

ORDER FORM – INFORMATICA

Name: .....

Title and Profession (optional): .....

.....

Home Address and Telephone (optional): .....

.....

Office Address and Telephone (optional): .....

.....

E-mail Address (optional): .....

Signature and Date: .....

Informatica WWW:

http://www.informatica.si/

Referees from 2008 on:

Ajith Abraham, Siby Abraham, Renato Accornero, Raheel Ahmad, Cutting Alfredo, Hameed Al-Qaheri, Gonzalo Alvarez, Wolfram Amme, Nicolas Anciaux, Rajan Arora, Costin Badica, Zoltán Balogh, Andrea Baruzzo, Borut Batagelj, Norman Beaulieu, Paolo Bellavista, Steven Bishop, Marko Bohanec, Zbigniew Bonikowski, Borko Bosković, Marco Botta, Pavel Brazdil, Johan Brichau, Andrej Brodnik, Ivan Bruha, Maurice Bruynooghe, Wray Buntine, Dumitru Dan Burdescu, Yunlong Cai, Juan Carlos Cano, Tianyu Cao, Norman Carver, Marc Cavazza, Jianwen Chen, L.M. Cheng, Chou Cheng-Fu, Girija Chetty, G. Chiola, Yu-Chiun Chiou, Ivan Chorbev, Shauvik Roy Choudhary, Sherman S.M. Chow, Lawrence Chung, Mojca Ciglarič, Jean-Noël Colin, Vittorio Cortellessa, Jinsong Cui, Alfredo Cuzzocrea, Darko Čerepnalkoski, Gunetti Daniele, Grégoire Danoy, Manoranjan Dash, Paul Debevec, Fathi Debili, Carl James Debono, Joze Dedic, Abdelkader Dekdouk, Bart Demoen, Sareewan Dendamrongvit, Tingquan Deng, Anna Derezinska, Gaël Dias, Ivica Dimitrovski, Jana Dittmann, Simon Dobrišek, Quansheng Dou, Jeroen Doumen, Erik Dovgan, Branko Dragovich, Dejan Drajic, Jozo Dujmovic, Umut Riza Ertürk, CHEN Fei, Ling Feng, YiXiong Feng, Bogdan Filipič, Iztok Fister, Andres Flores, Vladimir Fomichov, Stefano Forli, Massimo Franceschet, Alberto Freitas, Jessica Fridrich, Scott Friedman, Chong Fu, Gabriel Fung, David Galindo, Andrea Gambarara, Matjaž Gams, Maria Ganzha, Juan Garbajosa, Rosella Gennari, David S. Goodsell, Jaydeep Gore, Miha Grčar, Daniel Grosse, Zhi-Hong Guan, Donatella Gubiani, Bidyut Gupta, Marjan Gusev, Zhu Haiping, Kathryn Hempstalk, Gareth Howells, Juha Hyvärinen, Dino Ienco, Natarajan Jaisankar, Domagoj Jakobovic, Imad Jawhar, Yue Jia, Ivan Jureta, Dani Juričić, Zdravko Kačič, Slobodan Kalajdziski, Yannis Kalantidis, Boštjan Kaluža, Dimitris Kanellopoulos, Rishi Kapoor, Andreas Kassler, Daniel S. Katz, Samee U. Khan, Mustafa Khattak, Elham Sahebkar Khorasani, Ivan Kitanovski, Tomaž Klobučar, Ján Kollár, Peter Korošec, Valery Korzhik, Agnes Koschmider, Jure Kovač, Andrej Krajnc, Miroslav Kubat, Matjaz Kukar, Anthony Kulis, Chi-Sung Laih, Niels Landwehr, Andreas Lang, Mohamed Layouni, Gregor Leban, Alex Lee, Yung-Chuan Lee, John Leggett, Aleš Leonardis, Guohui Li, Guo-Zheng Li, Jen Li, Xiang Li, Xue Li, Yinsheng Li, Yuanping Li, Shiguo Lian, Lejian Liao, Ja-Chen Lin, Huan Liu, Jun Liu, Xin Liu, Suzana Loskovska, Zhiguo Lu, Hongen Lu, Mitja Luštrek, Inga V. Lyustig, Luiza de Macedo, Matt Mahoney, Domen Marinčič, Dirk Marwede, Maja Matijasevic, Andrew C. McPherson, Andrew McPherson, Zuqiang Meng, France Mihelič, Nasro Min-Allah, Vojislav Misic, Vojislav Mišić, Mihai L. Mocanu, Angelo Montanari, Jesper Mosegaard, Martin Možina, Marta Mrak, Yi Mu, Josef Mula, Phivos Mylonas, Marco Di Natale, Pavol Navrat, Nadia Nedjah, R. Nejabati, Wilfred Ng, Zhicheng Ni, Fred Niederman, Omar Nouali, Franc Novak, Petteri Nurmi, Denis Obrul, Barbara Oliboni, Matjaž Pančur, Wei Pang, Gregor Papa, Marcin Paprzycki, Marek Paralič, Byung-Kwon Park, Torben Bach Pedersen, Gert Schmeltz Pedersen, Zhiyong Peng, Ruggero G. Pensa, Dana Petcu, Marko Petkovšek, Rok Piltaver, Vid Podpečan, Macario Polo, Victor Pomponiu, Elvira Popescu, Božidar Potočnik, S. R. M. Prasanna, Kresimir Pripuzic, Gabriele Puppis, HaiFeng Qian, Lin Qiao, Jean-Jacques Quisquater, Vladislav Rajkovič, Dejan Rakovic, Jean Ramaekers, Jan Ramon, Robert Ravnik, Wilfried Reimche, Blagoj Ristevski, Juan Antonio Rodriguez-Aguilar, Pankaj Rohatgi, Wilhelm Rossak, Eng. Sattar Sadkhan, Sattar B. Sadkhan, Khalid Saeed, Motoshi Saeki, Evangelos Sakkopoulos, M. H. Samadzadeh, MariaLuisa Sapino, Piervito Scaglioso, Walter Schempp, Barabara Koroušić Seljak, Mehrdad Senobari, Subramaniam Shamala, Zhongzhi Shi, LIAN Shiguo, Heung-Yeung Shum, Tian Song, Andrea Soppera, Alessandro Sorniotti, Liana Stanescu, Martin Steinebach, Damjan Strnad, Xinghua Sun, Marko Robnik Šikonja, Jurij Šilc, Igor Škrjanc, Hotaka Takizawa, Carolyn Talcott, Camillo J. Taylor, Drago Torkar, Christos Tranoris, Denis Trček, Katarina Trojacanec, Mike Tschierschke, Filip De Turck, Aleš Ude, Wim Vanhoof, Alessia Visconti, Vuk Vojisavljevic, Petar Vračar, Valentino Vranić, Chih-Hung Wang, Huaqing Wang, Hao Wang, Hui Wang, YunHong Wang, Anita Wasilewska, Sigrid Wenzel, Woldemar Wolynski, Jennifer Wong, Allan Wong, Stefan Wrobel, Konrad Wrona, Bin Wu, Xindong Wu, Li Xiang, Yan Xiang, Di Xiao, Fei Xie, Yuandong Yang, Chen Yong-Sheng, Jane Jia You, Ge Yu, Borut Zalik, Aleš Zamuda, Mansour Zand, Zheng Zhao, Dong Zheng, Jinhua Zheng, Albrecht Zimmermann, Blaž Zupan, Meng Zuqiang

Informatica

An International Journal of Computing and Informatics

Web edition of Informatica may be accessed at: http://www.informatica.si.

Subscription Information Informatica (ISSN 0350-5596) is published four times a year in Spring, Summer, Autumn, and Winter (4 issues per year) by the Slovene Society Informatika, Litostrojska cesta 54, 1000 Ljubljana, Slovenia.
The subscription rate for 2013 (Volume 37) is
– 60 EUR for institutions,
– 30 EUR for individuals, and
– 15 EUR for students
Claims for missing issues will be honored free of charge within six months after the publication date of the issue.

Typesetting: Borut Žnidar.
Printing: ABO grafika d.o.o., Ob železnici 16, 1000 Ljubljana.

Orders may be placed by email (drago.torkar@ijs.si), telephone (+386 1 477 3900) or fax (+386 1 251 93 85). The payment should be made to our bank account no.: 02083-0013014662 at NLB d.d., 1520 Ljubljana, Trg republike 2, Slovenija, IBAN no.: SI56020830013014662, SWIFT Code: LJBASI2X.

Informatica is published by Slovene Society Informatika (president Niko Schlamberger) in cooperation with the following societies (and contact persons):
Robotics Society of Slovenia (Jadran Lenarčič)
Slovene Society for Pattern Recognition (Janez Perš)
Slovenian Artificial Intelligence Society (Dunja Mladenić)
Cognitive Science Society (Urban Kordeš)
Slovenian Society of Mathematicians, Physicists and Astronomers (Andrej Likar)
Automatic Control Society of Slovenia (Sašo Blažič)
Slovenian Association of Technical and Natural Sciences / Engineering Academy of Slovenia (Vojteh Leskovšek)
ACM Slovenia (Andrej Brodnik)

Informatica is surveyed by: ACM Digital Library, Citeseer, COBISS, Compendex, Computer & Information Systems Abstracts, Computer Database, Computer Science Index, Current Mathematical Publications, DBLP Computer Science Bibliography, Directory of Open Access Journals, InfoTrac OneFile, Inspec, Linguistic and Language Behaviour Abstracts, Mathematical Reviews, MatSciNet, MatSci on SilverPlatter, Scopus, Zentralblatt Math

Volume 37 Number 3 September 2013 **ISSN 0350-5596**

Informatica

An International Journal of Computing and Informatics

Informatica **37** (2013) Number 3, pp. 221–357