INTERNATIONAL JOURNAL

of

COMPUTERS, COMMUNICATIONS & CONTROL

With Emphasis on the Integration of Three Technologies

IJCCC
A Quarterly Journal

Year: 2009 Volume: IV Number: 1 (March)

Agora University Editing House

CCC Publications

Licensed partner: EBSCO Publishing

www.journal.univagora.ro

Publisher & Editorial Office
CCC Publications, Agora University
Piata Tineretului 8, Oradea, jud. Bihor, Romania, Zip Code 410526
Tel: +40 259 427 398, Fax: +40 259 434 925, E-mail: ccc@univagora.ro
Website: www.journal.univagora.ro
ISSN 1841-9836, E-ISSN 1841-9844

International Journal of Computers, Communications and Control (IJCCC) is published from 2006 and has 4 issues/year (March, June, September, December), print & online.

Founders of IJCCC: I. Dziţac, F.G. Filip and M.J. Manolescu (2006)

This publication is subsidized by:

1. Agora University
2. The Romanian Ministry of Education and Research / The National Authority for Scientific Research

CCC Publications, powered by Agora University Publishing House, currently publishes the "International Journal of Computers, Communications & Control" and its scope is to publish scientific literature (journals, books, monographs and conference proceedings) in the field of Computers, Communications and Control.

IJCCC is indexed and abstracted in a number of databases and services including:

1. ISI Thomson Reuters - Science Citation Index Expanded (also known as SciSearch®)
2. ISI Thomson Reuters - Journal Citation Reports/Science Edition
3. ISI Thomson Reuters - Master Journal List
4. SCOPUS
5. Computer & Applied Science Complete (EBSCO)
6. Vocational Studies Complete (EBSCO)
7. Current Abstracts (EBSCO)
8. Collection of Computer Science Bibliographies(CCSB)
9. Informatics portal io-port.net (FIZ KARLSRUHE)
10. MathSciNet
11. Open J-Gate
12. Google Scholar
13. Romanian Journals CNCSIS (cat.A, cod 849)
14. Information Systems Journals (ISJ)
15. Urlich's Periodicals Directory
16. Genamics JournalSeek
17. ISJ- Journal Popularity
18. Magistri et Scholares
19. SCIRUS
20. DOAJ

Contents

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 6-16

Incremental Improvement of the Evaluation Algorithm in the Concept Map Based Knowledge Assessment System

Alla Anohina, Marks Vilkelis, Romans Lukasenko

Riga Technical University, Department of Systems Theory and Design
Kalku street 1, Riga, Latvia, LV-1658
E-mail: alla.anohina@rtu.lv, markvilkel@inbox.lv, lrexpress@inbox.lv

Abstract: The paper is devoted to the knowledge assessment system that has been developed at the Department of Systems Theory and Design of Riga Technical University for the last four years. The system is based on concept maps that allow displaying the knowledge structure of a particular learner in the form of a graph. Teacher's created concept maps serve as a standard against which learner's concept maps are compared. However, it is not correct to compare teacher's and learners' concept maps by examining the exact equivalence of relationships in both maps, because people construct knowledge in different ways. Thus, an appropriate mechanism is needed for the flexible evaluation of learners' concept maps. The paper describes the algorithm implemented in the concept map based knowledge assessment system and its evolution through four prototypes of the system.
Keywords: knowledge assessment system, concept maps, evaluation algorithm

1 Introduction

Rapid development of information and communication technologies and availability of huge amount of electronic information resources has led to changes in the roles of the main actors of the educational process, namely, a teacher and a learner. Nowadays teachers should guide learners through the learning process by advising them and providing necessary stimuli while learners search information in different places, inter alia in e-learning environments (for example, [1, 2]) and turn it into knowledge. Special methods are needed for the evaluation of learners' knowledge structures created in such conditions and one of them is concept maps that allow displaying the knowledge structure of a particular learner in the form of a graph.

The Department of Systems Theory and Design of the Faculty of Computer Science and Information Technology of Riga Technical University has been developing the concept map based knowledge assessment system since the year 2005. The system has twofold goals in the context of the integration of technology into the traditional educational process: 1) to promote learners' knowledge self-assessment, and 2) to support the teacher in the improvement of the learning course through systematic assessment of learners' knowledge and analysis of its results. The goals are reached by the use of concept maps as an assessment tool. Three prototypes of the system have been already implemented and the fourth one is under development at the moment [3]. The paper is a logical continuation of the description of the system's working principles presented in [4] by focusing on a different aspect of its functionality, namely, on the algorithm for the comparison of learner's and teacher's concept maps and its incremental improvement through system's prototypes.

The paper is organized as follows. Section 2 gives an overview of the system. Section 3 discusses related works concerning scoring systems of concept maps. Evolution of the algorithm for the comparison of learner's and teacher's concept maps is described in Section 4. Finally, conclusions are presented.

2 Overview of the system

As it was mentioned in Introduction concept maps are used as an assessment tool in the developed knowledge assessment system. According to [5] they can foster the learning of well-integrated structural knowledge as opposed to the memorization of fragmentary, unintegrated facts and externalize the conceptual knowledge (both correct and erroneous) that learners hold in a knowledge domain. A concept map is a graph with labeled nodes corresponding to concepts in a problem domain and with arcs indicating relationships between pairs of concepts. Arcs can be directed or undirected and with or without linking phrases on them. A linking phrase specifies the kind of a relationship between concepts. Concept map based tasks can be divided in 1) "fill-in-the map" tasks, where the structure of a concept map is given to the learner and he/she must fill it using the provided set of concepts and/or linking phrases, and 2) "construct-a-map" tasks, where the learner must decide on the structure of the concept map by him/herself.

The developed concept map based knowledge assessment system is used in the following way. The teacher defines stages of knowledge assessment and creates concept maps for all of them by specifying relevant concepts and relationships among them in such a way that the concept map of each stage is nothing else then an extension of the previous one. Thus, the concept map of the last stage includes all concepts and relationships among them. Teacher's created concept maps serve as a standard against which the learners' concept maps are compared. During knowledge assessment the learner solves a concept map based task corresponding to the assessment stage. After the learner has submitted his/her solution, the system compares the concept maps of the learner and the teacher, calculates the score of the learner's result and generates feedback.

The system offers five concept map based tasks that are ranged from the easiest to the most difficult (Figure 1) [6]. Eight transitions between tasks are implemented allowing the learner to find a task most suitable for his/her knowledge level. Four transitions increase the degree of task difficulty. They are carried out after the analysis of the learner's solution if the learner has reached the teacher's specified number of points in the current assessment stage without reducing the degree of difficulty of the original task. So, this is a system's adaptive reaction to the learner's behavior. Other four transitions reduce the degree of task difficulty and they are carried out by the voluntary request from the learner during the solving of the task.

Thus, the system supports knowledge self-assessment as it makes the analysis and evaluation of learners' concept maps, as well as provides feedback about the learner's errors. It promotes systematic knowledge assessment because it allows the extension of the initially created concept map for other assessment stages. Moreover, statistical information about differences between learners' concept maps and teacher's concept map is collected providing opportunities for the teacher to improve the learning course [4].

The system is implemented as a Web-based three-tier client-server application [4, 7] consisting of the following architectural layers (Figure 2): 1) a data storage layer represented by Data Base Management System (DBMS); 2) an application logics layer composed of the application server and the server side code running on it; a special persistence and query framework is used to communicate with the DBMS; and 3) the representation layer or graphical user interface.

As it is shown in Figure 2 the concept map based knowledge assessment system can be divided into three logical domains: administrator, teacher and student. Each domain has its own goal, but they are strictly linked together. Functionality of each domain can be used by one of three user roles which names correspond to the names of the domains. An administrator is responsible for the administration and maintenance of the whole system using such functions as input, editing and deleting of data about users (teachers and students), courses and student groups. The teacher domain provides all necessary functions for the creation of concept maps and defining of their attributes, as well as for the viewing of learners' results. Functionality of the student domain includes all things related to the completion of the

concept map based tasks by learners and providing of feedback after the completion of the task.

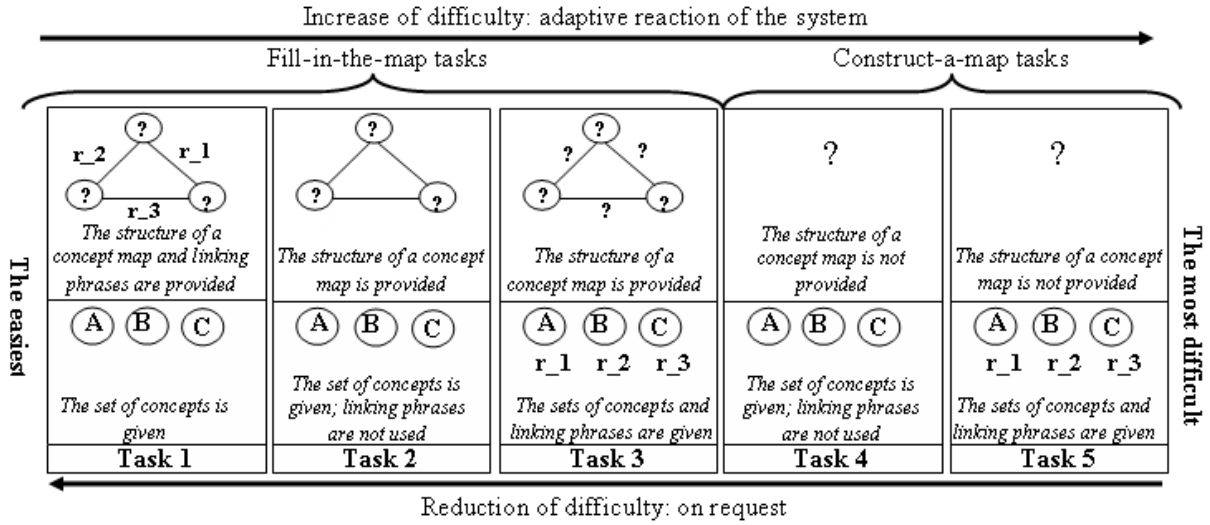


Figure 1: Tasks offered in the concept map based knowledge assessment system

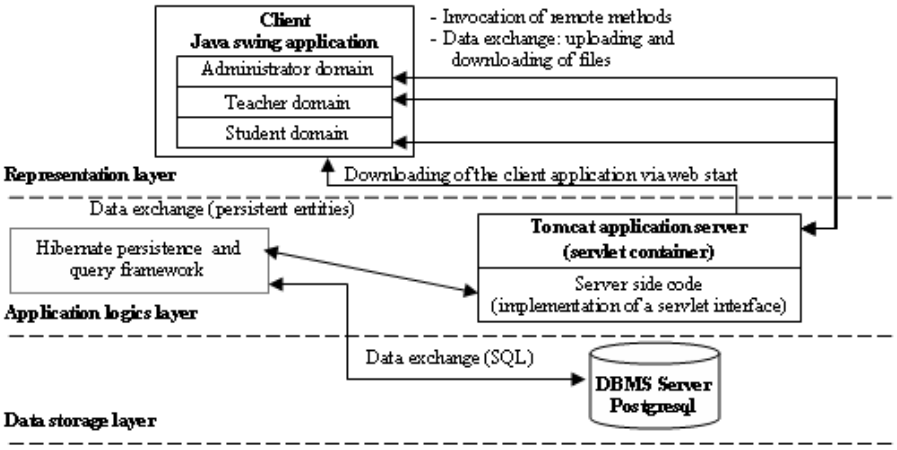


Figure 2: The three-tier architecture of the system

3 Related works

There are a lot of research related to scoring systems of concept maps. In [14] three main approaches are highlighted: evaluation of components of a concept map, comparison with the expert concept map, or combination of both previously mentioned kinds. Novak and Gowin [11] have offered a scheme in which different number of points are assigned to such components of a concept map as propositions, levels of hierarchy, cross-links, specific examples of concepts. In [13] the previously described set of components is extended by number of branchings in the concept map.

For the comparison of the learner's concept map with the expert map Goldsmith [8] has offered "the closeness index" indicating the degree of similarity between the expert and the learner's concept map. In turn, Herl and colleagues [9] use a matching algorithm that includes several expert maps evaluating each learner's concept map. In their work two evaluation indicators are calculated: (a) stringent semantic content score on the basis of the exact relationships matches between learner's and expert's concept

maps, and b) categorized semantic content score, when learner's defined relationship matches some set of possible relationships in the expert concept map. In [10] concept maps are scored by comparing each learner's concept map with two expert maps. The learner receives half a point for each relationship that matches with an expert relationship and a full point, if the relationship matches with both experts' relationships. Learners also receive additional points for relationships that are more valued by experts ("critical relationships") and relationships that are less like what experts might include.

Both the evaluation of components of concept maps and the comparison with the expert concept map are combined in [12], where three different approaches to the scoring of concept maps are used: (a) a total score, defined as a number of valid learner's relationships, (b) a congruence score, defined as a proportion of valid learner's relationships to all relationships in the expert map, (c) a salience score, defined as a proportion of valid learner's relationships to all relationships in the learner's concept map.

Regardless of the diversity of schemes for scoring of concept maps, it is necessary to note that basically they are developed for the evaluation of tasks, in which learners must create their own concept maps. Thus, a question about the evaluation of fill-in-the-map tasks remains open. Moreover, the greater part of the offered schemes are not considered in the context of computer-assisted assessment systems, thus, it is very difficult to evaluate, whether they are feasible and useful in the concept map based knowledge assessment system. These factors motivated the authors of the paper to develop an algorithm suitable for the evaluation of both fill-in-the-map tasks and construct-a-map tasks in the concept map based knowledge assessment system.

4 Evaluation algorithm

The developed algorithm is sensitive to the arrangement and coherence of concepts in the learners' concept maps. It is based on the assumption that the learner's understanding of the presence of a relationship between concepts has the primary value, while other aspects such as the type of the relationship, the linking phrase, the direction of the arc and the places of concepts are secondary things. The algorithm is capable of recognizing different patterns of the learner's solution. Two kinds of relationships are used in concept maps: 1) important relationships which show that relationships between the corresponding concepts are considered as important knowledge in the learning course, and 2) less important relationships that specify desirable knowledge. For each correctly provided important relationship the learner receives 5 points, but for each less important relationship only 2 points are assigned.

In the first system's prototype developed in 2005 only one fill-in-the-map task was offered to learners. Thus, all learners received the same structure of a concept map and a list of concepts. Learners must insert provided concepts in correct nodes of the concept map structure. Arcs were undirected and did not have semantics. Taking into account that the value of a completely correct relationship is 100 %, the following contributions of its constituent parts were defined: the presence of the relationship in the learner's concept map - 50% (a fact that the learner understands the presence of the relationship between concepts has the primary value), the correct type of the relationship - 30% (the learner should be able to distinguish, what is important and what is less important in the learning course), both concepts related by the relationship are placed in the correct places - 20% (this factor has the greatest subjectivity).

Thus, the patterns of the learner's solution which the algorithm was capable of recognizing are the following [15, 16]:

- Pattern 1. The learner has interrelated concepts in the same way as they are related in the teacher's concept map. In this case the learner receives the maximum score regarding the type of the relationship. Figure 3.b. shows that the concepts A and E in the learner's concept map are interrelated in the same way as they are interrelated in the teacher's concept map (Figure 3.a).
- Pattern 2. The learner has defined a relationship that does not exist in the concept map of the teacher. In this case he/she does not receive any points. Figure 3.c demonstrates that the learner

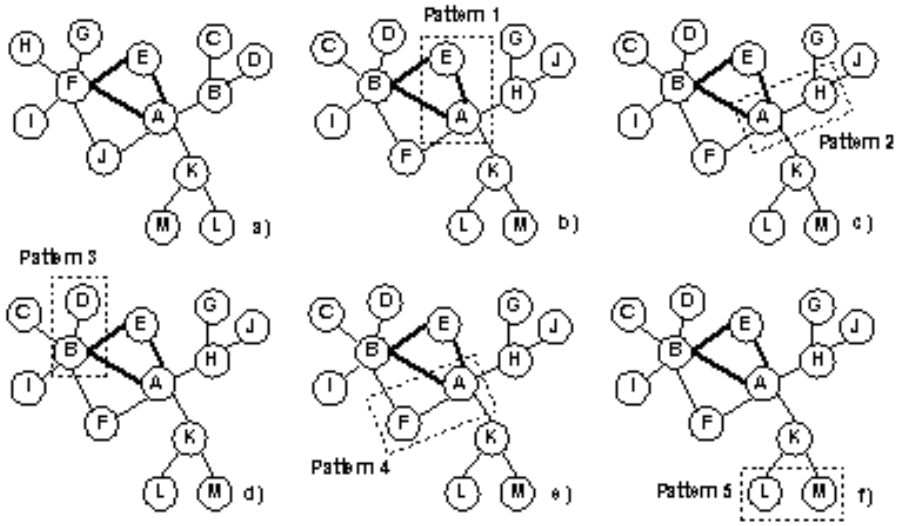


Figure 3: Patterns of the learner's solution that the first system's prototype was able to recognize: a) the teacher's concept map; b)-f) the patterns in the learner created concept map

has made a relationship between the concepts A and H that are not interrelated in the teacher's concept map (Figure 3.a).

- Pattern 3. The learner's defined relationship exists in the teacher's map, the type of the relationship is correct, but at least one of the concepts is placed in an incorrect place. The learner receives 80% of the maximum score for the correct relationship of this type. Figure 3.d shows that the learner has interrelated the concepts B and D. A similar relationship exists in the teacher's concept map (Figure 3.a). However, both concepts are located in incorrect places, although the type of the relationship is correct.

- Pattern 4. The learner's defined relationship exists in the teacher's map, the type of the relationship is wrong, and at least one of the concepts is placed in an incorrect place. The learner receives 50% of the maximum score for the correct relationship of this type. This pattern is displayed in Figure 3.e. Comparing the learner's defined relationship between the concepts A and F with the same teacher's relationship (Figure 3.a) it is possible to notice that the concept F is located in the incorrect place, as well as the type of the relationship is a less important relationship instead of an important relationship.

- Pattern 5. A concept is placed in a wrong place, but its place is not important. The learner receives the maximum score for a corresponding relationship. Figure 3.f demonstrates that the learner has changed places of the concepts M and L.

New tasks - 3 fill-in-the-map and 2 construct-a-map tasks (Figure 1)- were introduced in the second system's prototype developed in 2006. Moreover, linking phrases were added to concept maps causing modification of points received for relationships and extension of the set of patterns recognized by the system. Assuming that a value of the fully correct relationship is 100%, the following contributions of its constituent parts were defined: the presence of the relationship in the learner's concept map - 40%, a correct linking phrase provided for the relationship - 30% (semantics of relationships are important knowledge units), a correct type of the relationship - 20%, both concepts related by the relationship are placed in the correct places - 10%.

Therefore, the extended set of patterns of the learner's solution was the following [6]:

- Pattern 1. The learner has defined a completely correct relationship. In this case the learner receives the maximum score regarding the type of the relationship. Figure 4.b demonstrates that the concepts A and E in the learner's concept map are interrelated in the same way as they are interrelated in the teacher's concept map (Figure 4.a).

- Pattern 2. The learner has defined a relationship that does not exist in the concept map of the teacher and he/she does not receive any points. Figure 4.c shows that the learner has related the concepts A and H which are not related in the teacher's map (Figure 4.a).

- Pattern 3. The learner's defined relationship exists in the teacher's map, both the type of the relationship and the linking phrase are correct, but at least one of the concepts is located in an incorrect place. The learner receives 90% of the maximum score for that relationship. This pattern is displayed in Figure 4.d. The learner has defined the relationship between the concepts B and D by providing the correct type of the relationship and the linking phrase. However, both concepts are located in different places as compared to the teacher's concept map (Figure 4.a).

- Pattern 4. The learner's defined relationship exists in the teacher's map, but the type of the relationship is incorrect. The learner receives 80% of the maximum score for the correct relationship. This pattern is valid only for construct-a-map tasks (Tasks 4 and 5 in Figure 1) where places of concepts are not important. Assuming that Figure 4.e displays the learner's created concept map, one can see that the learner has interrelated the concepts B and C by defining the correct linking phrase, but providing the important relationship instead of a less important one as it is in the teacher's map (Figure 4.a).

- Pattern 5. The learner's defined relationship exists in the teacher's map, but the linking phrase is incorrect. The learner receives 70% of the maximum score for the correct relationship. Figure 4.f shows that the learner has located the concepts A and K in the same places as they are located in the teacher's concept map (Figure 4.a), but the linking phrase differs.

- Pattern 6. The learner's defined relationship exists in the teacher's map, the type of the relationship is incorrect, and at least one of concepts is placed in the incorrect place. The learner receives 70% of the maximum score. Figure 4.g shows that the learner has related the concepts A and F using the less important relationship instead of the important relationship (Figure 4.a) and has located the concept F in other place.

- Pattern 7. The learner's defined relationship exists in the teacher's map, the linking phrase is incorrect, and at least one of concepts is located in the incorrect place. The learner receives 60% of the maximum score. This pattern is displayed in Figure 4.h, where one can see that the learner has defined the relationship between the concepts G and F, but has pointed out the linking phrase that differs from the linking phrase in the teacher's concept map (Figure 4.a), as well as has located both concepts incorrectly.

- Pattern 8. The learner's defined relationship exists in the teacher's map, but both the type of the relationship and the linking phrase are incorrect. The learner receives 50% of the maximum score for the correct relationship. This pattern is valid only for construct-a-map tasks (Tasks 4 and 5 in Figure 1), where places of concepts are not important. Assuming that Figure 4.i displays the learner's created concept map, one can see that the learner has interrelated the concepts O and B that are also interrelated in the teacher's concept map (Figure 4.a). The learner has provided an incorrect type of the relationship (an important relationship instead of a less important relationship) and an incorrect linking phrase.

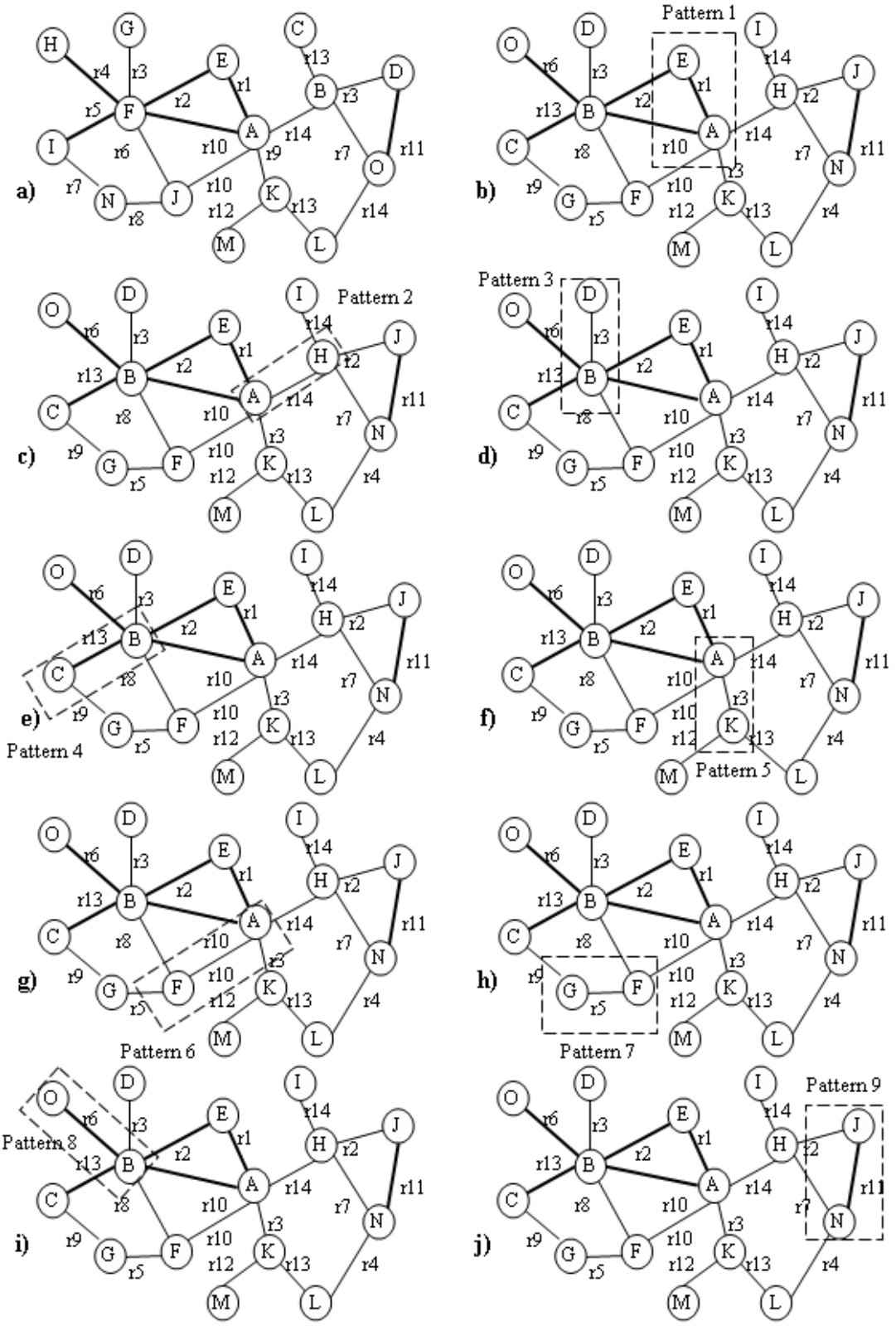


Figure 4: Patterns of the learner's solution that the second system's prototype was able to recognize: a) the teacher's concept map; b)-j) the patterns in the learner created concept map (numbers on links display different relationships and represent linking phrases)

- Pattern 9. The learner's defined relationship exists in the teacher's map, both the type of the relationship and the linking phrase are incorrect, in addition at least one of the concepts is placed in the incorrect place. The learner receives 40% of the maximum score for that relationship. This pattern is shown in Figure 4.j. By comparing the learner's defined relationship between the concepts J and N with the same relationship in the teacher's concept map (Figure 4.a), one can see that both concepts are located in incorrect places, and the type of the relationship and the linking phrase are incorrect.

In the third prototype implemented in 2007 directed arcs were introduced causing the following modifications of points received for relationships: the presence of the relationship in the learner's concept map - 40%, a correct linking phrase provided for the relationship - 30%, a correct direction of the arc corresponding to the relationship - 15%, a correct type of the relationship - 10%, both concepts related with the relationship are placed in the correct places - 10%. Totally 36 patterns were acquired for all tasks and they are summarized in Figure 5.

No modifications of points have been made in the fourth system's prototype. Thus, the evaluation algorithm is based on the set of patterns specified in Figure 5. However, improvement of the algorithm is continued by studying possibility to reveal extra relationships in learners' concept maps [17]. Figure 6 displays a situation when some relationships are "hidden": there are only 3 relationships in the teacher's concept map (Figure 6.a), but 2 more relationships can be derived (Figure 6.b). These derived relationships are correct too and could appear in learners' concept maps, so it is necessary to define a mechanism according to which the system could detect extra relationships and, thus, make assessment more flexible and automated.

With aim to determine extra relationships 6 types of relationships are considered [17]:is a - a relationship between concepts meaning that one of the concepts is a sub-class of other, part of - a relationship between concepts meaning that one of the concepts is a part of other, attribute - a relationship between a concept and its attribute, example - a relationship between a general concept and a particular example of it, value - a relationship between an attribute and its value, kind of - a relationship between levels of hierarchy.

Figure 7 shows the structure of a pattern further explained in Figure 8. The pattern has two main relationships (Relation 1 and Relation 2) that have types mentioned previously. In some cases combination between relationships is not allowed. In other cases an extra relationship (Relation 3) can be formed. Column "Combination allowed" identifies either combination between Relation 1 and Relation 2 is allowed or not. The entry "Cannot be specified" in the column "Relation 3" indicates that there can be situations when an extra relationship can be added, but not always. In turn, the entry "No extra relationship" in the same column points out that no additional relationship of considered 6 types can be added.

Additional relationships that can be derived from more than three concepts and two relationships between them can be revealed as well. In such case the algorithm must iteratively go through the concept map searching for patterns and adding extra relationships whenever it is possible. The algorithm stops when no new relationship has been added during the last iteration.

5 Conclusions

The use of concept maps for the evaluation of learners' knowledge structures demands an appropriate mechanism for the comparison of learners' concept map with the teacher's one. The mechanism should consider that people construct knowledge in different ways and there can be differences between concept maps that are not indicators of incorrect knowledge. The paper presents the algorithm capable of recognizing several patterns of the learner's solution on the basis of such criteria as location of concepts, types of relationships, direction of arcs and correctness of linking phrases allowing the learner to receive

The presence of the relationship	Location of concepts in the correct places	Type of the relationship	Direction of the arc	Linking phrase	Points	Comments
Exists	Both places are correct or places are not important (Tasks 4 and 5 in Figure 1)	Correct	Correct	Correct or not used (Tasks 2 and 4 in Figure 1)	Maximum score	A completely correct relationship
	Both concepts are located in correct places	Correct	Correct	Incorrect	0.7 * Maximum score	Applicable to Task 3 in Figure 1
	At least one of concepts is located in the incorrect place	Correct	Correct	Correct or not used (Task 2 in Figure 1)	0.95 * Maximum score	Applicable to Tasks 1, 2 and 3 in Figure 1
		Correct	Correct	Incorrect	0.65 * Maximum score	Applicable to Tasks 1 and 3 in Figure 1
		Correct	Incorrect	Correct or not used (Task 2 in Figure 1)	0.8 * Maximum score	Applicable to Tasks 1, 2 and 3 in Figure 1
		Incorrect	Correct	Correct or not used (Task 2 in Figure 1)	0.85 * Maximum score	Applicable to Tasks 1, 2 and 3 in Figure 1
		Incorrect	Correct	Incorrect	0.55 * Maximum score	Applicable to Tasks 1 and 3 in Figure 1
		Correct	Incorrect	Incorrect	0.5 * Maximum score	Applicable to Tasks 1 and 3 in Figure 1
		Incorrect	Incorrect	Correct or not used (Task 2 in Figure 1)	0.7 * Maximum score	Applicable to Tasks 1, 2 and 3 in Figure 1
		Incorrect	Incorrect	Incorrect	0.4 * Maximum score	Applicable to Tasks 1 and 3 in Figure 1
	Places of concepts are not important	Correct	Incorrect	Correct or not used (Task 4 in Figure 1)	0.85 * Maximum score	Applicable to Tasks 4 and 5 in Figure 1
		Incorrect	Correct	Correct or not used (Task 4 in Figure 1)	0.9 * Maximum score	Applicable to Tasks 4 and 5 in Figure 1
		Incorrect	Incorrect	Correct or not used (Task 4 in Figure 1)	0.75 * Maximum score	Applicable to Tasks 4 and 5 in Figure 1
		Correct	Correct	Incorrect	0.7 * Maximum score	Applicable to Task 5 in Figure 1
		Correct	Incorrect	Incorrect	0.55 * Maximum score	Applicable to Task 5 in Figure 1
		Incorrect	Correct	Incorrect	0.6 * Maximum score	Applicable to Task 5 in Figure 1
		Incorrect	Incorrect	Incorrect	0.45 * Maximum score	Applicable to Task 5 in Figure 1
Does not exist					0 points	A completely incorrect relationship

Figure 5: The extended set of patterns of the learner's solution in the third system's prototype

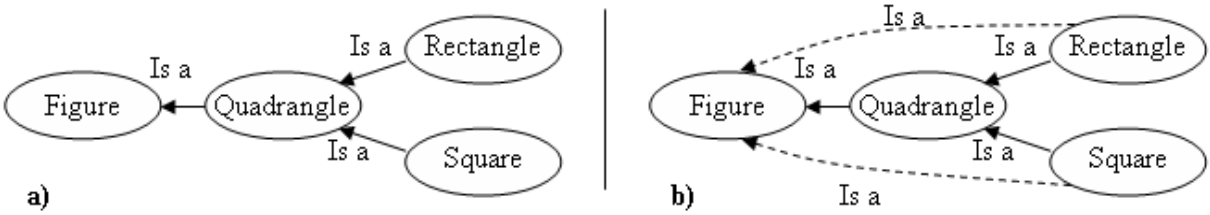


Figure 6: Hidden relationships (adopted from [17])

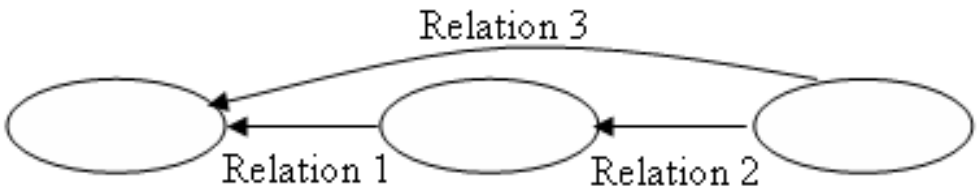


Figure 7: Structure of the pattern (adopted from [17])

some points in case if his/her relationships match teacher's ones only partly. Moreover, detection of extra relationships that can appear in learners' concept maps makes assessment more automated.

Relation 1	Relation 2	Combination allowed	Relation 3
Is a	Is a	Yes	Is a
Is a	Part of	Yes	Part of
Is a	Attribute	Yes	Cannot be specified
Is a	Example	Yes	Is a
Is a	Value	No	–
Is a	Kind of	Yes	Is a
Part of	Is a	Yes	Part of
Part of	Part of	Yes	Part of
Part of	Attribute	Yes	Cannot be specified
Part of	Example	Yes	Part of
Part of	Value	No	–
Part of	Kind of	Yes	Part of
Attribute	Value	Yes	No extra relationship
Attribute	Any other except "Value" and linguistic	No	–
Example	Is a	Yes	No extra relationship
Example	Part of	Yes	No extra relationship
Example	Attribute	Yes	Cannot be specified
Example	Example	No	–
Example	Value	No	–
Example	Kind of	Yes	No extra relationship
Value	Any other except linguistic	No	–
Kind of	Part of	Yes	Part of
Kind of	Is a	Yes	Is a
Kind of	Kind of	Yes	Is a
Kind of	Example	Yes	Example
Kind of	Attribute	Yes	Cannot be specified
Kind of	Value	No	–

Figure 8: Patterns containing three concepts and two relationships (adopted from [17])

Bibliography

[1] A.Styliadis, I.Karamitsos, D.Zachariou,Personalized e-Learning Implementation-the GIS Case, *International Journal of Computers,Communications & Control*, Vol.1, No.1, pp.59-67, 2006.

[2] G. Moise, A Software System for Online Learning Applied in the Field of Computer Science, *International Journal of Computers,Communications & Control*, Vol.2, No.1, pp.84-93, 2007.

[3] J. Grundspenkis, A. Anohina, Evolution of the Concept Map Based Adaptive Knowledge Assessment System: Implementation and Evaluation Results, *Scientific Proceedings of Riga Technical University*, RTU Publishing, Riga, 2008 (submitted).

[4] M. Vilkelis, A. Anohina, R. Lukashenko, Architecture and Working Principles of the Concept Map Based Knowledge Assessment System, *Proceedings of the 3rd International Conference on Virtual Learning*, October 31 - November 2, Constanta, Romania, pp. 81-90, 2008.

[5] A. J. Cañas, *A Summary of Literature Pertaining to the Use of Concept Mapping Techniques and Technologies for Education and Performance Support*, Technical report, 2003.

[6] A. Anohina, D. Pozdnakovs, J. Grundspenkis, Changing the Degree of Task Difficulty in Concept Map Based Assessment System, *Proceedings of the IADIS International Conference "e-Learning 2007"*, July 6-8, Lisbon, Portugal, pp. 443-450, 2007.

[7] R. Lukashenko, M. Vilkelis, A. Anohina, Deciding on the Architecture of the Concept Map Based Knowledge Assessment System,*Proceedings of the International Conference on Computer Systems and Technologies*, June 12-13, Gabrovo, Bulgaria, pp. V.3-1 - V.3-6, 2008.

[8] T. E. Goldsmith, P. J. Johnson, W. H. Acton, Assessing Structural Knowledge, *Journal of Educational Psychology*, Vol.83, No.1, pp. 88-96, 1991.

[9] H. Herl, E. Baker, D. Niemi, Construct Validation of an Approach to Modelling Cognitive Structure of U.S. History Knowledge, *Journal of Educational Research*, No. 89, pp. 206-219, 1996.

[10] D. C. D. Klein, G. K. W. K. Chung, E. Osmundson, et al., *Examining the Validity of Knowledge Mapping as a Measure of Elementary Students' Scientific Understanding*, CSE techical report No.557, 2002.

[11] J. D. Novak, D. B. Gowin, *Learning How to Learn*, London, 1984.

[12] M. A. Ruiz-Primo, R. J. Shavelson, S. E. Schultz, *On the Validity of Concept Map-Based Assessment Interpretations: an Experiment Testing the Assumption of Hierarchical Concept Maps in Science*, CSE Technical Report 455, 1997.

[13] J. D. Wallace, J. J. Mintzes, The Concept Map as a Research Tool: Exploring Conceptual Change in Biology, *Journal of Research in Science Teaching*, Vol. 27, pp. 1033-1052, 1990.

[14] M. A. Ruiz-Primo, R. J. Schavelson, Problems and Issues in the Use of Concept Maps in Science Assessment, *Journal of Research in Science Teaching*, No.6, pp. 569-600, 1996.

[15] A. Anohina, J. Grundspenkis, Prototype of Multiagent Knowledge Assessment System for Support of Process Oriented Learning, *Proceedings of the 7th International Baltic Conference on Databases and Information Systems*, July 3-6, Vilnius, Lithuania, pp. 211-219, 2006.

[16] A. Anohina, V. Graudina, J. Grundspenkis, Using Concept Maps in Adaptive Knowledge Assessment, *Advances in Information Systems Development "New Methods and Practice for the Networked Society"*, Springer, pp. 469-480, 2006.

[17] J. Grundspenkis, M. Strautmane, Usage of Graph Patterns for Knowledge Assessment Based on Concept Maps. *Scientific Proceedings of Riga Technical University*, RTU Publishing, Riga, 2008 (submitted).

Alla Anohina is a lecturer at the Department of Systems Theory and Design of Riga Technical University in Latvia. She obtained her doctoral degree in Computer Systems from Riga Technical University in 2007. Her main research fields are intelligent tutoring systems, computer-assisted assessment systems and artificial intelligence. She has seven years' experience of teaching in the field of computer science both in Riga Technical University, and in other educational institutions of Latvia. She has participated in several research projects related to the development of of educational software.

Marks Vilkelis is currently developing the master thesis "The knowledge assessment system as a client-server java application" at the Institute of Information Technology of Riga Technical University. In 2005 he obtained the bachelor degree from the mentioned institution. His scientific interests cover Java open source technologies and applications of artificial intelligence. He has experience of working as a programmer in some research projects. His professional activities are related to the programming of complex business applications in the international software development company.

Romans Lukasenko is a PhD student of Computer Systems at the Riga Technical University in Latvia. He obtained the master degree in Computer Systems from the mentioned institution in 2006. His main research fields are intelligent tutoring systems, computer-assisted assessment systems and student modelling. His thesis is related to the implementation of a student model in the knowledge assessment system. Romans Lukashenko works also as a researcher at the Department of Systems Theory and Design of Riga Technical University. He has participated in several research projects related to the development of educational software.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 17-26

Artificial Intelligence + Distributed Systems = Agents

Ioan Dzitac, Boldur E. Bărbat

Ioan Dzitac
"Aurel Vlaicu" University of Arad, Faculty of Exact Sciences, Department of Mathematics-Informatics
Str. Elena Dragoi, Nr. 2, Complex Universitar M (Micalaca, zona III), Arad, Romania
E-mail: idzitac@gmail.com

Boldur E. Bărbat
"Lucian Blaga" University of Sibiu, "Hermann Oberth" Faculty of Engineering, Department of Research
10, Victoriei Bd, Sibiu, 550024, România
E-mail: bbarbat@gmail.com

Every established religion was once a heresy
Henry Buckle - "Essays"

Abstract: The connection with Wirth's book goes beyond the title, albeit confining the area to modern Artificial Intelligence (AI). Whereas thirty years ago, to devise effective programs, it became necessary to enhance the classical algorithmic framework with approaches applied to limited and focused subdomains, in the context of broad-band technology and semantic web, applications - running in open, heterogeneous, dynamic and uncertain environments-current paradigms are not enough, because of the shift from programs to processes. Beside the structure as position paper, to give more weight to some basic assertions, results of recent research are abridged and commented upon in line with new paradigms. Among the conclusions: a) Non-deterministic software is unavoidable; its development entails not just new design principles but new computing paradigms. b) Agent-oriented systems, to be effectual, should merge conventional agent design with approaches employed in advanced distributed systems (where parallelism is intrinsic to the problem, not just a mean to speed up).
Keywords: open, heterogeneous, dynamic and uncertain environments (OHDUE); computer-aided decision-making; nonalgorithmic software; bounded rationality; agent-oriented software engineering (AOSE).

1 Introduction. Adapting to the Time(s) of Change

Three dominant features of the post-industrial society relevant here are:

a) *Growing speed of change* (due to the intense positive feedback entailed by Moore's law outcomes: Internet, broad-band technology, semantic Web, Google, [9] etc.);

b) *Growing complexity* (architectural, cognitive, structural [4, 5, 10]).

c) *Globalization* (expressed in IT context mainly through the modern enterprise paradigms). Berners-Lee - who coined also the term *GGG* (Giant Global Graph) to describe the semantic Web as a new stage of *WWW* - utters it very pointed: "I have gone from using a 300 board connection on one of those telephone couplers to a 3 million board connection, so that is a 10,000 factor. So the technology underneath this has tremendously increased in terms of speed and functionality, and the Web technology had happened on top" [http://dig.csail.mit.edu/, 2007].

Thus, modern IT environments, except for simple applications, move towards open and *heterogeneous* (resources are unalike and their availability is not warranted), *dynamic* (the pace of exogenous and endogenous changes is high) and *uncertain* (both information and its processing rules are revisable, fuzzy, and uncertain). Most situations to be controlled are complex and uncertain, and involve parallel processes. Thus, the applications developed to deal with must be *intelligent* [7] (to manage complexity and uncertainty) AND *distributed* (to handle parallelism) are intrinsically non-deterministic and end-users have to interact with them in manners they are not get used to. Moreover, a second (side-effect) vicious circle comes out from the interaction between difficulty to adapt and the follow-on frustration (for instance, the claims regarding "digital manipulation").

In brief, adapting to the speed of change is mandatory and the target of this paper is to show that it entails adopting modern IT paradigms. The approach is based on the homomorphism of the addition in the paper title to that in the famous book of Wirth [22].

Accordingly, Section 2 presents the *rationale* analysing and supporting the evolution *from programs to agents*, emphasizing the temporal dimension. Deepening the investigation, Section 3 explores the unavoidable *paradigm shift* (here, many and diverse paradigms are x-rayed). Section 4 includes the core: the misleading term "*distributed*" is thoroughly *revisited* because distribution was mostly part of the *solution* and is now recognized as main part of the problem too. Just to give more weight to some assertions in preceding *sections, models, methods,* and *mechanisms* based on the paradigms endorsed before, are abridged after earlier papers and commented upon in Section 5. *Conclusions* and *intentions* for future development close the paper (Section 6).

2 Rationale. From Programs to Agents

Like all changes induced by systems with intense positive feedback, the shift from programs to agents is at once a long way (*conceptually*, since it involves multiple paradigm changes), a swift leap (*historically*, since - as IT beings - algorithms are active, whereas agents are teenagers) and a hard chaotic fight (*epistemically*, since AI as a whole is at the same time feared and ridiculed, overrated and denied). Misinterpretations are eased because - confusing program architecture with code structure - some professionals still consider that x-rayed programs are nothing more than implemented algorithms, since in binary form data and instructions are indiscernible.

Moreover, in the case of AI, the argument about the difference between programs and agents is complicated by two conceptual inflations due to overstated advertising:

a) to many programs are labelled as intelligent;

b) most such "intelligent" programs are renamed as agents (with or without minor reshaping). Thus, the notion of agent (and the very metaphor behind it) are blurred.

Object orientation - as almost one and only software engineering paradigm - adds two more hindrances to:

a) already in the 1990s, real-time programming showed that even the conceptual equivalence "*program = object*" exposed intellectual difficulties (e.g., objects like "mutex", "event", or "exception", state "start", method "execute");

b) almost as a corollary, by transitivity, it is commonsensical that the equivalence "*agent = object*" is awful, since nobody could be happy to be considered "intelligent like an object".

Besides, objects have - at most - a very primitive temporal dimension. Tough, this dimension is fundamental because no software entity lacking it could be able to (inter)act in the e-world with other

entities, neither artificial (its peers), nor natural (its human end-users). Tellingly enough, time entered the software universe via data structures like dynamic data (e.g., lists in *Pascal*) or data types for concurrency (from monitors in *Concurrent Pascal* to tasks in *Ada*). Likewise, in AI, a genuine temporal dimension was entailed by distributed systems (to handle the parallelism involved - albeit architectural inefficiency due to approaches based on "light" multithreading and time-sharing). Thus, the homomorphism suggested in the paper title is even deeper than supposed at first.

On the other hand - at least in countries similar to Romania - IT curricula are obviously lagging behind the state of the art. Aspects relevant for this paper are:

a) AI syllabi show a long-lasting flavour of "GOFAI" (*G*ood *O*ld-*F*ashioned *AI*).

b) Even when rigid planning-based AI is replaced by "BIC" (Biologically Inspired Computing), the focus is not on *modelling* - inspired from biologic (sub-symbolic) paradigms -, but merely on *simulating* biologic behaviour.

c) Software engineering syllabi are entirely object-oriented; thus, agents - if considered - are designed as objects (e.g., using JADE).

d) Distributed systems are approached in still more conventional manner: the only concern is to boost speed, not to reflect real-world parallelism.

Hence the rationale is threefold, depending on the perspective:

a) *AI* (intelligent software must be *process*-based, not *program*-based);

b) *Software engineering* (intelligent applications should be agent-oriented, not object-oriented);

c) *IT curricula* (AI and Distributed systems should be merged based on agent-orientation).

3 Paradigm Shift. What is Certain?

Here "paradigm" means: "thought pattern in any scientific discipline or other epistemological context. The Merriam-Webster dictionary defines broadly: a philosophical or theoretical framework of any kind. [...] Perhaps the greatest barrier to a paradigm shift, in some cases, is the reality of paradigm paralysis, the inability or refusal to see beyond the current models of thinking" [http://en.wikipedia.org/wiki/Paradigm#Paradigm_shifts].

A lot of fundamental scientific concepts - inside and outside IT - changed dramatically their intension since IT begun to be the dominant "Novum Organon" of post-industrial technological development. They have been investigated from agent-oriented perspective in [4, 5] and lately in [6] where the manifold paradigmatic shift they contributed to engender was labelled "From Kelvin to Zadeh" (Figure 1) because the focus was on *precision* (unnatural in real world, hence inadequate in software) and the shift described there referred to swap from the conventional "Computing with Numbers" (based mainly on measurements) to the modern "Computing with Words" (based mainly on perceptions).

In addition, Figure 1 tries to remind - or at least to suggest - that:

- *Moore's Law* epitomised as feedback loop - in the way usual in automation and electronics - emphasises not only the growing psychological difficulty to adapt to an unprecedented speed of change but also its known side effects: instability, distortion, complexity (mainly, cognitive), frustration (e.g., the irritation about "digital manipulation").

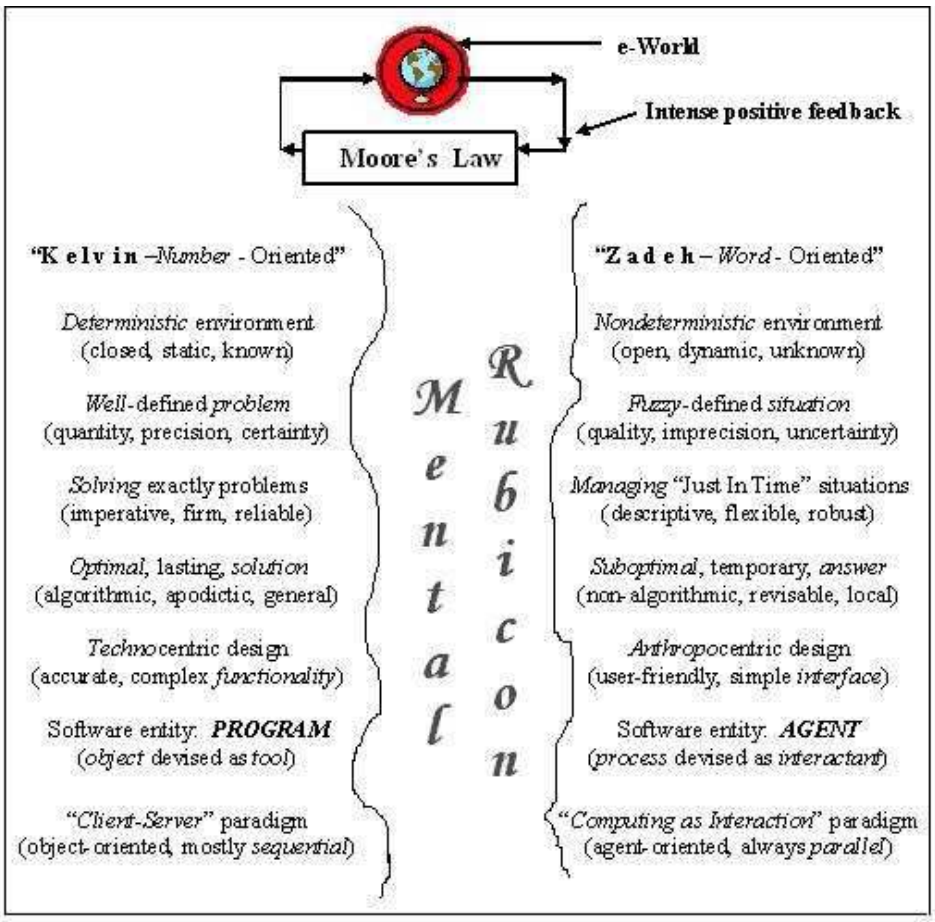


Figure 1: Manifold paradigm shift (adapted and extended from [6])

- "*Manifold*" as "Diverse": In line with common usage, "paradigm" regards also specific areas within a discipline (e.g., "programming paradigm", mainly when it "leverages a computer language's power" [21]). In Figure 1 the paradigmatic level lowers from a wide-ranging one to a narrow, specialised one usual in software engineering. Thus, the "Kelvin paradigm" suggests that IT must stay firmly based on mathematical precision, while the "Zadeh paradigm" replays that it should shift towards semiotic flexibility [23]. On the other hand, shifting from "Client-Server", to "Computing as Interaction" [1, 24]) is focused - not involving (im)precision, (non)determinism, (un)certainty, etc. (Besides, another dimension of reason diversity was recently shown: the ethnographical one [17].)

- "*Manifold*" as "*Many*": After redressing the balance (i.e., accepting all kinds of paradigms, not just GOFAI), AI was inundated by sub-symbolic paradigms; among the best known are artificial neural networks (ANN) and genetic algorithms (GA). The most nihilist and powerful, i.e. the ethological paradigm (based on the physical-grounding hypothesis [4]), seems to be still in vogue for developing agents (above all since the agent is accredited as *process* by a formal standard [12]).

- "*Manifold*" as "*Non-exhaustive*": to save space some important paradigm shifts having rather philosophical/epistemological nature are skipped over. Just one example: humans can communicate only among themselves or with machines too? Here, the problem could be circumvented replacing "*communication*" by "*interaction*" but the epistemological facet is still there: it has to be admitted (or rejected) that, in the framework of "computing as interaction", their connotations are

very similar. Perhaps, the definition given by Sieckenius de Souza could help: "*Communication* is the process through which, for a variety of purposes, sign producers (i.e., signification system users in this specific role) choose to express intended meanings by exploring the possibilities of existing signification systems or, occasionally, by resorting to non-systematized signs, which they invent or use in unpredicted ways" [19].

- Bounded Rationality. The term is used as defined, explained and endorsed in [20, 18, 14]. Since it is a very rich concept, often applied as principle, there are many "models of bounded rationality" [20] - including a Nobel Prize winner one ("psychology for behavioural economics" [16]). It entails that almost any human undertaking - to be effective as regards the time required - must be imperfect. In AI context, perfection suggests the ideal of mathematicians - and conventional software developers too - to achieve algorithm-based optimisation. Just one unquestionable example: "The problem is neither to admit that for any medical act (and for even stronger reasons as regards nursing) "just in time" is a sine qua non condition, nor that bounded rationality is the only practical means to achieve it [13]. Nevertheless, there is a double hindrance, due to a yet prevalent mentality:

  a) therapeutic decision-making is an exclusively human attribute;

  b) non-algorithmic software is - if not nonsensical - applicable at most to toy problems" [5].

  Hence, bounded rationality explodes in a fascicle of interrelated, versatile, and highly application-dependent features. Examples: learning or negotiation strategies; most features meant to reduce complexity (like the "zero-overhead rule" in generative programming).

- Time enters the picture threefold:

  a) From left to right, it represent the very essence of paradigm shift in the sense of Kuhn (for some conservatives unable to adapt the time to accomplish the shift can attain infinity; on the other hand, children have no problem to "communicate" with their artificial playmates).

  b) Explicitly, through the need to manage "Just In Time" rapidly changing situations.

  c) Implicitly, by handling real-world parallelism. Here, parallelism is intrinsic to any interaction (even between an interface agent and its owner), because interactants coexist in time. Of course, distribution adds new facets to an already complex temporal dimension.

In short, to be effectual in e-World both users and software developers have to pass the mental Rubicon separating *programs* from *agents*. Indeed, agents are here to stay (even if not yet very intelligent); "affective computing", "semantic web", "ambient intelligence" and so on, are more than slogans (they are recognized as main IT development directions by EU-promoted acts [1]).

Thus, *"What is certain?"* in the section title goes beyond the necessary enquiry of uncertainty, questioning the very essence of nowadays agent-oriented application development. For instance, even when blending vigorous well established paradigms, research trends are in line with the suggestions in the right part of Figure 1: "Interpretability is considered to be the main advantage of fuzzy systems over alternatives like neural networks, statistical models, etc. [...] In the recent years, research has started to focus on the trade-off between interpretability and accuracy [...]. Analysis of the model interpretability and comprehensibility is always convenient, and it is a necessity when accuracy is not a model feature" [15].

4 "Distributed" Revisited. A Solution becomes Problem

To distribute means "To divide and dispense in portions" implying a previous entirety able to be divided [http://www.thefreedictionary.com/Distributed]. Paradoxically, the epistemic trouble with dis-

tributed systems is that they are conceived, designed and implemented corresponding to the common connotation of "Distributed" [11, 8]. Thus, "In distributed computing a program is split up into parts that run simultaneously on multiple computers communicating over a network. [...] The main goal of a distributed computing system is to connect users and resources in a transparent, open, and scalable way. [...] Distributed computing implements a kind of concurrency. It interrelates tightly with concurrent programming so much that they are sometimes not taught as distinct subjects [...] If not planned properly, a distributed system can decrease the overall reliability" [http:// en.wikipedia.org/wiki/Distributed_computing].

"Distribution", as "the act of distributing or the condition of being distributed; apportionment" [http://www.thefreedictionary.com/distribution], is related to resource management (if something is plenty, no need to apportion).

In short - and maybe oversimplified - there are four kinds of circumstances where, within the IT treatment of the case, *distribution* - in its conventional meaning - can help (inside parentheses are examples) [4]:

a) *In space*. Spatial distribution is the oldest and most familiar type of resource *apportionment* (equipment components, process phases, networking, credit-card terminals).

b) *In time*. Time-sharing preceded its name (learning in schools, delegating authority, or reading on a ride are much older than UNIX-like operating systems or parallel buses).

c) *In organization*. Any organism is based on distributed order (human body, company, state, flower). For virtual enterprises it becomes a major raison d'étre.

d) *In problem solving*. "Divide et impera" was always a foremost strategy to fight complexity, chiefly cognitive one (most reductionist theories, most methodologies - from Euclid's algorithm to structured programming).

The difficulties begun when "distributed", and "parallel" were perceived as quasi-synonymic in the syntagm "distributed computing". Moreover, epistemic confusion escalates when other debatable (semantically antinomic) concepts generated even more doubtful pair of antonyms: "sequential"/"simultaneous" (instead of "parallel"), "holistic"/"analytical" (instead of "reductionist"). A relevant case is related to the basic metaphor of ANN: despite the same distributed neural network structure in both brain hemispheres, an antinomic pair of functions is stated as "linear algorithmic processing" vs. "holistical algorithmic processing" [http://en.wikipedia.org/wiki/Cerebral_hemispheres].

Here the reluctance to accept the paradigmatic shift that processing could be also non-algorithmic (even in the left brain) generated a quasi-pleonasm (could a step-by-step procedure be nonlinear?) and a quasi-contradiction (could a deterministic procedure be holistical?). As regards *processing*, the real opposition is "sequential"/"parallel", where parallelism involves distribution. For instance, "as regards the *learning process* as such - prefixed with "e-" or not - the viewpoint is that human learning is best described by the information-processing approach in cognitive psychology, in line with the ideas endorsed in [2]: "Most modern information-processing theories are "learning-by-doing" theories which imply that learning would occur best with a combination of abstract instruction and concrete illustrations". Learning should be considered - in both humans and agents -as a process where most effectiveness is reached through a blend of symbolic ("left-hemisphere"-like) and subsymbolic ("right-hemisphere"-like) modi operandi" [5].

Though, confusion become delusion when "concurrency" and "distribution" were perceived as conceptually close enough that agent-oriented applications - concurrent *par excellence* - could be effective if they are designed using mechanisms conceived (and used successfully) for distributed systems. Since that is a central claim of this paper, it must be elaborated a bit.

When designing distributed systems - examples above show it clearly - distribution was mostly part of the *solution* not part of the *problem*. Indeed, in most cases, the problem was a whole and, *iff* it

such a problem can be split into subproblems, the entirety is disaggregated, the subproblems are solved and, finally, the partial solutions are re-aggregated. At the programming level they run in parallel and, when accessing shared resources, need *mutual exclusion*. On the contrary, applications devised in line with the "Computing as Interaction" paradigm - above all agent-based applications - entail at least two interactants (the interface agent and its owner), evolving in *parallel*, *autonomously* but not *independently* (since they interact as in any normal communication process: inform, wait, interrupt each other, etc.). Briefly, they are *concurrent* processing threads; their programming entails *multithreading*. The crucial software engineering problem is that, while an API (application programming interface) able to support multithreading covers all requirements for mutual exclusion, the opposite is true just in very simple cases. Worse, in most cases, designing concurrent applications with API intended for distributed systems results in severe ineffectiveness. Hence, acknowledging the difference between distribution and concurrency is paramount not just at epistemic level, but at engineering level too. A relevant step into diminishing confusions was the different name given in *C#* to a instruction existing in *Java*, without changing its semantics: *"Synchronize"* is now called *"Lock"* - expressing what it really does (tellingly, almost the opposite of what its previous name claimed to do, since - to preserve coherence - it reduces parallelism).

5 Models, Methods, Mechanisms

Beside the structure as position paper, to give more weight to the assertions regarding the paradigm shifts symbolised in Figure 1, results of recent research (in line with the paradigms endorsed above) are abridged and commented upon. All software entities mentioned below are *models, methods,* and *mechanisms* but they are abridged without grouping them corresponding to their kind, since they are commented upon in software engineering papers, as well as in [6, 5] and papers referred to there.

There are three categories of mechanisms developed for affordable non-algorithmic agent-based applications in OHDUE:

A) Innovative mechanisms dedicated to "Computer-Aided *x*", where *x* stays for almost any intellectual activity, within the software engineering toolbox AGORITHM (*AG*ent-*OR*iented *I*nteractive *T*ime-sensitive *H*euristic *M*echanisms). So far they are implemented or in earlier development stages, but only for solving toy problems or even in simple experimental models, for *x* = Decision, Learning, Semiosis.

B) Existing mechanisms employed in previous research (before 2005, mainly in experimental models for captologic virtual therapists, carried out as pseudoavatars). Their structure is based on common API functions callable from a customary Java development environment.

C) Conceptualized within the framework of some PhD theses in preparation. To increase paradigmatic relevance they are ordered in relation to Figure 1, that is focusing on the missing "L" in the toolbox name. (Of course, a bijection is out of question.):

- *Decision-Making with Future Contingents. DOMINO* (*D*ecision-*O*riented *M*echanism for "*I*F" as *N*on-deterministic *O*perator). Developed primarily for decision making (typical application: managing overbooking) it is meant to deal with undecidability due to any kind of future contingents. It is a "three-output IF" where the semantics of the third value is a blend of a Łukasiewicz "i" interpreted as "unknowable" and a Kleene *"u"* interpreted as "temporary lack of knowledge". (In fact, the semantics of "Undecidable" is re'fined to "Undecidable in the time span given".)
- *Analog Input*. Scrollbar input is proposed for all kind of data: uncertain knowledge, intrinsically fuzzy, roughly estimated, etc.

- *Dynamic priorities*. Are applied for:
  a) fine-tuning agent priorities (mainly the features of Multi-Agent Systems) to manage situations "Just In Time";
  b) fading out retention in "thick time";
  c) boosting response of exception handlers.
- *Exception-Driven Reactivity*. Prompt response to asynchronous must be mostly stimulus-driven because interaction between basically reactive entities. To respond promptly, interrupts are reflected asynchronously in exceptions with dynamic propagation.
- *Antientropic Self-Cloning.* Developed to implement "strange loops" (via Gödelian self-reference) as first step of investigating agent self-awareness, it means spawning an agent identical to itself, preserving self-representation (its *"I"*), but with an enriched world model (via Lamarckian evolution).
- *Flexible Cloning.* To reach efficient polymorphism the copies are purposely imperfect, spawning an agent quasi-identical to its parent; differences between clones become extensive only after recurring cloning (a clone is just a "slightly altered alter ego").

Unfortunately, the mechanisms listed above have - beside lacking validation *in vivo* (some of them not even in *ovo*) - a double vulnerability: they are either incremental as regards the "Kelvin way of thinking" or too loosely linked to new paradigms. Thus, what is their relevance? To break the vicious circle - since there is no "methodology for paradigm shift" - to leave behind the 3^{rd} Order Ignorance [3], software should be considered "not a product, but rather a medium for the storage of knowledge. [...] The other knowledge storage media being, in historical order: DNA, brains, hardware, and books. [...] Software development is not a product-producing activity, it is a knowledge-acquiring activity." (That is neither fact, nor proof, it is just expectation.)

6 Conclusions and Intentions

a) Non-deterministic software is unavoidable; its development entails not just new design principles but new computing paradigms.

b) Agent-oriented systems, to be effectual, should merge conventional agent design with approaches employed in advanced distributed systems (where parallelism is intrinsic to the problem, not just a mean to speed up).

c) the AGORITHM toolbox - still not sufficient as technological infrastructure for agent-oriented software - is a good framework to go ahead.

d) The paradigm shift "from Kelvin to Zadeh" becomes urgent to keep pace with a rapidly changing e-world.

e) Almost as corollary, these paradigm shifts entails also attitudinal ones: shifting from *multi-*, through *inter-*, towards genuine *trans*-disciplinarity.

As regards the prospects of non-algorithmic agent-oriented software, short-term intentions include enhancing the AGORITHM toolbox with two mini-ontologies:

a) *Dynamic: I* (agent), *You* (master), and *Rest of the world*.

b) *Visual*: Visual rules should simulate *"the arrow of time"*.

Acknowledgement. This work was partially supported by the Ministry of Education and Research through contract No. 12 - 092/2007.

Bibliography

[1] AgentLink III. *Agent based computing. AgentLink Roadmap: Overview and Consultation Report.* University of Southampton. http://www.agentlink.org/roadmap/al3rm.pdf, 2005.

[2] J.R. Anderson, L.M. Reder, H.A. Simon. *Applications and Misapplications of Cognitive Psychology to Mathematics Education.* Texas Educational Review, 2000.

[3] P.G. Armour, *The Five Orders of Ignorance.* Comm. ACM, 43 (10), 17-20, 2000.

[4] B.E. Bărbat, *Agent-Oriented Intelligent Systems.* Romanian Academy Publ. House, Bucharest, 2002 (in Romanian, "Grigore Moisil" Prize of the Romanian Academy).

[5] B.E. Bărbat, E-Maieutics. Rationale and Approach. *International Journal of Computers, Communications & Control*, 3, Supplement: Suppl. S, 40-54, 2008.

[6] B.E. Bărbat, Natural time for artificial agents. *Abstracts of ICCCC Papers*, Băile Felix, May 15-17, 27-27, 2008.(Invited paper.)

[7] I. Dziţac, *Artificial Intelligence.* Ed. House "Aurel Vlaicu" University, 2008 (in Romanian).

[8] I. Dziţac, *Parallel and Distributed Methods for Algebraic Systems Resolution*, CCC Publications, Agora University Publishing House, 2006. (in Romanian).

[9] I. Dziţac, ICCCC 2008 & EWNLC 2008 Celebrates Bardeen's Centenary and Welcomes Professor Zadeh, *International Journal of Computers Communications & Control*, 3 (Suppl. S):16-25, 2008.

[10] I. Dziţac, I. Moisil, Advanced AI Techniques for Web Mining, *Proc. of 10th WSEAS International Conference on Mathematical Methods, Computational Techniques and Intelligent Systems* (MAMECTIS '08, Corfu, Greece, 343-346, 2008.

[11] I. Dzitac, G. Moldovan, *Distributed Systems: Information Models*, Agora University Publishing House 2006 (in Romanian).

[12] FIPA TC Agent Management. *FIPA Agent Management Specification*. Standard SC00023K (2004/18/03). http://www.fipa.org/specs/fipa00023/ SC00023K.pdf

[13] G. Gigerenzer, A. Edwards. Simple tools for understanding risks: from innumeracy to insight. *British Medical Journal*, 327, 741-744, 2003.

[14] G. Gigerenzer, R. Selten. *Bounded Rationality*. MIT Press, Cambridge, 2002.

[15] F. Herrera, Genetic fuzzy systems: taxonomy, current research trends and prospects. *Evol. Intel.*, 1, 27-46, 2008.

[16] D. Kahneman, *Maps of Bounded Rationality: Psychology for Behavioral Economics.* Lecture (when receiving Nobel Prize; revised version). Stockholm, Nobel Foundation, 2002.

[17] E. Livingston, *Ethnographies of Reason.* Ashgate, Aldershot, UK, 2008.

[18] A. Rubinstein, *Modeling Bounded Rationality*. MIT Press, Cambridge, 1998.

[19] C. Sieckenius de Souza, *The Semiotic Engineering of Human-Computer Interaction.* The MIT Press, 2005.

[20] H.A. Simon, *Models of Bounded Rationality.* MIT Press, Cambridge, 1997.

[21] D. Spinellis, *Rational Metaprogramming.* IEEE Software, 25, 1, 78-79, Jan/Feb, 2008.

[22] N. Wirth, *Algorithms + Data Structures = Programs.* Prentice Hall, 1978.

[23] L.A. Zadeh, D. Tufis, F.G. Filip, I. Dzitac, (Eds.), *From Natural Language to Soft Computing: New Paradigms in Artificial Intelligence*, Romanian Academy Publishing House, 2008.

[24] F. Zambonelli, A. Omicini. Challenges and Research Directions in Agent-Oriented Software Engineering. *Autonomous Agents and Multi-Agent Systems*, 9, 253-283, Kluwer Academic Publishers, 2004.

Ioan Dzitac received M. Sc. in Mathematics (1977) and Ph.D in Information Sc. (2002) from "Babes-Bolyai" University of Cluj- Napoca. His current research interests include different aspects of Artificial Intelligence and Parallel and Distributed Computing. He has edited 6 conference proceedings, published 16 books and more than 50 scientific papers in journals and conferences proceedings. He was member of the Program Committee of more than 30 international conferences.

Boldur E. Bărbat M.Sc. in Electronic Engineering, postgraduate specialising in Programming, Ph.D. in Digital Computers ("Politehnica" University Bucharest). He is with "Lucian Blaga" University Sibiu from 1997, (full professor, Faculty of Engineering, Faculty of Sciences) and "Politehnica" University Timişoara from 2005 (Faculty of Automation and Computers, advisor for doctoral studies in Computer Science). 25 books (a monograph received the Romanian Academy IT Prize, 2002). About 50 papers in English from 2001. Current research interests: Time in Artificial Intelligence; Nonalgorithmic decision support systems; Computer-Aided Semiosis; Self-awareness of bodiless agents; Agent-Oriented Software Engineering; Logics for agents; Emergence in Agent-Based Systems.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 27-40

The Impact of Java Applications at Microarchitectural Level from Branch Prediction Perspective

Adrian Florea, Arpad Gellert, Lucian Vinţan, Marius Velţan

Adrian Florea, Arpad Gellert, Lucian N. Vinţan
"Lucian Blaga" University of Sibiu, Computer Science Department
Emil Cioran Street, No. 4, Sibiu 550025, Romania
E-mail: {adrian.florea,arpad.gellert,lucian.vintan}@ulbsibiu.ro

Marius N. Velţan
SC IMC Information Multimedia Communications SRL Sibiu, Product Development Department
Cristian Street, No. 21, Sibiu 550073, Romania
E-mail: marius.veltan@im-c.de

Abstract: The portability, the object-oriented and distributed programming models, multithreading support and automatic garbage collection are features that make Java very attractive for application developers. The main goal of this paper consists in pointing out the impact of Java applications at microarchitectural level from two perspectives: unbiased branches and indirect jumps/calls, such branches limiting the ceiling of dynamic branch prediction and causing significant performance degradation. Therefore, accurately predicting this kind of branches remains an open problem. The simulation part of the paper mainly refers to determining the context length influence on the percentage of unbiased branches from Java applications, the prediction accuracy and the usage degree obtained using a *Fast Path-Based Perceptron* predictor. We realize a comparison with C/C++ application behavior from unbiased branches perspective. We also analyze some Java testing programs, built using *design patterns* or including inheritance, polymorphism, backtracking and recursivity, in order to determine the features of indirect branches, the arity of each indirect jump and the prediction accuracy using the Target Cache predictor.
Keywords: object-oriented programs, branch prediction, indirect jumps/calls, unbiased branches, benchmarking

1 Introduction

As the Internet evolves and becomes increasingly popular, the need for a portable programming language becomes increasingly important. Providing a common language interface Java technology, that includes Java language, Java Virtual Machine (JVM) and Java API, allows programs to be used across a wide range of machine platforms from browsers, e-commerce, financial and bioinformatics applications on desktop computing systems to software for real-time embedded systems (intelligent mobile phones, palms, PDA, etc.). The portability, the object-oriented and distributed programming models, multithreading support and *automatic garbage* collection are features that make Java very attractive for application developers. In addition to portability, security and ease of application development has made it very popular with the software community. According to Sun Microsystems, in 2007 there were more than 5 billion java enabled devices (desktops, phones, cards, settop boxes, etc), from among 2.1 billions are phone handsets or PDA. Also, more than 6 million professional java developers are programming these devices [1].

Besides, the last decade is characterized by the advance of visual or object-oriented applications and the portability trend of many of them, as well as the usage of Dynamically-Linked Libraries. To support polymorphism the object-oriented languages such as Java, C#, and C++ include dynamically-dispatched function calls (i.e. virtual functions) whose targets are not known until run-time because they depend on the dynamic type of the object on which the function is called. Virtual function calls are implemented using indirect jump/call instructions in the instruction set architecture (ISA). From microarchitectural point of view object-oriented programming techniques exercise different aspects of computer architecture to support the object-oriented programming style [2], [3]. The object-oriented languages have significantly more indirect jumps than procedural languages. In addition to virtual function calls, indirect jumps are commonly used in the implementation of programming language constructs such as switch-case statements (with more than five options [3]), jump tables, indirect calls through function pointers, and interface calls [4].

In current pipelined processor designs a particularly difficult challenge consists in target prediction for indirect jumps and calls. Because the target of an indirect jump (call) can change with every dynamic instance of that jump, predicting the target of such an instruction is really difficult. Such hard-to-predict jumps not only limit processor performance and cause wasted energy consumption but also contribute significantly to the performance difference between procedural and object-oriented languages. Simulations made on Intel Core2 Duo processor analyzing suggestive applications such Matlab, Cygwin, Excel, Firefox, Winamp and InternetExplorer show that about 41% of all jump mispredictions are due to indirect jumps [5]. Besides indirect jumps/calls, another class of hard-to-predict branches consists in unbiased branches. In two of our previous papers [6], [7] we found a minority of dynamic conditional branches showing a low degree of polarization towards a specific prediction context since they tend to shuffle between taken and not-taken. We called them unbiased branches and we have demonstrated that, irrespective of the prediction information length and type, used in the state of the art branch predictors, these branches are characterized by low prediction accuracies (at average about 70%). We have demonstrated that actual programs have a significant fraction of such difficult to predict branches that severely affect prediction accuracy leading to performance degradation and additional power consumption, which are very important especially in the embedded systems.

The main goal of this paper consists in pointing out the impact of Java applications at microarchitectural level from two perspectives: unbiased branches and indirect jumps/calls, such branches limiting the ceiling of dynamic branch prediction and causing significant performance degradation. Therefore, accurately predicting this kind of branches remains an open problem. Following our aim, first we extend our tool Advanced Branch Prediction trace driven Simulator (ABPS) [8] for analyzing (detecting and predicting) unbiased branches from SPEC JVM98 benchmarks [9], a suite of eight standardized Java applications proposed to better understand where performance bottleneck exist in the Java Virtual Machines and what optimizations are possible. We determine the context length influence on the percentage of unbiased branches from Java applications, the prediction accuracy and the usage degree obtained using a *Fast Path-Based Perceptron* (FPBP) predictor [10]. Also, we realize a comparison with the behavior of C/C++ applications from unbiased branches perspective.

The differences between object-oriented and procedural applications from execution viewpoint on ILP processors guided us to analyze some Java testing programs built using *design patterns*. Based on these Java applications we illustrated that the two computer science components (software and hardware) are just apparently *"disjointed"*. Thus, in section 4 we describe six well-known design patterns [11], [12] and applications developed based on them. Additionally, we created four simple applications that include inheritance, polymorphism, backtracking and recursivity. These programs were *execution-driven* simulated on "Dynamic SimpleScalar" environment (DSS) [13] in order to determine the features of indirect branches, the arity of each indirect jump and the prediction accuracy using the Target Cache predictor.

The remainder of this paper is organized as follows. In section 2 we illustrated comparatively the

C/C++ and Java languages, emphasizing especially the differences between them from the execution viewpoint on host architectures. Section 3 describes the implemented predictors used for studying unbiased branches and indirect jumps/calls. Section 4 includes simulation methodology and presents the benchmarks used for simulation. In section 5 we illustrate the experimental results obtained using our developed simulator. Finally, section 6 suggests further work directions and concludes the paper.

2 Some differences between C/C++ and Java languages

C++ programming language was designed and implemented as an extension of C language that supports data abstraction, object-oriented concepts and generic programming, dedicated for desktop programming. Unfortunately, it inherited also all problems from C. Java was created as a new object-oriented and distributed language, without compatibility constraints, mainly dedicated for small electronic devices that embeds networking capabilities and has cross platform portability and also for Internet applications programming. In Java were removed all the procedural programming concepts that proved to be dangerous or unsecured.

Unfortunately, the portability leads to decreasing the execution speed of Java programs. Thus, it was observed that the interpreted Java bytecode is 30 times slower than an optimized C++ code. However, just in time (JIT) compilers could produce considerable performance improvements over interpretation (20 times) by removing dispatch overhead and applying some optimizations to the generated code. The classes used by server applications are executed using JIT compilers in order to obtain high processing performance. Though, the memory required by JIT compilers (additional space for the generated native code, for profiling and other data structures) is very expensive especially in the case of embedded systems. For these systems are preferred native Java processors that use small memory and have a small power consumption, and last but not least, have small costs, easy to support by any consumer. From this reason, the modern researches try to optimize the Java virtual machines. Using server applications, when Sun JVM was updated from the 3^{rd} version to the 6^{th} version the processing performance was improved with 344%. Similarly, using desktop applications, the performance increases but not very spectacular [1]. Nowadays Java virtual machines use even *dynamic metrics* that optimize the JVM taking into account not only the host processor but also the applications' behavior. There are applications that run better if they are written in Java than in C++. Simulation results on the SciMark benchmark have shown that Java runs with 4% quicker than C++, and LINPACK benchmarks are slower with only 2% in Java against C++. Also, the Garbage Collector is much quicker than dynamic memory allocation / de-allocation realized with the *malloc* and *free C++* instructions [1]. Thus, after releasing the last JVM version (JRE 6.0), Sun Microsystems engineers claim that they destroyed the myth *interpreted language means not performing*.

3 Predictors implemented for studying unbiased branches and indirect jumps/calls

For indirect branch prediction we considered a tagged Target Cache (TC) predictor first introduced by Chang [15]. The Target Cache improves the prediction accuracy for indirect branches by choosing its prediction from the last N targets of the indirect branch that have already been encountered. When an indirect jump is fetched, both the PC and the *globalHR* (a history register that retains the behavior of last k conditional branches) are used to access the TC for predicting the target address. As the program executes, the TC records the target for each indirect jump target encountered. Our proposed scheme uses for set selection the least significant bits of the word obtained by hashing (XOR) the indirect jump's address (PC) and the global history (*globalHR*). The most significant bits of this hashing form the *Tag*. In the case of a hit in the TC the predicted target consists in the corresponding address belonging to that TC set (field *Adr* from Figure 1). In the case of a misprediction, (the Tags coincide but the target addresses

differ) after the indirect branch is resolved, the Target Cache entry is updated with its real target address. We have implemented and simulated a *P*-way set associative TC, where P=1, 2, 4 like that presented in Figure 1. In the case of a miss in the TC the prediction is considered wrong, it does not provide any value and a new entry updated with the proper *tag* and *target* is added to the respective set, according with the implemented LRU replacement algorithm.

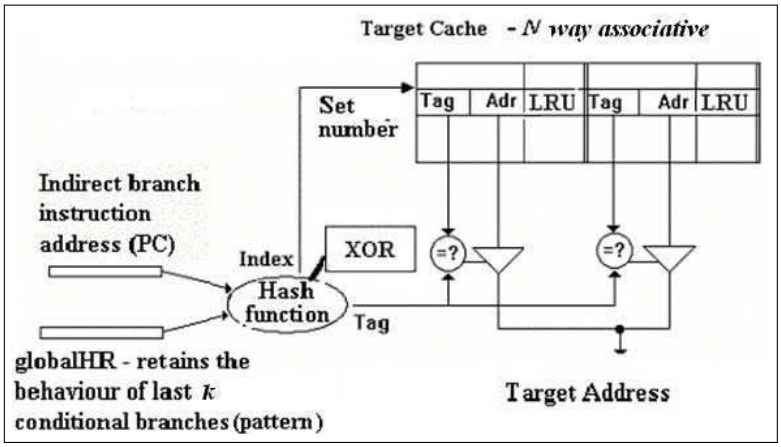


Figure 1: The structure of Target Cache predictor

The most accurate single-component branch predictors in the literature are neural branch predictors [10], [16] and [17]. Their main advantages consist in the possibility of using longer correlation information at linear cost. The *Perceptron* predictor - the simplest neural branch predictor - keeps a table of *weight vectors* (small integers that are learned through the *perceptron* learning rule) [16]. As in global two-level adaptive branch prediction, a shift register records a global history of conditional branche outcomes, recording *true* for *taken*, or *false* for *not taken*. To predict a branch outcome, a weight vector is selected by indexing the table with the branch address modulo the number of weight vectors. The dot product of the selected vector and the global history register is computed, where *true* in the history represents 1 and *false* represents -1. If the dot product is at least 0, then the branch is predicted taken, otherwise it is predicted not taken. Once the perceptron output has been computed, the training algorithm starts: it increments the *i*-th correlation weight when the branch outcome agrees with the *i*-th bit from the global branch history shift register and decrements the weight otherwise. Unfortunately, the high latency of the perceptron predictor and its impossibility to predict the linearly inseparable branches makes it impractical yet for hardware implementation. In order to reduce the prediction latency, the *Fast Path-based Perceptron* [10] chooses its weights for generating a prediction according to the current branch's path, rather than according to the branch's PC and history register. The prediction latency is hidden due to the speculative calculation of the perceptron's output. Let PC_0 be the current branch address, and PC_i be the i^{th} most recent branch address in the *path history*. For the perceptron, each weight is chosen with the same index based on PC_0. For the FPBP, each weight w_i is chosen based on an index derived from PC_i. This provides path history information that can improve prediction accuracy, and spreading out the weights in different entries also helps to reduce the impact of inter-branch aliasing. To implement the FPBP, the lookup phase is actually pipelined over many stages based on the overall branch path / global history length. Undertaken from [18], Figure 2 exposes a very intuitive scheme of *Fast Path-based Perceptron*. For a branch at cycle *t*, the FPBP starts the prediction at cycle *t* - *h* using PC_h. For each cycle after *t* - *h*, the FPBP computes partial sums of the dot-product of weights vector and global history register. Pipeline stage *i* contains the partial sum for the branch prediction that will be needed in *i* cycles. At the very end of the pipeline, the critical lookup latency consists of looking up the final weight and performing the final addition.

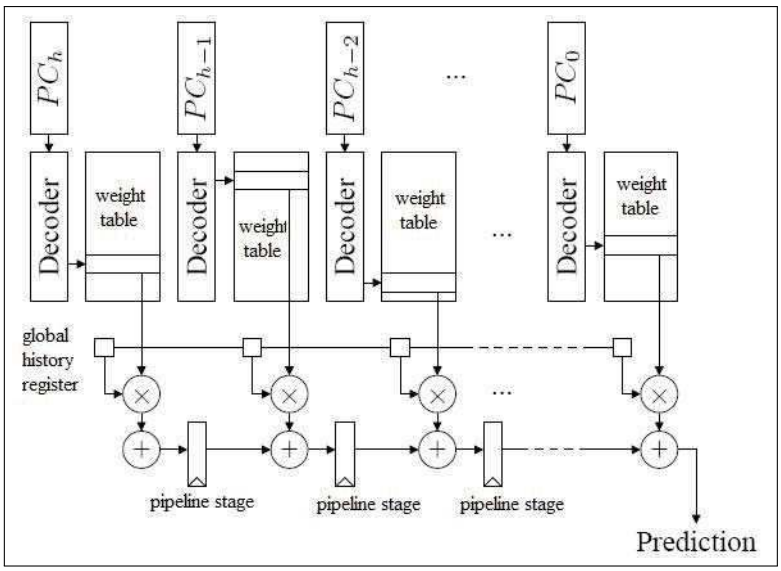


Figure 2: The structure of Fast Path-based Perceptron predictor

4 Simulation Methodology and Benchmarking

Related to our evaluations on procedural programs, we collected results from different versions of SPEC benchmarks: 3 integer (*li, go, cc1*) and 4 floating point (*applu, apsi, fpppp, hydro*) SPEC'95 benchmarks. From the integer SPEC2000 suite, we simulated 8 benchmarks (*gzip, b2zip, parser, crafty, gap, gcc, twolf* and *mcf*). We also simulated some SPEC'95 benchmarks in order to compare their behavior with that of the more recent SPEC2000. All these benchmarks cover a lot of applications ranging from compression (text/image) to word processing, from compilers and architectures to games enhanced with artificial intelligence, etc. These programs were selected based on their characteristics: a high number of indirect branches and high target entropy.

Recall that the majority of indirect branches are generated by object oriented programs. In software engineering, after structured programming and object orientation, one of the most important innovation with impact on the art of constructing software systems were *design patterns* [11]. It is hard to understand and practice effective object orientation without being conscious of design patterns and their suitability for the problem at hand. According to [19] a design pattern represents a formal way of documenting a solution to a design problem in a particular field of expertise. In software engineering, a design pattern is a general reusable solution to a commonly occurring problem in software design. Using real-life scenarios helps in understanding programming abstraction to some extent but do not provide the deep insight that one gets by actually seeing these abstractions. A design pattern is not a finished design that can be transformed directly into code. Patterns show how to build systems with good object oriented design qualities. Most patterns allow some part of a system to vary independently of all other parts. Patterns provide a share language that can maximize the value of communication between developers.

For indirect branch analysis of Java applications we simulated six design pattern applications [11], [12] and also four other simple object-oriented programs: *Jbytemark* that sorts arrays of values through many sorting methods, *Queens_5* that uses inheritance and polymorphism for solving different problems through recursive backtracking, *Switch_06* that illustrates the indirect jump generation from switch/case statements and *Simple_Inheritance* that uses polymorphism and treats uniformly the heterogenic array of objects. Table 1 illustrates the characteristics of the design pattern applications [11], [12] used for indirect branch analysis.

The results related to the indirect branch analysis illustrated in chapter V were obtained after simula-

Table 1: Design Pattern characteristics

Design Pattern	**Application**	**Characteristics**
Strategy	DuckSimulator	It is a *behavioral* pattern that defines a family of algorithms, encapsulate each one, and make them interchangeable. Strategy lets the algorithm vary independently on the clients that are using it. Example: *Adventure Game*
Observer	WeatherStation	It is a *behavioral* pattern that defines a one-to-many dependency between objects so that when one object's state changes, all its dependents are notified and updated automatically. Example: *Newspaper subscription*
Decorator	StarbuzzCoffee	It is a *structural* pattern that dynamically attaches additional responsibilities to an object. Decorators provide a flexible alternative to sub-classing for extending functionality. Example: *Dressing a character in a social game*
Factory Method	PreparingPizza	It is a *creational* pattern that defines an interface for creating an object, but let subclasses decide which class to instantiate. Example: *Factory methods*
Template Method	ServeingDrinks	It is a *behavioral* pattern that defines the skeleton of an algorithm in an operation, deferring some steps to subclasses. Template Method lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure. Example: *Searching in a database*
State	Gumball Machine Test	It is a *behavioral* pattern that allows an object to alter its behavior when its internal state changes. The object will appear to change its class. Example: *Automated Teller Machine*.

tion on 2.4 GHz Pentium IV microprocessor under Microsoft WindowsXP operating system, disposing of Cygwin emulator. Taking into account that the simulation time of over 600.000.000 dynamic instructions from aforementioned Java benchmarks vary between 35 minutes (Queens_5) and more than 24 hours (jBytemark) we chose to limit the simulation to maximum 10 billion instructions. From the SPEC'95 suite we simulated 100 millions instructions, from the SPEC2000 set we simulated 500 millions instructions, whilst all Java applications were completely simulated (except JBytemark benchmark), the total number of instructions ranging between 600 millions and 4290 millions instructions, respectively.

For indirect jump/call analysis we developed a cycle-accurate execution driven simulator (*sim-jindir* [12]), derived from the *sim-cache* simulator of the Dynamic SimpleScalar toolset [13]. We modified it to incorporate the indirect jump predictor proposed in Section 3 in order to measure the number of indirect (static / dynamic) branches, the arity of each indirect jump and to predict targets of indirect jumps and calls, respectively. DSS represents a standardized software tool used to simulate Java programs executed on a Java Virtual Machine (in our case IBM Jikes Research Virtual Machine) and was created to extend the SimpleScalar toolset [20]. Jikes RVM is a virtual machine that runs Java programs by compiling them to native code at runtime. It comes with two Java bytecode to native code compilers. The fast baseline compiler does not perform any optimizations. The optimizing compiler performs a complete set of optimizations most importantly *inlining* and *register allocation*. The *sim-jindir* provides a wider variety of configuration options. We can vary the TC associativity degree, the number of sets of TC, the maximum number of instructions executed, the simulated benchmark, and the platform used for simulation.

Table 2 illustrates the characteristics of standardized SPECJVM98 benchmarks [21] used for unbiased branch analyze.

ABPS is a trace driven simulator used to analyze (detect and predict) unbiased branches. It is written in Java and includes state of the art branch predictors such as Fast Path-based Perceptron predictor. This request as inputs parameters: the number of entries in prediction table, the global history length, the threshold value used by the learning algorithm, number of bits for storing the weights. ABPS can detect unbiased branches or predict all branches or only those that are unbiased. The ABPS simulation results consist in four important metrics. The prediction accuracy is the number of correct predictions divided to total number of dynamic branches. We compute also a confidence metric that represents the total cases when the prediction was correct and the perceptron did not need to be trained (the magnitude of perceptron output was greater than the threshold) divided to the total number of correct predictions. While the first two have impact on processor's performance, the next two metrics have direct influence on transistors' budget and integration area (the number of perceptrons used in the prediction process and the saturation degree of perceptrons). The saturation degree represents the percentage of cases when the weights of perceptrons cannot be increased / decreased because they are saturated. If the last two metrics are quite low means that the perceptrons are underused.

5 Experimental Results

Tables 3 and 4 illustrate comparatively the percentage of static / dynamic (indirect) branches within the tested programs.

Figures 3 and 4 illustrate the arity of indirect branches from static and dynamic points of view, respectively. The arity means the number of distinct dynamic targets of the same indirect jump/call. This can be determined either after simulations or knowing profile information (identifying branch class) either estimated through source code level analysis. After this process, monomorphic jumps/calls (having a single target) can be successful predicted by a predictor without history (BTB), yet, for polymorphic branch prediction (with two or more distinct targets) additional information are necessary.

Analyzing Tables 3 and 4 and Figures 3 and 4, we observe some significant differences related to the behavior of Java applications against C/C++ applications at microarchitectural level, from indirect

Table 2: SPEC JVM98 characteristics

Benchmark	**Characteristics**
227_mtrt	It represents a modification of 205_raytrace, a ray tracer that creates a pictorial scene portraying a dinosaur. It uses a multi-threaded driver, where the threads render the scene from an input file.
202_jess	This benchmark is an expert system shell written entirely in Java. The intent of Jess is to give Java applets the ability to "reason" by continuously applying a set of rules. The workload used in the benchmark solves a set of puzzles.
201_compress	Represents Java version of the 129.compress benchmark from the SPEC'95 benchmark suite, but improves upon that benchmark in that it compresses real data from files using a modified Lempel-Ziv method (LZW). It is highly recursive.
209_db	It performs multiple database operations (insertion, removing, finding, sorting) on a 1 MBytes memory resident database.
222_mpegaudio	This benchmark is an application that decompresses audio files defined by the ISO MPEG Layer-3 audio specification. The MP3 encoding technique allows data compression of digital audio signals up to a factor of 12, without losing sound quality as perceived by the human ear. The workload consists of about 4MB of audio data.
228_jack	It represents a Java parser that is based on the Purdue Compiler Construction Tool Set, in fact an earlier version of JavaCC compiler. The workload consists of a file named jack.jack, which contains instructions used for the generation of jack itself. This is fed to jack so that the parser generates itself multiple times (highly recursive).
213_javac	It represents the Java compiler from the JDK 1.0.2.

Table 3: Statistics about *static* (indirect) branches from the analyzed benchmarks

Testing programs (1)	**Average of indirect static branches** (2)	**Average of static branches** (3)	**Average percentage of indirect static branches** (4) = (2)/(3)*100%
SPEC 95	59	4759	2.15%
SPEC 2000	57	8187	0.86%
Design Patterns	6765	16619	40.69%

Table 4: Statistics about *dynamic* (indirect) branches from the analyzed benchmarks

Testing programs (1)	**Average of indirect dynamic branches** (2)	**Average of dynamic branches** (3)	**Average percentage of indirect dynamic branches** (4) = (2)/(3)*100%	**Maximum percentage of indirect dynamic branches**
SPEC 95	432041	13807188	2.47	5.63%
SPEC 2000	790826	95218337	0.78	2.03%
Design Patterns	17904385	93676540	18.82	20.41%

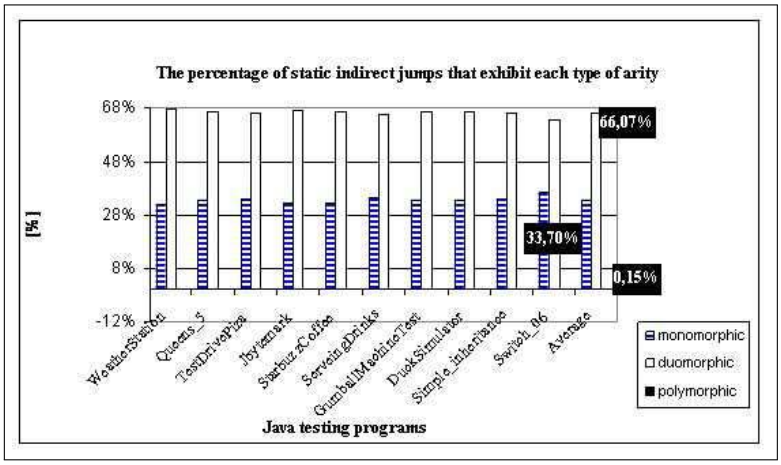


Figure 3: The arity of indirect branches from static point of view

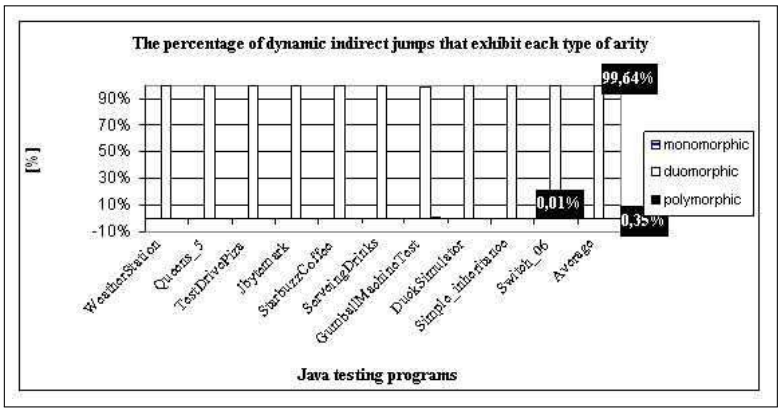


Figure 4: The arity of indirect branches from dynamic point of view

branches perspective: first, the percentage of dynamic indirect branches from Java testing programs (maximum 20.41%) is much higher than that from procedural or object oriented C/ C++ programs (maximum 5.63%, see also [22]). The results are similar with those reported in [23]. According to Tao, the performance results are different depending on the JVM mode of execution. In interpreter mode, 19.5% of all dynamic branches are indirect branches and 11.8% of all branches are polymorphic indirect branches. In JIT compiler mode, 12.4% of dynamic branches are indirect branches and only 5% are polymorphic branches. Second, the significant percentage of duomorphic branches from Java programs, recommends using a minimum 2-way associative Target Cache structure for indirect branch prediction. The very small percentage of dynamic polymorphic indirect branches from our proposed Java programs could be due to the fact that these programs do not represent standardized benchmarks but only simple testing programs. However, we consider that there are two major reasons for the high number of indirect branches targets within Java applications: 1) as an object-oriented programming language, Java promotes a polymorphic programming style in which late binding of subroutine invocations is the main instrument for modular code design. With the help of virtual method table, the implementation chosen for Java interpreters and compilers, it executes an indirect branch for every polymorphic call. More than that, in Java, instance methods are virtually declared by default. If they are not explicitly declared final, they can be overridden in subclasses. 2) Switch/case statements (in the bytecode translation routine of the interpreter) and indirect function calls through pointers (calls to the dynamically shared native interface libraries) from runtime interpretation and JIT compilation of bytecodes performed by the Java Virtual Machine are subject to high indirect branch frequency.

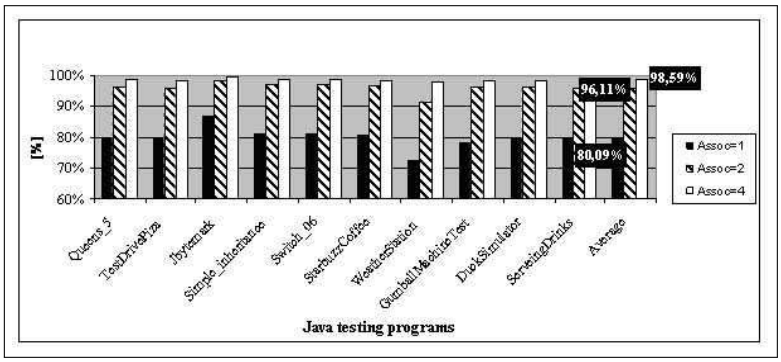


Figure 5: The indirect branch prediction accuracy using a Target Cache Predictor with 512 entries

From Figure 5 we observe that indirect branch prediction accuracy increases as the associativity degree of TC structure increases. The improvement is significant for a 2-way associative TC against a direct mapped one (20%) whilst the improvements diminish to 2.58% for a 4-way associative TC compared with a 2-way associative one. The results from Figure 5 are correlated with those from Figure 4 which show a high percentage of duomorphic and an insignificant percentage of polymorphic indirect branches. The improvement of prediction accuracy from 96.11% to 98.59% is due to the miss conflict and capacity miss accesses at a 2-way associative TC (that are solved by increasing the associativity degree), but also to the smaller percentage of polymorphic indirect branches.

Analyzing procedural programs (SPEC'95 and SPEC2000), we have shown in [24] that the best prediction accuracy of indirect branches obtained using a 8-way associative Target Cache with 128 entries was 88.97%, for Target Cache capacities greater or equal with 256 entries the prediction accuracy becoming saturated.

Figure 6 exhibit comparatively the behavior of procedural benchmarks (SPEC2000) versus object-oriented benchmarks (SPEC JVM98) from unbiased branches perspectives. Repeating the detection methodology for a length-ordered set of contexts used by us in [25] it could be observed how the number of unbiased branches decreases within both procedural and object-oriented applications. On

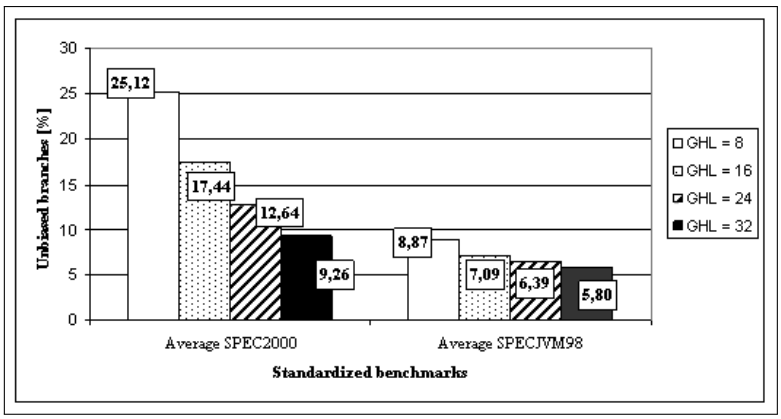


Figure 6: Reducing the number of unbiased branches by increasing global history length (GHL)

the SPEC2000 benchmarks the percentage of unbiased branches decreases in average from **25.12%** to **9.26%**. We consider that this value is still too high and further investigations are required. For the SPEC JVM98 benchmarks the percentage of unbiased branches decreases from **8.87%** to **5.80%** (almost half compared to the SPEC2000 integer benchmarks).

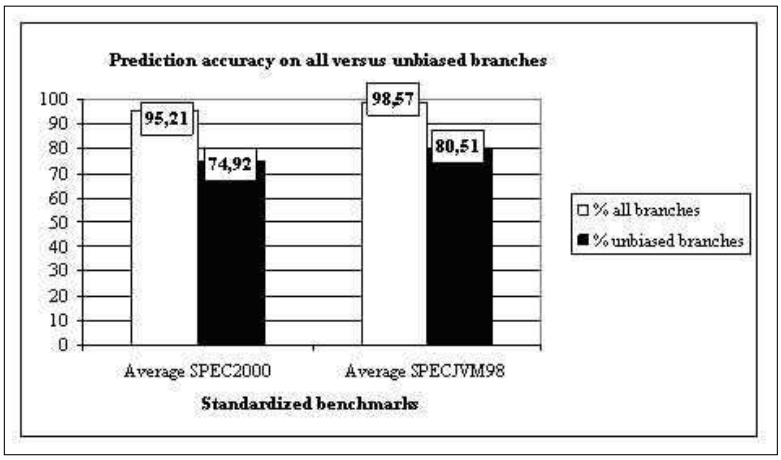


Figure 7: Fast Path-Based Perceptron prediction accuracies on SPEC2000 versus SPEC JVM98 benchmarks using a table of 1024 entries

In Figure 7 we considered, comparatively on SPEC2000 versus SPEC JVM98 benchmarks, the impact of unbiased branches on a FPBP predictor with a table of 1024 entries considering a global history register of 32. We achieved on the SPEC2000 suite an average prediction accuracy of 95.21% on all branches but the impact of unbiased branches still remained significant at 74.92%. However, in Java benchmarks, the smaller percentage of unbiased branches has a lower impact on prediction accuracy. The prediction accuracy obtained on all branches is very high - 98.57% at average and even unbiased branches are predicted more accurately than those from the SPEC2000 integer benchmarks (where 11 of 12 are procedural applications).

Based on the number of perceptrons used, we determined the usage degree of perceptron table. This metric will directly affect the prediction accuracy and indirectly the processing performance lost. The reduced usage degree (39.77% in average on the SPEC2000 benchmarks) proves a high interference degree in the perceptrons table, further diminishing the prediction accuracy, the only advantage being the reduced integration area cost. However, this aspect has been not observed on the SPEC JVM98 benchmarks where the perceptron table is almost entirely exploited (99.65% in average).

6 Conclusions and Further Work

Java technology from software for embedded systems to commercial and research applications and also, visual, interactive and desktop applications are characterized by high percentage of indirect branches. Unfortunately, the prediction accuracy of indirect branches is still very low because many indirect branches have multiple targets that are difficult to predict even with specialized hardware. This paper shows that besides indirect jumps, the unbiased branches represent another class of hard to predict branches in Java applications that limits the ceiling of dynamic branch prediction and causes performance degradation and additional power consumption.

Studying the impact of Java applications at microarchitectural level from branch prediction perspectives we concluded:

- The percentage of dynamic indirect branches from Java testing programs (maximum 20.41%) is much higher than that from procedural or object oriented C/ C++ programs (maximum 5.63%).
- The significant percentage of duomorphic indirect branches ($\geqslant$90%) from Java programs recommends using a minimum 2-way associative Target Cache structure for indirect branch prediction.
- The prediction accuracy improvement is significant for a 2-way associative TC against a direct mapped one (20%) whilst the improvements diminish to 2.58% for a 4-way associative TC compared with a 2-way associative one.
- For Java benchmarks the percentage of unbiased branches decreases from 8.87% to 5.80% (almost half compared to the C/C++ integer benchmarks).
- For Java benchmarks, the smaller percentage of unbiased branches has a lower impact on prediction accuracy. The prediction accuracy obtained on all benchmarks is very high (98.57% in average) and even unbiased branches are predicted more accurately than those from C/C++ integer benchmarks. The lower percentage of unbiased branches occurred in Java benchmarks could be determined by the reduced number of conditional branches from object oriented benchmarks compared with procedural applications. It is well known that object-oriented programs contain many methods with few instructions, and fewer conditional branches implicitly.

Hardware-software interface optimizations are not possible without a deeper understanding process that requires an integrated vision about multiple stages of information processing, coded at different semantic levels. A solution could consist in developing some "*semantic predictors*", based on High Level Language (HLL) information whose importance related to the generation of indirect jumps (polymorphism, indirect function calls, etc.) is proved by us. This might be a completely new approach in branch prediction domain, where HLL semantics are often hidden. In order to efficiently use such information we consider it will be necessary to have a significant amount of compiler support.

Bibliography

[1] Gosling J., *Java: a Tour of the Landscape*, Sun Technology Days at a Glance, Frankfurt, Germany, 3-5 December, 2007.

[2] Calder B., Grunwald D., Zorn B., *Quantifying behavioral differences between C and C++ programs*, Journal of Programming Languages, Volume 2, Issue 4, 1994, pp. 313-351.

[3] Florea A., Vinţan L., Mihu I.Z., *Understanding and Predicting Indirect Branch Behavior, Studies in Informatics and Control*, Volume13, Issue 1, March 2004, pp. 61-82, National Institute for Research and Development in Informatics, Bucharest.

[4] Alpern B., Cocchi A., Fink S., Grove D., Lieber D., *Efficient implementation of Java interfaces: Invoke interface considered harmless*, In Proceedings of the 16th ACM SIGPLAN Conference on Object-Oriented Programming Systems, Languages and Applications, pp. 108-124, October 14-18, 2001, Tampa, Florida, USA.

[5] Joao J.A., Mutlu O., Kim H., Agarwal R., Patt Y., *Improving the Performance of Object-Oriented Languages with Dynamic Predication of Indirect Jumps*, ACM SIGOPS Operating Systems Review, Volume 42, Issue 2, March 2008, pp. 80-90.

[6] Vinţan L., Gellert A., Florea A., Oancea M., Egan C. *Understanding Prediction Limits through Unbiased Branches*, Lecture Notes in Computer Science, Advances in Computer Systems Architecture, vol. 4186, pp. 480-487, 2006, Springer-Verlag Berlin.

[7] Gellert A., Florea A., Vinţan M., Egan C., Vinţan L. *Unbiased Branches: An Open Problem*, Lecture Notes in Computer Science, Advances in Computer Systems Architecture, vol. 4697, pp. 16-27, 2007, Springer-Verlag Berlin / Heidelberg.

[8] Radu C., Calborean H., Crapciu A., Gellert A., Florea A., *An Interactive Graphical Trace-Driven Simulator for Teaching Branch Prediction in Computer Architecture*, The 6th EUROSIM Congress on Modeling and Simulation, September 9-13, 2007, pp. 58 (6 pg.), Ljubljana, Slovenia.

[9] *SPEC JVM98 benchmarks*, http://www.spec.org/jvm98/

[10] Jiménez D. *Fast Path-Based Neural Branch Prediction*, Proceedings of the 36th Annual International Symposium on Microarchitecture, December 3-5, 2003, pp. 243-252, San Diego, CA, USA.

[11] Freeman E., Freeman E., Sierra K., Bates B. *Head First Design Patterns*, O'Reilly Publishing House, First Edition, October 2004.

[12] Veltan N.M. *The Interaction of Java Programs at Microarchitectural Level from Branch Prediction Viewpoint (in Romanian)*, MSc Thesis, 'Lucian Blaga' University of Sibiu, Computer Science Department, Romania, 2008.

[13] *The University of Massachusetts Amherst and the University of Texas - Dynamic SimpleSscalar*, 2004. http://www-ali.cs.umass.edu/DSS/index.html.

[14] Lindholm T., Yellin F., *The JavaTM Virtual Machine Specification*, Sun Press Publishing Hall, 1999.

[15] Chang P.Y., E. Hao, Y.N. Patt., *Target Prediction for Indirect Jumps*, Proceedings of the International Symposium on Computer Architecture, 1997.

[16] Jiménez D., Lin C. *Neural Methods for Dynamic Branch Prediction*, ACM Transactions on Computer Systems, Vol. 20, Issue 4, November 2002, pp. 369-397, New York, USA.

[17] Jiménez D., *Idealized Piecewise Linear Branch Prediction*, Journal of Instruction-Level Parallelism, Vol. 7, April, 2005, pp. 1-11.

[18] Loh G. H., Jiménez D., *Reducing the Power and Complexity of Path-Based Neural Branch Prediction*, Proceedings of the 5th Workshop on Complexity Effective Design (WCED5), pp. 1-8, June 5, 2005, Madison, WI, USA.

[19] *http://en.wikipedia.org/wiki/Design_pattern*

[20] Burger D., Austin T., *The SimpleScalar Tool Set*, Version 2.0, University of Wisconsin Madison, USA, Computer Science Department, Technical Report no. 1342, June, 1997.

[21] Bowers K. R., Kaeli D., *Characterizing the SPEC JVM98 benchmarks on the Java virtual machine*, Technical Report, Northeastern University, ECE Department, Computer Architecture Group, 1998, pp. 1-20, Boston, Massachusetts, USA.

[22] Florea A., *The dynamic values prediction in the next generation microprocessors (in Romanian)*, MatrixRom Publishing House, 2005, Bucharest.

[23] Tao Li, Lizy K. John, *Adapting Branch-Target Buffer to Improve the Target Predictability of Java Code*, ACM Transactions on Architecture and Code Optimization, Volume 2, Issue 2, June 2005, pp. 109-130.

[24] Florea A., Vinţan L., *Advanced techniques for improving indirect branch prediction accuracy*, Proceedings of 19th European Conference on Modelling and Simulation, June 2005, pp. 750-759, Riga, Latvia.

[25] Florea A., Radu C., Calborean H., Crapciu A., Gellert A., Vinţan L., *Designing an Advanced Simulator for Unbiased Branches Prediction*, Proceedings of 9th International Symposium on Automatic Control and Computer Science, ISSN 1843-665X, Iasi, 2007.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 41-56

Flatness-based Control and Conventional RST Polynomial Control of a Thermal Process

Hajer Gharsallaoui, Mounir Ayadi, Mohamed Benrejeb, Pierre Borne

Hajer Gharsallaoui
École Nationale d'Ingénieurs de Tunis, LAboratoire de Recherche en Automatique (LA.R.A.)
B.P. 37, le Belvédère, 1002 Tunis, Tunisia
Laboratoire d'Automatique, Génie Informatique et Signal (LAGIS), École Centrale de Lille
Avenue Paul Langevin BP 48 59651 VILLENEUVE D'ASCQ CEDEX, France
E-mail: hajer.gharsallaoui@voila.fr

Abstract: In this paper, a comparison between conventional RST polynomial control by poles placement and RST flatness-based control is proposed. These approaches were developed, in discrete-time formalism by carrying out a generation of a reference starting from a flat output and by then analyzing the tracking error in closed-loop. The case of the thermal system, that we applied an output disturbance, was considered by simulation to study the effectiveness of given flatness-based robust controller with the integration of an anti-windup device, in terms of tracking trajectory and disturbance rejection.
Keywords: Flatness, RST control, tracking trajectory, robustness, anti-windup.

1 Introduction

The flatness property was introduced by M. Fliess, J. Lévine, P. Martin and P. Rouchon in 1992 to propose a new strategy to control continuous nonlinear systems with good performances in term of tracking trajectory. At first, the use of this property consists in the definition of an output trajectory allowing the determination of the variables of the flat system. Secondly, it concerns the elaboration of the control in closed- loop allowing to obtain a stable system giving place to a tracking of a desired trajectory with an error which tends asymptotically towards zero [1, 2, 5, 18, 19].

Several techniques of control resulting from the theory of control were established and applied to real systems, especially the RST polynomial control. The exploitation of robust control with an argued choice for its design satisfying the control objectives in term of the tracking trajectory of reference as well as the regulation in the presence of disturbances, constitute the object of this paper.

A comparison between this well-known conventional RST polynomial control and a recent approach of a RST controller design based on the property of flatness of the mono-variable linear and controllable systems are carried out by considering the case of the thermal process control.

2 Conventional RST polynomial control

2.1 Structure of RST controller by poles placement

In the RST control approach it is possible to impose poles in closed-loop and to carry out in a separate way the objectives of tracking and regulation. In a discrete formalism, the blocks of RST controller are given by figure 1, [4, 9, 12].

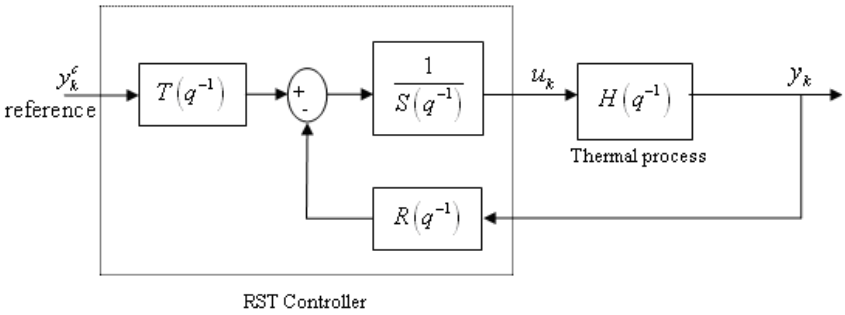


Figure 1: Conventional structure RST controller

$H(q^{-1}) = \dfrac{B(q^{-1})}{A(q^{-1})}$ is the transfer function of the dynamic linear discrete system where the polynomials A and B represent respectively the denominator and the numerator of the transfer function with respective degrees n_A and n_B:

$$A\left(q^{-1}\right) = a_0 + a_1 q^{-1} + ... + a_{n_A} q^{-n_A} \tag{1}$$

$$B\left(q^{-1}\right) = b_0 + b_1 q^{-1} + ... + b_{n_B} q^{-n_B} \tag{2}$$

The choice of the polynomials R, S and T allows to solve as well the problem of regulation as well as of tracking. These polynomials are given by:

$$R\left(q^{-1}\right) = r_0 + r_1 q^{-1} + ... + r_{nR} q^{-n_R} \tag{3}$$

$$S\left(q^{-1}\right) = s_0 + s_1 q^{-1} + ... + s_{nS} q^{-n_S} \tag{4}$$

$$T\left(q^{-1}\right) = t_0 + t_1 q^{-1} + ... + t_{nT} q^{-n_T} \tag{5}$$

where $n_R = deg(R)$, $n_S = deg(S)$ and $n_T = deg(T)$. The transfer function in closed-loop, in this case, is given by:

$$H_{BF}\left(q^{-1}\right) = \frac{T(q^{-1})B(q^{-1})}{A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1})} \tag{6}$$

The calculation of the polynomials R and S is carried out starting from the choice of the polynomial P:

$$\begin{aligned} P(q^{-1}) &= A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1}) \\ &= p_0 + p_1 q^{-1} + \cdots + p_{n_p} q^{-n_p} \end{aligned} \tag{7}$$

If $\deg(P) < \deg(A) + \deg(B)$, this equation is regular and it can be written in the following matrix form:

$$\underbrace{\begin{pmatrix} a_0 & 0 & \dots & 0 & b_0 & 0 & \cdots & 0 \\ a_1 & a_0 & \ddots & \vdots & b_1 & b_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 & \vdots & \ddots & \ddots & 0 \\ a_{n_A} & \vdots & \ddots & a_0 & b_{n_B} & \vdots & \ddots & b_0 \\ 0 & \ddots & \vdots & a_1 & 0 & \ddots & \vdots & b_1 \\ \vdots & \ddots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & a_{n_A} & 0 & \cdots & \cdots & b_{n_B} \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} s_0 \\ \vdots \\ s_{n_S} \\ r_0 \\ \vdots \\ \vdots \\ r_{n_R} \end{pmatrix}}_{x} = \underbrace{\begin{pmatrix} p_0 \\ p_1 \\ \vdots \\ p_{n_P} \\ 0 \\ \vdots \\ 0 \end{pmatrix}}_{M_P} \tag{8}$$

where $M \in \Re^{(n_A \times n_B)} \times \Re^{(n_A \times n_B)}$ is called Sylvester matrix. The coefficients of $R(q^{-1})$ and $S(q^{-1})$ contained in vector x are given by:

$$x = M^{-1} M_P \tag{9}$$

For the determination of the coefficients of the polynomial T, in the case of the tracking of a desired trajectory y_k^c corresponding to a reference model:

$$H_m\left(q^{-1}\right) = \frac{B_m(q^{-1})}{A_m(q^{-1})} \tag{10}$$

we choose $T(q^{-1}) = \beta B(q^{-1})$, to obtain:

$$H_{BF_1}\left(q^{-1}\right) = \beta B\left(q^{-1}\right) \tag{11}$$

$$H_{BF}\left(q^{-1}\right) = \beta \frac{B(q^{-1}) B_m(q^{-1})}{A_m(q^{-1})} \tag{12}$$

with taking $\beta = \frac{1}{B(1)}$, a static gain in closed-loop equal to 1 is then imposed.

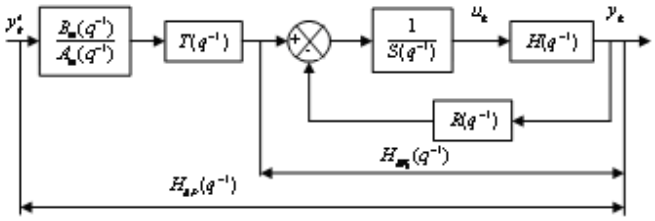


Figure 2: Detailed structure of RST regulator

2.2 Rejection of disturbances

Let us consider $S(q^{-1})$ the polynomial of the following form:

$$S\left(q^{-1}\right) = \left(1 - q^{-1}\right)^m S'\left(q^{-1}\right) \tag{13}$$

To reject a step disturbance, we can choose $m = 1$ with $\deg(S') = \deg(S) - m$.

2.3 Anti-saturation of the control

The direct implementation of RST controller often leads to a control signal whose amplitude can be very important. In order to keep this amplitude in an acceptable interval $[u_{\min}, u_{\max}]$, we add to the structure of control an anti-windup device [4], d is the output disturbance and $S^*(q^{-1})$ is a polynomial defined from the relation bellow:

$$S\left(q^{-1}\right) = s_0 + q^{-1} S^*\left(q^{-1}\right) \tag{14}$$

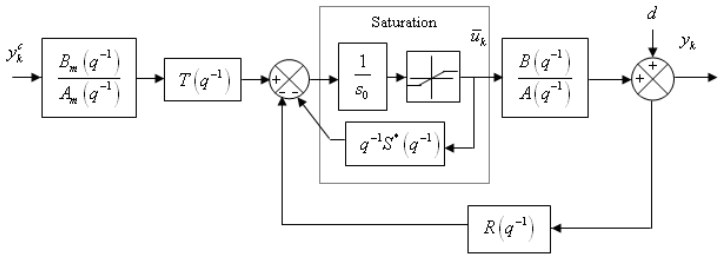


Figure 3: Conventional structure of RST controller with anti-windup device

3 Design of flatness-based controller

3.1 Determination of flat output

In this part, let us consider the discrete-time linear system Single Input Single Output (SISO), described by the following equation:

$$A\left(q^{-1}\right) y_k = B\left(q^{-1}\right) u_k \tag{15}$$

where y_k and u_k represent the output and the control signals respectively.

The flat output is a variable defined in continuous time; we consider the expressions of the polynomials $A\left(q^{-1}\right)$ and $B\left(q^{-1}\right)$ in term of the operator q^{-1}, to be able to define thereafter the discrete flat output of the system.

The system is considered as linear and controllable, consequently it is flat and its flat output z_k is given by [2]:

$$z_k = N\left(q^{-1}\right) y_k + D\left(q^{-1}\right) u_k \tag{16}$$

where $N\left(q^{-1}\right)$ and $D\left(q^{-1}\right)$ are polynomials which represent the solutions of the equation of Bezout:

$$A\left(q^{-1}\right) D\left(q^{-1}\right) + B\left(q^{-1}\right) N\left(q^{-1}\right) = 1 \tag{17}$$

The flat output z_k on which depend the output y_k and the control u_k, can be also seen as being the partial state of a linear system [3]:

$$u_k = A\left(q^{-1}\right) z_k \tag{18}$$

$$y_k = B\left(q^{-1}\right) z_k \tag{19}$$

The knowledge of the flat output z_k allows to determine the open loop control u_k of the system.

3.2 Trajectories planning

The objective of the trajectories planning [17] is to determine an open-loop control $u^d(t)$, carrying out the objective bringing a given system, of a certain initial state in a known final state:

$$u^d(t) = \mathrm{B}\left(z^d(t), ..., z^{d(\alpha+1)}(t)\right) \tag{20}$$

$$y^d(t) = \mathrm{C}\left(z^d(t), ..., z^{d(\sigma)}(t)\right) \tag{21}$$

z^d is the desired trajectory for the flat output that is $\sup(\alpha+1,\sigma)$ time continuously derivable.

If the objective is to reach two equilibrium points of balance: $(u(t_0),y(t_0),z(t_0))$ and $(u(t_f),y(t_f),z(t_f))$, where t_0 and t_f are the two times known in advance, it is necessary to solve the problem of trajectory generation offline. A solution for the calculation of $Z^d(t)$, vector of $z^d(t)$ and its successive derivatives, is given by the relation:

$$Z^d(t) = M_1(t-t_0)c_1 + M_2(t-t_0)c_2 \tag{22}$$

where $M_1(t)$ and $M_2(t)$ are the two following matrices:

$$M_1(t) = \begin{pmatrix} 1 & t & \cdots & \frac{t^{n-1}}{(n-1)!} \\ 0 & 1 & \cdots & \frac{t^{n-2}}{(n-2)!} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 \end{pmatrix} \tag{23}$$

$$M_2(t) = \begin{pmatrix} \frac{t^n}{n!} & \frac{t^{n+1}}{(n+1)!} & \cdots & \frac{t^{2n-1}}{(2n-1)!} \\ \frac{t^{n-1}}{(n-1)!} & \frac{t^n}{n!} & \cdots & \frac{t^{2n-2}}{(2n-2)!} \\ \vdots & \ddots & \ddots & \vdots \\ t & \cdots & \cdots & \frac{t^n}{n!} \end{pmatrix} \tag{24}$$

The vectors c_1 and c_2 can be defined from 20 and 21 by:

$$c_1 = Z^d(t_0) \tag{25}$$

$$c_2 = M_2^{-1}(t_f-t_0)\left(Z^d(t_f) - M_1(t_f-t_0)Z^d(t_0)\right) \tag{26}$$

In the discrete-time formalism, the real output of system y_k to be controlled, follows the desired trajectory y_k^d expressed by:

$$y_k^d = B\left(q^{-1}\right)z_k^d \tag{27}$$

The synoptic diagram of figure 5 summarizes the different steps as well as the necessary tools to generate the desired trajectory y_k^d in a discrete-time formalism with considering the sampling period T_e [2, 5, 12].

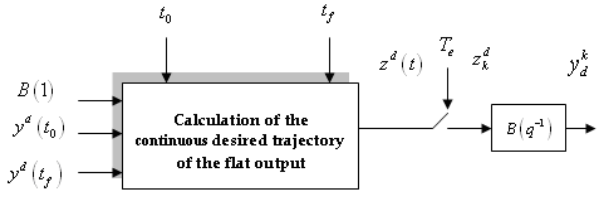


Figure 4: Generation of the desired trajectory y_k^d

3.3 Flatness-based RST polynomial controller

The realization of flatness RST controller is carried out here by considering the method of direct calculation of the state vector $Z_k = \begin{pmatrix} z_k & z_{k+1} & \cdots & z_{k+n-1} \end{pmatrix}^T$, suggested in [2, 5].

From the relations (20) and (21), we can determine the representation of state of the system in its controllable form:

$$\begin{cases} Z_{k+1} = \mathbf{A}Z_k + \mathbf{B}u_k, \\ y_k = \mathbf{C}Z_k, \end{cases} \tag{28}$$

with:

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ 0 & \ddots & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 0 & 1 \\ -a_0 & -a_1 & \cdots & -a_{n-2} & -a_{n-1} \end{pmatrix}, \mathbf{B} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} \text{ and}$$

$\mathbf{C} = \begin{pmatrix} b_0 & b_1 & \cdots & b_{n-1} \end{pmatrix}$

From this representation, we can write [5]:

$$Z_k = O^{-1}_{(\mathbf{A},\mathbf{C})} \left(Y_k - M_{(\mathbf{A},\mathbf{B},\mathbf{C})} U_k \right) \tag{29}$$

with the observability matrix is:

$$O_{(\mathbf{A},\mathbf{C})} = \begin{pmatrix} \mathbf{C} \\ \mathbf{CA} \\ \vdots \\ \mathbf{CA}^{n-1} \end{pmatrix} \tag{30}$$

and the controllability matrix $M_{(\mathbf{A},\mathbf{B},\mathbf{C})}$ given by:

$$M_{(\mathbf{A},\mathbf{B},\mathbf{C})} = \begin{pmatrix} 0 & \cdots & \cdots & 0 \\ \mathbf{CB} & \ddots & & \vdots \\ \mathbf{CAB} & \mathbf{CB} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ \mathbf{CA}^{n-2}\mathbf{B} & \cdots & \mathbf{CAB} & \mathbf{CB} \end{pmatrix} \tag{31}$$

From [5], we obtain the flatness control law by the following relation [12]:

$$u_k = K(q) z_k^d + (a-k) Z_k \tag{32}$$

where a and k are two constant vectors constituted by the a_i and k_i coefficients of the $A(q^{-1})$ and $K(q)$ polynomials given by: $a = \begin{pmatrix} a_0 & a_1 & \ldots & a_{n-1} \end{pmatrix}$, $k = \begin{pmatrix} k_0 & k_1 & \ldots & k_{n-1} \end{pmatrix}$, where $K(q)$ is a polynomial containing the closed-loop poles. The realisable structure of RST controller by flatness with q^{-1} operator can be then obtained by:

$$S\left(q^{-1}\right) u_k = K(q) z_k^d + R\left(q^{-1}\right) y_k \tag{33}$$

where:

$$R\left(q^{-1}\right) = -(a-k)\, \mathbf{A}^{n-1} O^{-1}_{(\mathbf{A},\mathbf{C})} Q \tag{34}$$

$$S\left(q^{-1}\right) = 1 + (a-k) \left(\mathbf{A}^{n-1} O^{-1}_{(\mathbf{A},\mathbf{C})} M_{(\mathbf{A},\mathbf{B},\mathbf{C})} - \begin{pmatrix} \mathbf{A}^{n-2}\mathbf{B} & \ldots & \mathbf{B} \end{pmatrix} \right) Q^* \tag{35}$$

and:

$Q = \begin{pmatrix} q^{-(n-1)} & q^{-(n-2)} & \ldots & q^{-1} & 1 \end{pmatrix}^T$, $Q^* = \begin{pmatrix} q^{-(n-1)} & q^{-(n-2)} & \ldots & q^{-1} \end{pmatrix}^T$.

The dynamics of the closed-loop is defined by the tracking polynomial $K(q^{-1})$ such as:

$$A\left(q^{-1}\right)S\left(q^{-1}\right)+B\left(q^{-1}\right)R\left(q^{-1}\right)=K\left(q^{-1}\right) \tag{36}$$

This relation represents the Bezout equation already defined in the traditional approach of RST controller. The choice of poles must be well optimized in order to satisfy the desired performances. This step appears the fundamental contribution of the exploitation of the flatness property in the design of such controller.

Indeed, the choice of the closed-loop poles corresponds to this of the tracking model of a desired trajectory and more precisely, to the poles of the $K(q)$ polynomial.

4 Study of the robustness

4.1 Disturbances rejection and noises attenuation

By using the principle of the internal model, a deterministic disturbance can be rejected by considering in the transfer function of the direct chain an integrator. In fact, in order to obtain a robust controller, the polynomial $S(q^{-1})$ must have a term in $(1-q^{-1})$ which allows the rejection of static disturbance present on the output signal. Moreover, to introduce a filtering effect into a certain zone of frequencies, the principle of the blocking of the signal is often used by forcing the system to behave as an open loop system at these frequencies; what is equivalent to impose of the zeros in polynomial $R(q^{-1})$. This led so that polynomial $R(q^{-1})$ contains a term of $(1+q^{-1})$ making it possible to ensure an attenuation of the noises effect in high frequencies on the system input [4, 6].

Thus, the following polynomials $H_S(q^{-1})$ and $H_R(q^{-1})$ are introduced into the model of the system:

$$H_S\left(q^{-1}\right)=1-q^{-1} \tag{37}$$

$$H_R\left(q^{-1}\right)=1+q^{-1} \tag{38}$$

By taking in account of these pre-specified fixed parts, polynomials $\tilde{R}(q^{-1})$ and $\tilde{S}(q^{-1})$ of new RST controller can be expressed by:

$$\tilde{R}\left(q^{-1}\right)=H_R\left(q^{-1}\right)R\left(q^{-1}\right) \tag{39}$$

$$\tilde{S}\left(q^{-1}\right)=H_S\left(q^{-1}\right)S\left(q^{-1}\right) \tag{40}$$

In addition, the dynamics of the regulation is defined by the choice of the dominant and auxiliary poles of polynomial $K(q^{-1})$ of the closed loop given by the relation:

$$A\left(q^{-1}\right)H_S\left(q^{-1}\right)S\left(q^{-1}\right)+B\left(q^{-1}\right)H_R\left(q^{-1}\right)R\left(q^{-1}\right)=K\left(q^{-1}\right) \tag{41}$$

In order to take in account the pre-specified parts in the design of the new controller, it is enough to consider the increased model governed by the following transfer function:

$$\tilde{H}\left(q^{-1}\right)=\frac{B(q^{-1})H_R(q^{-1})}{A(q^{-1})H_S(q^{-1})} \tag{42}$$

Then, it is a question to redo the calculations of flatness-based RST controller using the method presented previously.

4.2 Analyze of the controller robustness

In order to maintain nominal performances in rejection of disturbances and in the presence of modelling errors, the sensitivity functions defined below, are calibrated and recomputed so as to satisfy the performances required. The sensitivity functions are given according to the type of disturbance to consider. By considering the three types of following disturbances: output-input disturbances and noises of measurement, we can thus deduce the output-input functions of sensitivity described in [4, 6]:

- Output disturbance: represented by the transfer function $S_{yd}(q^{-1})$ between the output disturbance $d(t)$ and the system output $y(t)$:

$$S_{yd}\left(q^{-1}\right) = \frac{A(q^{-1})\tilde{S}(q^{-1})}{A(q^{-1})\tilde{R}(q^{-1}) + B(q^{-1})\tilde{S}(q^{-1})} = \frac{A(q^{-1})\tilde{S}(q^{-1})}{K(q^{-1})} \tag{43}$$

- Input disturbance: represented by the transfer function $S_{ud}(q^{-1})$ between the output disturbance $d(t)$ and the input $u(t)$:

$$S_{ud}\left(q^{-1}\right) = \frac{-A(q^{-1})\tilde{R}(q^{-1})}{K(q^{-1})} \tag{44}$$

The study of the robustness of the developed RST controller is based on the frequencies analysis of the modules of the various functions of sensitivity. Desirable templates for the sensitivity functions will be defined through the constraints of imposed performances and tolerated robustness margins. The purpose is to have the disturbance output sensitivity function inside the upper and lower templates as shown further in figures 14 and 15. Also, input sensitivity function must have a decreasing shape meaning that the controller is insensitive to noise.

5 Saturation effects

The additive static disturbances cause generally the increase in the amplitudes of the control signal applied to the system exceeding certain limit values. This especially presents a problem in the control of process by digital computers having thresholds in tension not to exceed on the level of their input-outputs. Consequently, it is necessary to design a device of anti-saturation according to the technique already developed in [4], where the law of control has the following forms:

$$u_k = K(q)\, z_k^d - \tilde{R}\left(q^{-1}\right) y_k - \bar{S}\left(q^{-1}\right) u_{k-1} \tag{45}$$

with:

$$\tilde{S}\left(q^{-1}\right) = 1 + q^{-1}\bar{S}\left(q^{-1}\right) \tag{46}$$

In addition, by considering $u_{\min}$ and $u_{\max}$ respectively the lower and higher limits of the control saturation, we obtain:

$$\bar{u}_k = \begin{cases} u_k \quad if\ u_{\min} \leq u_k \leq u_{\max} \\ u_{\max}\ if\ u_k > u_{\max} \\ u_{\min}\ if\ u_k < u_{\min} \end{cases} \tag{47}$$

However, it is possible to impose certain dynamics when the system leaves the saturation. This is illustrated in figure 5. The desired dynamics is defined by the polynomial $P_S(q^{-1})$ given by equation (48).

$$P_S\left(q^{-1}\right) = 1 - \exp\left(\frac{T_e}{\tau_{sat}}\right) q^{-1} \tag{48}$$

where τ_{sat} indicates a time-constant of a system of first order. It is it should be noted that outside of saturation, $u_{\min} < u_k < u_{\max}$, this mechanism of anti-saturation is equivalent to block $\frac{1}{\tilde{S}(q^{-1})}$ of RST controller given by figure 5.

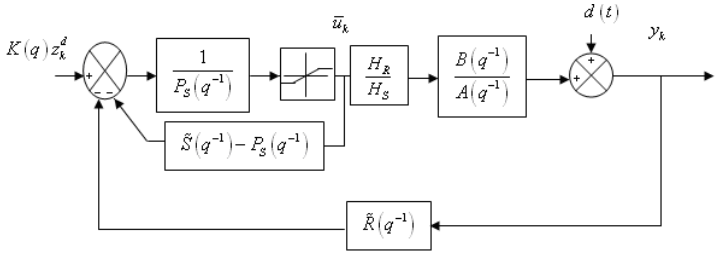


Figure 5: Diagram of flatness-based RST controller with taking in consideration of anti-saturation

Finally, the new control law of the flatness-based polynomial controller in the presence of saturation (47) is given by the expression:

$$P_S\left(q^{-1}\right)u_k = K\left(q\right)z_k^d - \tilde{R}\left(q^{-1}\right)y_k - \left(\tilde{S}\left(q^{-1}\right) + \exp\left(\frac{T_e}{\tau_{sat}}\right)\right)\bar{u}_{k-1} \tag{49}$$

In the following part, a comparative study between the two laws of RST polynomial controllers, RST flatness-based control and conventional RST polynomial control, is carried out by digital simulation by considering the case of the thermal process control.

6 Application to the control of a thermal system

6.1 Modelling of the thermal system

The thermal process whose simplified diagram is given by figure 6, is constituted of a tube of constant volume $V\,[m^3]$ and of a heating resistance R_c connected to a direct current power supply $u(t)$ applied to resistance for heating the air entering at a desired temperature by Joule effect 6, $C\,[J.m^{-3}.^{\circ}K^{-1}]$ is the specific heat constant of air, $T_E\,[^{\circ}K]$ the ambient temperature, $f_j\,[m^3.s^{-1}]$ the air rate flow entering according to the aperture of the valve j. The purpose of the control approach is to regulate the temperature $T_S\,[^{\circ}K]$ of the outgoing air at the constant temperature, given that the air flows into the tube with an initial temperature $T_E\,[^{\circ}K]$ and at the flow rate $f_j\,[m^3.s^{-1}]$.

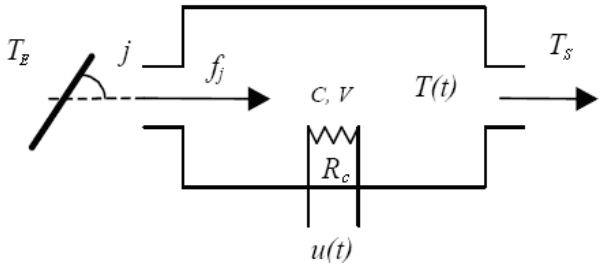


Figure 6: Simplified diagram of the thermal process

The flow rate signal is assumed piecewise constant and can be vary by changing the throttle position j.

The model of the thermal process, dependent, primarily of the valve position, if it is admitted that the temperature of the entering air remains constant, the transfer function of the model is given by [10, 16]:

$$G(p) = \frac{ke^{-\tau p}}{1 + T_c p} \tag{50}$$

where k is the delay, T is the time-constant of the process and p is the Laplace operator. For an ambient temperature equal to $20C$, the identified parameters of $G(p)$ are: $k = 0.86$, $\tau = 0.27s$ and $T_c = 0.49s$ [10, 11, 16].

The corresponding discrete-time transfer function is then given by:

$$G\left(q^{-1}\right) = \frac{0.0510q^{-1} + 0.3427q^{-2}}{1 - 0.5421q^{-1}} \tag{51}$$

T_e is the sampling period selected equal to $0.3s$.

6.2 Determination of the desired trajectory

Let us consider that$B(1) = 0.7875$, we choose to generate the trajectory expressed in continuous time $z^d(t)$ according to the following polynomial form:

$$z^d(t) = \begin{cases} \frac{y^d(t_0)}{B(1)}, & if\ 0 \le t \le t_0 \\ Poly(t), & if\ t_0 \le t \le t_f \\ \frac{y^d(t_f)}{B(1)}, & if\ t \ge t_f \end{cases} \tag{52}$$

By taking the transition times $t_0 = 10s$ and $t_f = 20s$, and choosing the polynomial $Poly(t)$, like reference trajectory between these two moments:

$$Poly(t) = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} (M_1(t - t_0)c_1 + M_2(t - t_0)c_2) \tag{53}$$

The desired trajectory z^d calculated is represented on figure 7.

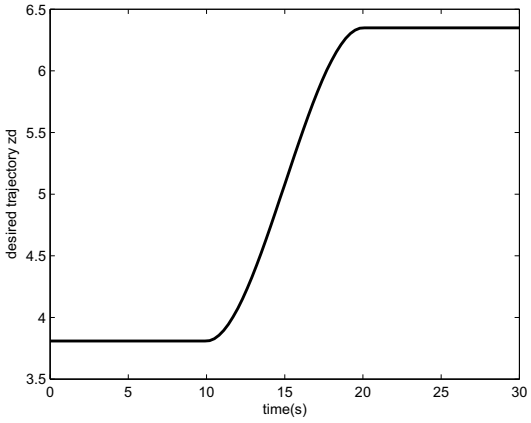


Figure 7: Desired trajectory $z^d(t)$

6.3 Conventional RST controller by poles placement

The tracking polynomial $K(q)$ is obtained by discretizing of the continuous model as third order system formed by the setting in cascade of two subsystems of which one is of second order, characterized

by $\omega_n = 2.5\,rad.s^{-1}$ and $\xi = 0.7$, and the other first order one having a time-constant $\tau = 0.2s$. It becomes then:

$$K(q) = q^3 - 1.241q^2 + 0.577q - 0.07808 \tag{54}$$

Simulations were carried out by considering the following conventional RST controller polynomials:

$$R\left(q^{-1}\right) = 2.528 - 3.216q^{-1} + 1.015q^{-2} \tag{55}$$

$$S\left(q^{-1}\right) = 1 + 0.1724q^{-1} - 0.5306q^{-2} - 0.6419q^{-3} \tag{56}$$

$$T\left(q^{-1}\right) = \frac{K(q^{-1})}{B(1)} = -0.09915 + 0.7326q^{-1} - 1.575q^{-2} + 1.27q^{-3} \tag{57}$$

The simulation results show that the response of the thermal process presents an important tracking error. The thermal process presenting one zero unstable it is thus with no minimum of phase that explain the response of the system given in figure 8.

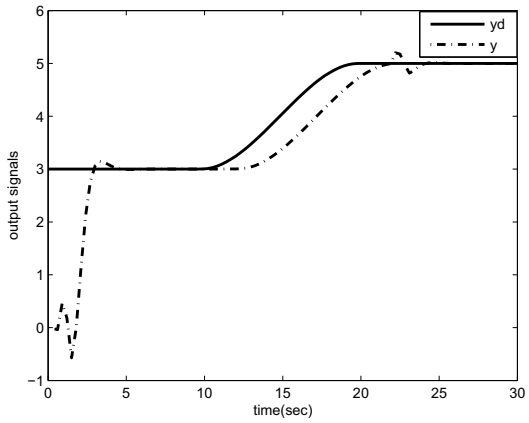


Figure 8: Conventional RST control: y the real system output and y^d the desired output

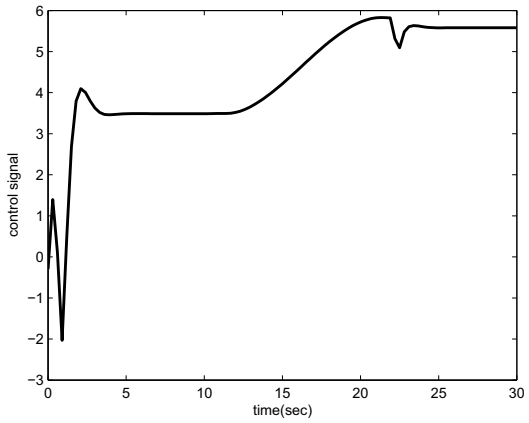


Figure 9: Conventional RST control: Control signal

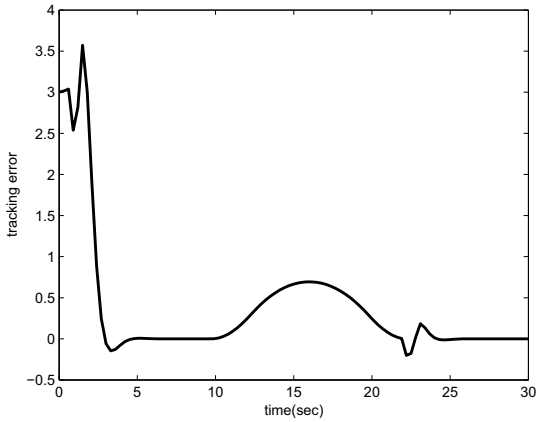


Figure 10: Conventional RST control: Tracking error

6.4 Flatness-based RST controller

The tracking polynomial $K(q)$ is selected starting from the discretization of the continuous tracking model considered as third order system. Consequently, we deduce the flatness-based RST regulator polynomials designed by flatness are bellow:

$$R\left(q^{-1}\right) = 0.6554 - 0.3274q^{-1} \tag{58}$$

$$S\left(q^{-1}\right) = 1 + 0.2681q^{-1} + 0.2069q^{-2} \tag{59}$$

leading to the results of figures 11, 12 and 13.

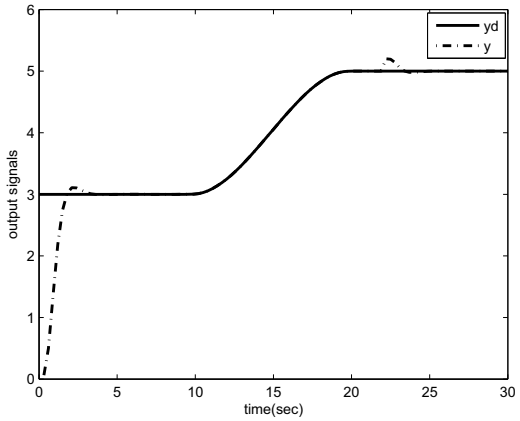


Figure 11: Flatness-based RST control: y the real system output and y^d the desired output

The results in figures 11, 12 and 13 underline the importance of the implementation of flatness-based controller in term of trajectory tracking. In fact, we notice in addition that the tracking error in the case of first approach RST is significant compared with that obtained by flatness-based controller. Thus, the flatness property is very interesting in terms of planning and tracking trajectory exploited for the development of a closed-loop system with high performances.

The results of figures 14 and 15 show that the controller obtained is robust considering the sensitivity functions which remain inside the specified templates of robustness [4]. In addition, we underline the

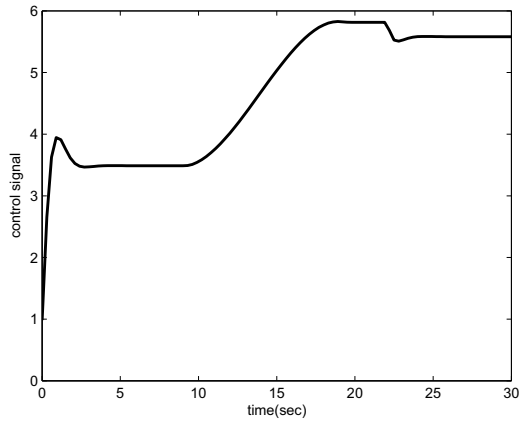


Figure 12: Flatness-based RST control: Control signal

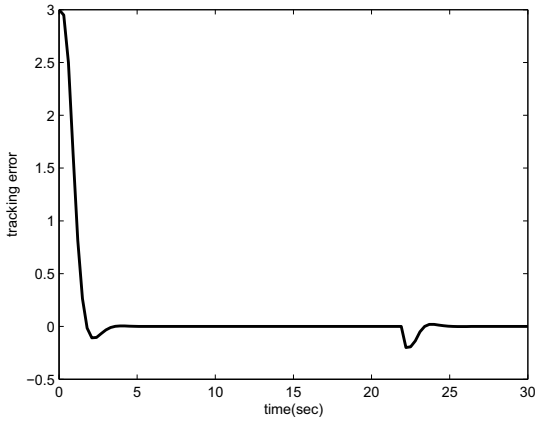


Figure 13: Flatness-based RST control: Tracking error

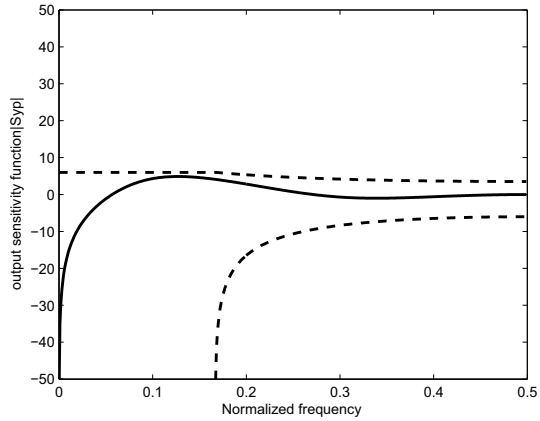


Figure 14: Output sensitivity function $|Syd|$ case of output disturbance

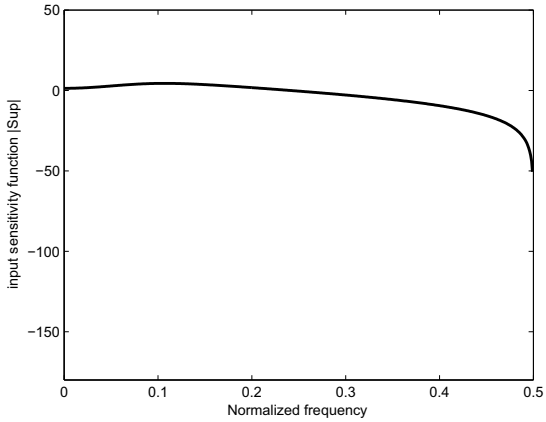


Figure 15: Input sensitivity function $|Sud|$ case of input disturbance

robustness of such controller related to the static disturbance and the high frequency noises rejection is clearly guaranteed. Moreover, the input sensitivity function presents attenuation in the high frequencies which define an elimination of noises in the input.

7 Conclusion

In conclusion, the study of the thermal process control shows that flatness-based RST regulator ended in a regulation of the robust and powerful type RST in term of tracking trajectories compared with such obtained by the use of the conventional polynomial RST controller. The design of flatness-based RST controller, depending on the choice of the poles of the tracking dynamic system and on the desired trajectory, allowed the satisfaction of the templates imposed by the typical values of the robustness margins.

Bibliography

[1] M. Fliess, J. Lévine , Ph. Martin and P. Rouchon, Sur les systčmes non linéaires différentiellement plats, *Compte Rendu de l'Académie des Sciences de Paris*, Série I, 315, pp. 619-624, 1992.

[2] M. Ayadi, *Contribution ŕ la commande des systčmes linéaires plats de dimension finie*, PhD Thesis, Institut National Polytechnique de Toulouse, 2002.

[3] T. Kailath, *Linear systems*, Prentice Hall, 1980.

[4] I. Landau , R. Lozano and M. M'Saad, *Adaptative control*, Springer-Verlag, London, 1998.

[5] F. Rotella, F J. Carrillo and M. Ayadi, Polynomial controller design based on flatness, *Kybernetica Special Issue on System Structure and Control*, Vol. 38, n°5, pp. 571-584.

[6] A. Oustaloup, *La robustesse. Analyse et synthčse de commandes robustes*, Hermčs, 1994.

[7] M. Fliess and R. Marquez, Towards a module theoretic approach to discrete time linear predictive control, *14th International Symposium on Mathematical Theory of Networks and Systems, MTNS'2004*, Perpignan, 2001.

[8] M. M'Saad M. and J. Chebassier, Commande adaptative des systčmes, *Techniques de l'Ingénieur*, Vol. S2, n°S7426, pp. 1-25, Paris, 1999.

[9] P. Borne, G. Dauphin-Tanguy , J. P. Richard, F. Rotella and I. Zambettakis, *Commande et optimisation des processus*, Edition Technip, Paris, 1990.

[10] S. Médar, *Supervision et reconfiguration de la commande des systčmes dynamiques en présence de variations dans les conditions de fonctionnement*, PhD Thesis, Institut National Polytechnique de Toulouse, 2002.

[11] F. Rotella, F. J. Carrilo and M. Ayadi, Digital flatness-based robust controller applied to a thermal process, *IEEE International Conference on Control Application*, pp. 936-941, Mexico, 2001.

[12] K. J. Aström and B. Wittenlmark, *Adaptative Control*, Addison Wesley, United States, 1989.

[13] M. Ayadi, N. Langlois, M. Benrejeb and H. Chafouk, Flatness-based robust adaptative polynomial controller for a diesel engine model, *Transaction on Systems, Signals & Devices*, Vol. 2, n°1, pp. 71-90, 2006.

[14] M. Fliess, J. Lévine, Ph. Martin and P. Rouchon, On Differentially Flat non Linear Systems, *IFAC-Symposium NOLCOS'92*, pp. 408-412, Bordeaux ,1992.

[15] R. Marquez, and M. Fliess, Linear Predictive Control Revisited. A Flatness Based Approach, *European Control Conference, ECC'99*, Karlshrule, 1999.

[16] Ph. Charbonnaud, J. F. Carrilo and S. Médar, Robust control reconfiguration of a thermal process with multiple operating modes,*IEEE transaction of control systems technology*, Vol. 11, n°4, 2003.

[17] J.Y. Dieulot, I. Thimoumi, F. Colas and R. Béarée, Numerical Aspects and Performances of Trajectory Planning Methods of Flexible Axes, *International Journal of Computers, Communications and Control*, Vol.I, n°4, pp. 35-44, 2006.

[18] J. Lévine, On the necessary and sufficent conditions for differential flatness, 6^{th} *IFAC Symposium on Nonlinear Control Systems, NOLCOS'04*, pp. 463-468, Stuttgart, 2004.

[19] H. Sira-Ramirez and S.K. Agrawal, *Differentially flat systems*, Marcel Dekker, 2004.

Hajer Gharsallaoui was born in 1979 and received the Engineer Diploma degree in 2004 and the Master Degree in Automatic Control in 2005 from the Ecole Nationale d'Ingénieurs de Tunis (ENIT). She is currently preparing the Ph.D. Degree in Automatic Control within the framework of LAGIS-EC-Lille and LARA-ENIT cooperation. Her research is related to Accommodation Control and FTC Control, diagnostic of flat system, and adaptive control.

Mounir Ayadi graduated from the Ecole Nationale d'Ingénieurs de Tunis in 1998 and received his PhD degree in Automatic Control from the Institut National Polytechnique de Toulouse in 2002. He was a post-doctoral fellow at the Ecole Supérieure d'Ingénieurs en Génie Electrique de Rouen in 2003. He is currently Maître-Assistant at the Ecole Nationale d'Ingénieurs de Tunis. His research interests are in the area of control system theory, predictive and adaptive control, and flat systems.

Mohamed Benrejeb was born in Tunisia in 1950. He obtained the Diploma of "Ingénieur IDN" (French "Grande Ecole") in 1973, the Master degree of Automatic Control in 1974, the PhD in Automatic Control of the University of Lille in 1976 and the DSc of the same University in 1980. He is currently a full Professor at the Ecole Nationale d'Ingénieurs de Tunis and an invited Professor at the Ecole Centrale de Lille. His research interests are in the area of analysis and synthesis of complex systems based on classical and non conventional approaches.

Pierre Borne received the Master degree of Physics in 1967, the Masters of Electronics, of Mechanics and of Applied Mathematics in 1968. The same year he obtained the Diploma of "Ingénieur IDN" (French "Grande Ecole"). He obtained the PhD in Automatic Control of the University of Lille in 1970 and the DSc of the same University in 1976. He became Doctor Honoris Causa of the Moscow Institute of Electronics and Mathematics (Russia) in 1999, of the University of Waterloo (Canada) in 2006 and of the Polytechnic University of Bucarest (Romania). He is author or co-author of about 200 Journal articles and book chapters, and of 34 plenary lectures and of more than 250 communications in international conferences. He has been the supervisor of 71 PhD thesis and is author of 20 books . He is Fellow of IEEE and has been President of the IEEE/SMC society in 2000 and 2001 .He is Vice-Chair of the IFAC TC on Large Scale Complex Systems. He is presently Professor "de classe exceptionnelle" at the Ecole Centrale de Lille and Director of the national french pluriformations group of research in Automatic Control.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 57-64

Discussion Support System for Intra-class Discussions and the Criteria for Group Making

Ikuo Kitagaki

Hiroshima University, Research Institute for Higher Education
2-12-1 Kagamiyama, Higashi-hiroshima, 739-8512, Japan
E-mail: kitagaki@hiroshima-u.ac.jp

Abstract: A computerized system has been discussed. It assists group discussion done in a classroom in the way that, first, it presents a topic with the relevant choices, second, each student selects a choice and sends it to the server, third, the server determines the groups according to the choices and other information, forth, it sends to the students' cell phone the group information with each choice, lastly, students actually make group according to the given information then start to discuss. Relating to the system, this paper describes the configuration of the proposed computer system, two aspects of group division(difference in learning and similarity in learning), the algorithm of the group division, and the execution process of actual group discussions, assisted by this system, about specific topics.
Keywords: Group division, Discussion support system, Algorithm

1 Introduction

Computer-assisted collaborative learning and group learning have become more popular with the development of e-learning systems. The computer system which is characterized by the algorithm for dividing one class of students into several groups for group discussion is discussed (Akahori, 1997). This computer system is used for the group division, however, it is not used in the later group discussions because the actual discussions are made as traditional face-to-face communication activities. Thus the computer system can be also called the computer-assisted, group discussion support system. Concretely speaking, this system divides one class into multiple groups according to students' answers of a discussion topic by using a specific algorithm, and each student is notified of the names of the members who belong to the same group. The students then form groups according to the notified information, exchange opinions, and discuss the topic to increase understanding. As almost all students have their own cell-phones, the computer server collects information necessary from students and distributes information to students via cell-phones.

Although the configuration of the above computer system and the algorithm of the group division are presented in this paper, the criteria of the group division has not been discussed in terms of their educational characteristics. This paper deals with not only the mathematical criteria(Kitagaki, 2008) but also the educational characteristics of two criteria: *similarity in learning* and *difference in learning* both of which are exemplified. This paper also describes the execution process of actual group discussions, assisted by this system, about specific topics.

Relating the algorithm shown in this paper, we have already discussed how to make group divisions when a test problem and the answers are the topic for discussion(Kitagaki et.al., 2007). In this case, I divided a student class based on the students' answers to the test. That is, values 1 and 0 were assigned to the right and wrong answers of the test respectively, and the class was divided into groups according to these assigned values. On the other hand, there are no concepts of correct/wrong answer in discussion.

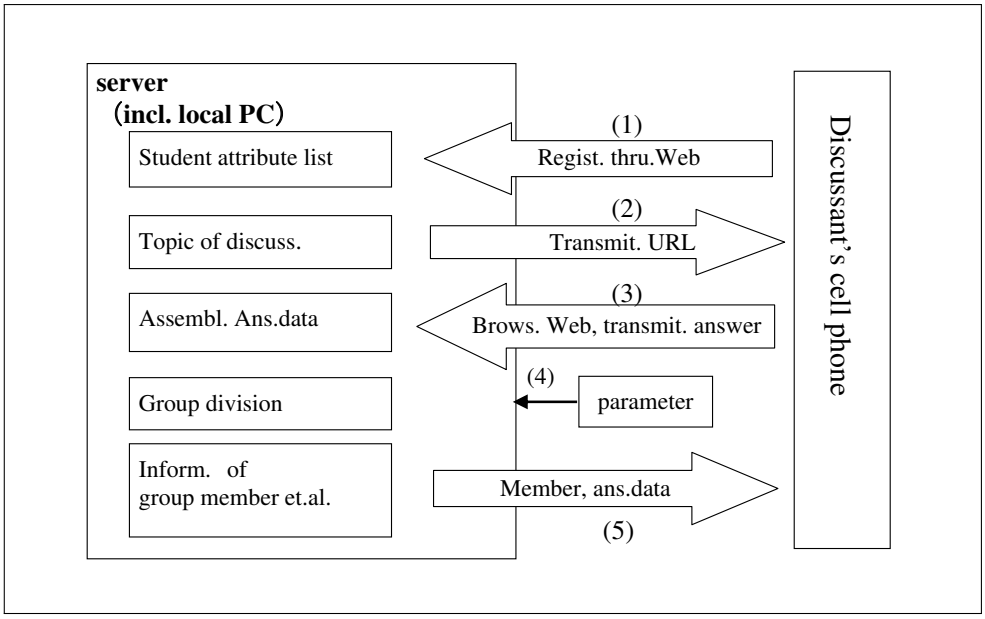


Figure 1: System configuration.

Thus the previous paper has been extended theoretically to make a revised paper this time. In the case of discussion, there could be the concepts of similar remarks and different remarks. Thus it is rational to deal with the choices so that we let them have 'distance' between 0 and 1.

2 Discussion Support System

Discussion support system is implementing the system flow shown in Figure 1.

1. To register student attribute list: As the initial process of the proposed system, it is necessary to register student's name, number, the mail address, sex and so forth. Among these, mail address is used for sending relevant URL and their names are used for students to know all the group member. These registrations are done on a Web page. The URL of the page is informed to all students in advance.

2. To determine a topic with its choices for discussion: There could be usually installed many topic for discussion. The teacher selects a topic among them. Then the computer sends the URL for browsing the topic to all discussants.

3. To gather the answer data as a choice: All students make an access to the URL mentioned in (2), then read the topic and the choices. They select a choice among the choices for the given topic then send it to the server. All the answer are gathered and stored in the server.

4. To make groups: The computer server makes the group division according to the student's answer as transaction data. The basic idea of the group division is explained in the later section. In the actual administration, the following parameters and the necessary information ought to be inputted prior to the group division: the value with which a topic is discussed.: the difference of choices in their contents.: the number of a group constituents

   When the information of group division is obtained as a processed result, it is possible for a teacher to add, to it, remarks to each group, remarks to each individual and remarks to the all.

5. To Inform each student of the group member, selected choice, teacher's remarks: The computer server sends to each student's mail address the group members, each answer and the remarks above if any.

   Through the process above, each student is informed the name of all member which belong to the same group. Then, each group gathers somewhere inside or outside of the classroom and starts to deeply consider the topic by discussion. In the actual classroom it could be that the students don't know each other, thus, in order to make the students' group making easier, it is desirable for the server to allocate a group ID to each group and send the ID with the group information mentioned above.

3 Two aspects of criteria for group making

A community, in general, is made under some sort of homogeneity: similar thought, similar culture, and so forth, leading to its stable existence. On the other hand, group making for developing creative products which may be called as a small community, often needs heterogeneity such as different idea, different situation, and so forth. Thus which is valuable criterion, homogeneity or heterogeneity, will naturally depend upon the objective, characteristics and others of making the relevant human groups. Upon the consideration, I here discuss the criteria in the case of educational objectives in a classroom.

1. *difference in learning*: I here raise the example of regarding difference in learning in a classroom. Let us suppose that students make groups and discuss the physical problem/solution in physics. In this case it is natural to think that there could be only one correct answer/solution, which will be proved provided the relevant experiment is done. Thus if there happened to be different answers in the group, it could be valuable to make*the difference* clear and discuss which idea will be correct, leading to extensive thinking a new idea. As various predictions done by the students may be converged to get a correct answer by observing the experiment for verification, fruitful discussions will be anticipated to proceed.

   As an another example, a management game in a classroom will be raised where several groups compete with each other for profit in order to survive. One group may be called as a company, which generally consists of different role of constituents. If we think those constituents ought to be selected regarding different ability/characteristics, it is important to set *difference in learning* as a criterion for making appropriate groups.

2. *similarity in learning*: If student's values is important in order to get a solution as learning, there will be a case where *similarity* is regarded in making students' groups. Let us suppose that students plan to go abroad in a group for a learning objective thus the teacher at first surveys the place which each student wants to visit. If the places proposed in a group locate near to each other in a group, it gets possible for them to visit all the places as one trip. Consequently, I can summarize that, in the case of regarding values in order to do the group based determination, the criterion of *similarity in learning* ought to be used.

4 Method of group division

In accordance with the previous section, group division using student's choice of an answer can be made by two kinds of criterion as the followings.

1. *difference in answers*: Groups are made so that choices of each member may be different from those of others as much as possible.

2. *similarity in answers* Group are made so that choices of each member are similar with those of others as much as possible.

Two criteria are reverse in their evaluation of 'goodness'. Thus it is enough only to explain criterion (a). As for the criterion (a), two methods have been proposed(Kitagaki, 1996; Kitagaki et.al., 1981). The proposed system in this material adopts the simpler method(Kitagaki et.al., 1981). The algorithm is outlined below.

topic sets: M
topic: $m_i(\in M)$
value of the topic: $v(m_i)$
group sets: G
relevant group: $g(\in G)$
bigness of group 'g': $\mid g \mid$
discussant(student): $x_j(\in g)$
selected choice: $a(m_i, x_j)$
difference of choices selected by discussant x_j
and discussant x_k : $d\{a(m_i, x_j), a(m_i, x_k)\}$
goodness of group division using criterion (a): α_g
goodness of group division using criterion (b): α_g
goodness per capita of group division: β

In the definition above, both of 'value of the topic' and "difference of choices selected by discussant x_j and discussant x_k' are the value between 0 and 1. The 'bigness of group g" is the number of a group. The number is not always the same for all group. But its algorithm is abbreviated here. All the said variables have to be determined in advance. The 'goodness per capita of group division β' is determined as the following.

$$\alpha_g = \sum_{m_i \in M} \sum_{x_j \in g} v(m_i) \bigcup_k \{d\{a(m_i, x_j), a(m_i, x_k)\}\} \tag{1}$$

$$\beta = \sum_{g \in G} \alpha_g / \sum_{g \in G} \mid g \mid \tag{2}$$

In the equation [2], group division which makes the value maximum is the optimal solution. In order to get the optimum, however, it is necessary to administrate the calculation for all the combination of groups. It is actually difficult to get it because of time for its calculation. Thus a simple method is implemented(Kitagaki et.al., 1980) as the following. Its example deals with the case that thirty discussants are divided into ten groups with three discussants each.

As the initial status, I suppose that the computer fix the discussants $x_1, \ldots, x_{30}$ as shown in 'n=1' Figure 2, and define the value of β in eq.[2] as β_1. Then I let it compare cell1 with each cell thereafter one by one. First, let it exchange cell1 x_1 for cell2 x_2 to obtain the pattern as shown in 'n=2' then get the value as $\beta_{1,2}$. It is clear that $\beta_{1,2}$ is same as β_1 in their value. Thus there is no reason to exchange thus it ought to be withdrawn. Second, it is obvious that the exchange of x_1 and x_3 leads to the same result as above. It is cell1 and cell4 that has actual meaning of exchange because they belong to different groups in the initial pattern. If β_1 is bigger than(or equal to) $\beta_{1,4}$, the computer regards the pattern of 'n=4' as not better pattern than the one of 'n=1' then the exchange ought to be withdrawn. On the other hand, if β_1 is smaller than $\beta_{1,4}$, it regards the pattern of 'n=4' better than the one of 'n=1' then the exchange ought to be withdrawn. On the other hand, if β_1 is smaller than $\beta_{1,4}$, it regards the pattern of 'n=4' better than the one of 'n=1' then the exchange ought to be done to get the new pattern. Based upon the new pattern,

it searches for a better pattern.(In the new pattern, the content of each cell is re-designated in the way that the content of cell'i' are written as xi (i=1,30). It means that the new pattern is shown as x1...x30 from the leftmost column to the rightmost column.) The search for the better pattern is succeeded in the same way. If the comparison of cell1 and cell30 has been done, cell2 becomes the base of comparison. The exchange of cell2 and cdll3 is done as shown in the pattern of 'n=31', resulting in meaning nothing, followed by the exchange of cell2 and cell4 in 'n=32'.

Consequently, the exchange of two cells are done in the following order, and as a result, the number of exchange becomes 870(=29*30) in all.(Actually the exchange of two cells in a group ought to be omitted.)

cell1 and cell2, cell1 and cell3, cell1 and cell4,cell1 and cell5, 'Ś,cell1 and cell29, cell1 and cell30
cell2 and cell3, cell2 and cell4, cell2and cell5, 'Ś,cell2 and cell29, cell2 and cell30
cell3 and cell4, cell3 and cell5, 'Ś,cell3 and cell29, cell3 and cell30
.....
cell29 and cell30

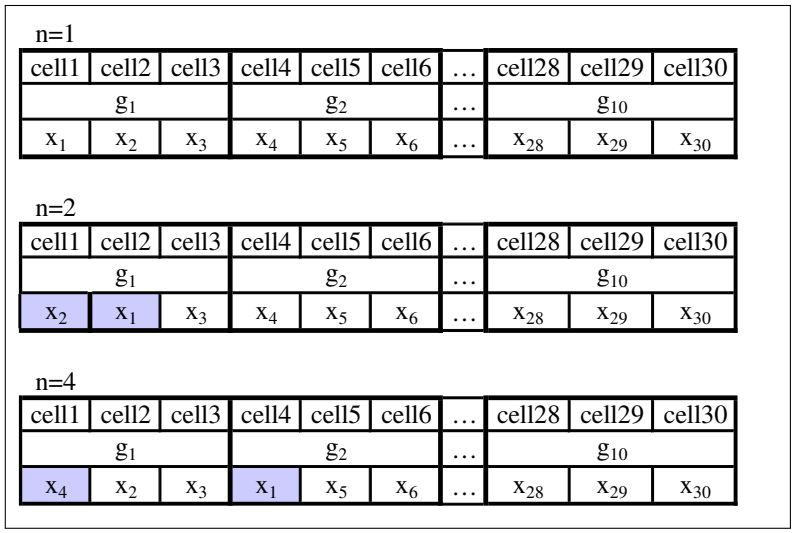


Figure 2: The exchange of two cells(in the case that a classroom consists of thirty students).

Supposing the number of discussant to be 'n', the number of the said exchange becomes 'n(n-1)'. For each exchange, the computer gets the value of β, then the optimal group division is obtained.

If method (b) shown in the beginning of this section is implemented for group division criterion, we have only to use eq.[3] instead of eq.[1].

$$\alpha_g^{'} = \sum_{m_i \in M} \sum_{x_j \in g} v(m_i) \bigcup_k \{1 - d\{a(m_i, x_j), a(m_i, x_k)\}\} \tag{3}$$

5 The administration of discussion classroom

As a topic for the proposed discussion, I raised a topic of career development to which most students might be relevant. The topic relates the consideration on the answer in an interview in job hunting. I administrated the classroom discussion four times. In every administration, the same topic was used. Two experimental administrations are discussed below.

experimental 1(E1) Fifteen Hiroshima University students served as subject(twelve undergraduate students and three graduate students). Among them, nine students were science in major., Jul. 26, 2007.

experimental 2(E2) Fourteen Hiroshima University students served as subject(seven undergraduate students and seven graduate students). Among them, nine students were science in major., Nov. 11, 2007.

Table 1: A questionnaire and the answer [6]

Q: What worker do you want to be?
A: As human is not working alone, I want to be a worker regarding communication with others in the actual work where I should say whatever necessary to say and listen to them whenever to do that .
As your remark for the underlined parts, select a choice that is nearest to it.
1. The expression is vague and fuzzy thus have little impact.
2. No fresh awareness as a new comer. can be felt.
3. Everybody can say that thus little impact are there.
4. It is the president of a company to say that. Thus, if the matter gets worse, it may give them a feeling of impoliteness.

In both of E1 and E2, I used the same room and implemented the same topic which consists of two rounds, R1 and R2, that is, two 'Question-Answer' set. But the students were different in those two experiments. In either round, the question exemplified in Table 1 was used. In its actual administration, I let the students inform others in a group of the choice which each student has selected then discuss what would be a better answer for the given answer. Lastly I let them make and write a better answer then offer it as a report with of their consensus.

As most students in a group were not acquainted with each other, it was assumed that, when the information of a group member were presented in the step of Figure 1(5), it gets difficult for them to make a group. Thus ID number proper to each group was informed all the classroom, leading to easier making groups. Addition to that, as group leaders is necessary in order to facilitate the discussion well, how to determine a leader in a group was also informed them. Discussion time was set to nearly fifteen minutes, which was also informed them as a comment.

After each administration of R1 and R2, the following questionnaires have been examined to all the students.

(Influence of selecting a choice on the discussion flow)

1. I don't think that the information of a choice selection had an influence on the relevant discussion flow. In other words, discussion flow must have been the same as the one without selecting a choice.

2. I think that the information of a choice selection had an influence on the relevant discussion flow to some extent.

3. I think that the information of a choice selection had an influence on the relevant discussion flow to great extent.

Table 2: Answer data processing of 'influence of selecting a choice'
(a)basic statistics

condition	number of students selecting a choice (%)			X	σ
	1	2	3		
$(E1, R1\&R2)$	0.30	0.57	0.13	1.90	0.65
$(E2, R1\&R2)$	0.07	0.46	0.46	2.39	0.61
$(E1\&E2, R1\&R2)$	0.19	0.52	0.29	2.10	0.69

Number of the students is fifteen 15 in E1, and fourteen in E2.
X: average, σ: standard deviation

(b)Z-test(using normalized distribution)

condition	m		
	1	2	3
(E1, R1&R2)	-7.42^*	0.86	9.13^*
(E2, R1&R2)	-11.9^*	-3.37^*	5.20^*
(E1&E2, R1&R2)	-12.2^*	-1.15	9.94^*

* Hypothesis 'X=m' has been rejected ($p < 0.01$), $Z = \dfrac{(X-m)\sqrt{N}}{\sigma}$

In R1 and R2, presented topics were the same type thus those answers were added to have been statistically processed altogether. The result is shown in Table 2(a). There is also shown the result in the case that E1 and E2 were combined in their data.

From the test result shown in the table, the average choice for the questionnaire is said to be 2(or in-between 2 and 3). It means that their average remark is that selecting a choice has an influence on the discussion flow as their awareness.

Besides the experiment shown in this material, I have done the questionnaire survey comparing criterion (a) with criterion (b) shown in the 3^{rd} section supposing the case that groups were divided according the criterion (b). As their awareness, it got obvious that they felt more fruitful in their discussion with variety of choices in a group than that with a similar choices.

6 Conclusion

This paper discussed a computerized system which assists group discussion done in a classroom. First, configuration of the system was explained. Second, two aspects of criteria for making groups, *similarity in learning* and *difference in learning* have been discussed. Third, a math. model for group division was presented. A trial of administrating a classroom discussion was done.

In its administration, a topic and the choices were presented to the students, each student selected a choice and sent it to the server, the server determined the groups, then it sent to the students the group information with each choice. From the administration, it got clear that the discussants have the awareness that letting them know their selected choice each other gave an influence on the further discussion flow.

Bibliography

[1] Akahori K. et.al.(1997, in Japanese): *The skill of university classroom teaching*, Daiichi-houki, Tokyo, 142-145.

[2] Kitagaki I.,Hikita A.,Takeya M., Fujihara Y.(2007): Development of an algorithm for groupware modeling for a collaborative learning, *Int'l Journal of Computers, Communication & Control*, II,1, 66-73.

[3] Kitagaki I.(1996): Evaluation of students' group using fuzzy integral, *IEICE*, J79-D-II,11, 1888-1896.

[4] Kitagaki I., Shimizu Y. and Suetake K.(1980): An instructional method which permits the students to critically discuss their own test answers, *Japan Journal of Educational Technology*, 5,1, 23-33.

[5] Kitagaki I.: Development of Group Division Algorithm And Discussion Support System for Intra-class Discussions, Proceedings of the 3^{rd} International Conference on Virtual Learning, pp.101-108, Constanta, 2008.

[6] Nakatani A.(1995, in Japanese): *Expert of interviewing*, Diamond Co., Tokyo.

Ikuo Kitagaki was born in Toyohashi, Japan, in 1947. He has received B.E., M.E. and Doctoral Degrees from Tokyo Institute of Technology in 1970, 1972 and 1981, respectively. He has worked in Tokyo Institute of Technology, The Institute of Vocational Training, and so forth. During the time, he has been involved in educational technology, literacy in science/technology, fuzzy science, human science, et.al.. He has been a member of the Editorial Advisory Board of the Advances in Web-based Learning(AWBL) Book Series(USA). He has received ICVL Excellence Award "Intel® Education" from the ICVL(the International Conference on Virtual Learning) Scientific Committee.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 65-72

Bayesian Network Classifier for Medical Data Analysis

Beáta Reiz, Lehel Csató

Beáta Reiz
Biological Research Center, Central Labs, Bioinformatics Group
62 Temesvári krt., HU-6701, Szeged, Hungary
E-mail: beareiz@brc.hu
Lehel Csató
Babeş Bolyai University, Faculty of Mathematics and Computer Science
1 Kogălniceanu str. RO-400084 Cluj-Napoca, Romania
E-mail: csatol@cs.ubbcluj.ro

Abstract: Bayesian networks encode causal relations between variables using probability and graph theory. They can be used both for prediction of an outcome and interpretation of predictions based on the encoded causal relations. In this paper we analyse a tree-like Bayesian network learning algorithm optimised for classification of data and we give solutions to the interpretation and analysis of predictions. The classification of logical – i.e. binary – data arises specifically in the field of medical diagnosis, where we have to predict the survival chance based on different types of medical observations or we must select the most relevant cause corresponding again to a given patient record.
Surgery survival prediction was examined with the algorithm. Bypass surgery survival chance must be computed for a given patient, having a data-set of 66 medical examinations for 313 patients.
Keywords: Bayesian networks, classification, medical data analysis, causal discovery.

1 Introduction

In this paper we analyse tree-like Bayesian network (BN) implementation for medical data classification. We consider a general case for data attributes, where the observations can be both continuous and discrete, and - general to almost all medical data - missing observations also can occur. We aim to establish causal relationships between variables representing medical examinations. Whilst interested in a good classification performance, we also want to interpret and analyse the predictions in terms of the encoded causal relations.

The database we used consists of 66 medical examinations of 313 people containing both discrete and continuous observations. The task is thus to predict the surgery survival chance based on the available data – the medical examinations [1, 2], and analysis of impact of a specific examination on patient survival. Partial observability characterises the database, the number of missing values is 2413.

Our aim is to predict target variables value – survival – for a particular patient and to obtain the "most relevant" variables affecting the output of the classifier. Of equal interest is to analyse the decisions in terms of encoded relationships between data attributes. This analysis is usually done to provide support for physicians. We encode the dependencies between the class variable and the observations using a tree with root node the class variable. The other attributes are inside the tree with coresponding conditional probability tables "learned" from the data-set. Finding the most appropriate structure is an extremely difficult task. We reduce the complexity of constructing the tree of *immediate causal* relationships between

class variable and observations [3]. A tree-like Bayesian network structure was inferred from the data [4], where the root of the tree is the class variable and remaining nodes are attributes. Direct causal relations between attributes and class variable were revealed in the first phase of the algorithm, constructing a Naive Bayesian network. Attribute-attribute correlations were searched based on Chow-Liu's algorithm in the second phase of the algorithm. In practical situations we also have to face the problem – general to almost all medical data – of missing observations for some patients, meaning incomplete data items; this issue can also be considered in a principled way with a Bayesian network. The paper is organised as follows: next we present the Bayesian networks, then a stochastic algorithm to extract a plausible network sructures from the data, and we also analise experimental results of applying the algorithm to real data-sets.

2 Bayesian Networks

Bayesian networks (BNs) [5] are triplets $(V,E,\mathcal{P})$, where (V,E) is a directed acyclic graph (DAG) with nodes V, edges E, and a set of probability distributions $\mathcal{P}$, called parameters, whose elements are assigned to the nodes of the graph. The nodes represent domain variables and edges mark direct causal relations between these variables.

The network encodes a joint probability distribution function representative to the domain:

$$P(X) = \prod_{i=1}^{n} P(X_i|\text{par}(X_i))$$

where n is the number of domain variables, X_i is a node from the BN and $\text{par}(X_i)$ is the set of X_i's parents. The aciclicity of the graph ensures the product to be finite.

We employed a tree-like representation for the topology of BN in order to increase efficiency in class variable estimation and interpretation. In section 3. we describe this algorithm, where we construct a tree in such a way that the root of the tree will be the class variable and the remaining nodes are attributes. Direct causal relations encoded by the BN are interpreted as the maximum of mutual respective conditional mutual information [6, 7, 8] between nodes. Now we present the necessary information theoretical concepts [9] for our algorithm.

We will use the following notations: X and Y are random variables defined on probability spaces Ω_X respective Ω_Y with corresponding distribution functions $p(x)$ respective $p(y)$. We use their joint and conditional probability functions, denoted with $p(x,y)$ and $p(x|y)$ respectively.

Information theory offers us numerical characterisation of uncertainty in domain variables. Uncertainty is measured using the information entropy of the respective variable. Information entropy can be understood as the average minimal message length that should be sent on a channel to encode the message and is defined as follows:

$$H(X) = -\sum_{x\in\Omega_X} p(x)\log p(x)$$

Mutual information is the quantity of information two random variables contain about each other, defined as:

$$I(X,Y) = \sum_{x\in\Omega_X}\sum_{y\in\Omega_Y} p(x,y)\ \log\frac{p(x,y)}{p(x)p(y)} \tag{1}$$

In next section we present a two-phase tree-like Bayesian network structure learning algorithm. The algorithm consists of an extension of Naive Bayesian structure learning algorithm with inner structure learning for finding causal relations between attributes.

3 Network topology learning

Bayesian network classification consists of emphasising the node corresponding to the class variable during inference. As an optimisation of the learning process and inference we construct the network topology such a way to optimise the efficiency of prediction and data attribute impact estimation on target variable. We set out the class variable and in the first phase we're searching for direct dependencies between attributes and class variable, this way constructing a Naive Bayesian network.

Naive Bayes classifiers [10, 11] are widely used in classification problems. They are called Naive because of the independence assumption of the attributes. Although this is strong assumption when facing real data-sets, the Naive Bayes classification is a powerful tool for its simplicity and often gives convenient results.

During the Naive Bayesian network learning process direct dependencies between class variable and attributes has to be find. Dependency relations are interpreted as class variable specifiers, so an edge from X to class variable Y means that variable X has information about class variable Y. Mutual information between class variable and attributes, conditioning on attributes already placed between direct dependencies of class variable, gives the amount of new information the respective attribute has regarding the class variable [12].

Considering the problem this solutions means that we choose some medical examinations which we place in the network and exclude the rest of the attributes. This is a strong restriction considering that some examinations are replaced by others in different hospitals. The second phase of the algorithm consists of applying Chow-Liu algorithm [13] to learn the inner structure of network and reveal attribute-attribute correlations.

The Naive Bayesian network is formed of class variable Y respective variables X directly linked to the class variable. Our next task is to place the excluded attributes in the Bayesian network. We use mutual information maximisation to discover the causal relations between attributes from the network and excluded attributes. Class variable could be ignored. Mutual information maximisation is enough in this case, because dependency now has the meaning of replaceability. We are searching for the excluded attributes that carry almost the same information about class variable as the attributes already placed in the network. Before presenting the algorithm, we introduce the notations used in the following:

$\boldsymbol{X}$	set of attributes not *yet* placed in the net	$\boldsymbol{Z}$	set of attributes in the net
X	one attribute from $\boldsymbol{X}$	Z_i	an element from $\boldsymbol{Z}$
Y	the class variable	$I(X,Y\|\boldsymbol{Z})$	conditional mutual information of
$I(X,Y)$	mutual information of X and Y		X and Y given $\boldsymbol{Z}$

Our algorithm introduces a threshold parameter – denoted with α_1 – which is the minimum "information" required when putting a new attribute in the network during the Naive Bayesian structure learning. This parameter controls the number of direct connections between class variable and attributes. The algorithm is presented in algorithm. 1.

The threshold parameter α_1 assures the selection of relevant attributes respect to the class variable, controlling the number of direct causal relations of class variable and attributes. The result is a tree-like Bayesian network as in Figure 2, where the root of the tree is the class variable, and the other nodes are attribute variables. The orientation of edges is from parent to the child, this way minimising the modification of network parameters during a learning step.

The algorithm above is deterministic in sense that it generates the same network for the same data all the time. We introduce importance sampling [14] in order to avoid the determinism of the algorithm in case of selection from equal information quantities. The distribution used for sampling is based on mutual information. It has the maximum where the mutual information is maximal.

We used two functions during the tests. The first function – denoted f_1 – is the conversion of the

Algorithm 1 Tree-like Bayesian network structure learning.

place the class variable Y in the network
$\mathbf{Z} = \emptyset$
{Naive Bayesian structure learning}
while $\mathrm{I}(X, Y|\mathbf{Z}) \geq \alpha_1$ **do**
  $\hat{X} = \underset{\mathbf{X}}{\operatorname{argmax}}\, \mathrm{I}(X, Y|\mathbf{Z})$
  place $\hat{X}$ in the network
  $\mathbf{X} = \mathbf{X} - \{\hat{X}\}$
  $\mathbf{Z} = \mathbf{Z} \cup \{\hat{X}\}$
end while
{Inner structure learning}
while $\mathbf{X} \neq \emptyset$ **do**
  $\left[\hat{X}, \hat{Z}\right] = \underset{X_i, Z_j}{\operatorname{argmax}}\, \mathrm{I}(X_i, Z_j)$
  place edge between $\hat{X}$ and $\hat{Z}$
  $\mathbf{X} = \mathbf{X} - \{\hat{X}\}$
  $\mathbf{Z} = \mathbf{Z} \cup \{\hat{X}\}$
end while

Figure 1: Tree-like Bayesian network structure learning.

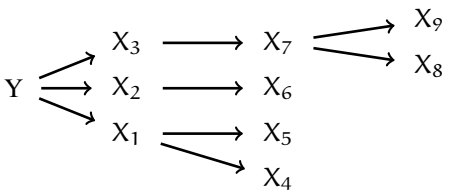


Figure 2: Structure of a BN

mutual information to a distribution function:

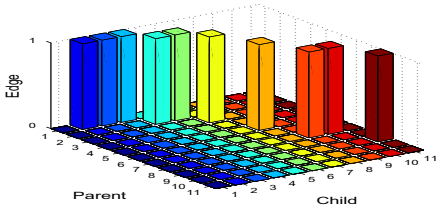
$$\mathrm{f}_1(X) = \frac{I(X,Y)}{\sum_{X' \in \boldsymbol{X}} I(X',Y)} \tag{2}$$

Figure 3(b). illustrates what edges are inferred when using importance sampling with function f_1 from artificial data. The generator network for the data is presented on Figure 3(a). On the horizontal plane is the adjacency matrix of the graph and the non-zero columns represent the edges. Figure 3(a). points the edges of the generator network, and Figure 3(b). shows the frequency of learned edges during 300 tests.

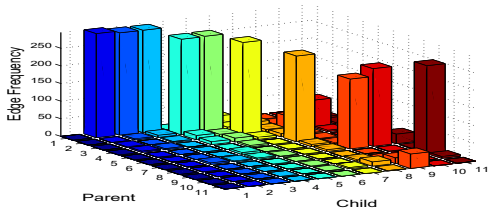
The second function – denoted f_2 – uses the exponentiation of the mutual information. It has a β parameter which can be understood as a temperature parameter and it controls the constructed distribution function. The higher this parameter is the higher is the probability of selecting the maximum mutual information. On lower values of β the probabilities of selecting an attribute becomes closer to the uniform distribution.

$$f_2(X) = \frac{\exp(\beta \cdot \mathrm{f}_1(X))}{\sum_{X' \in \boldsymbol{X}} \exp(\beta \cdot \mathrm{f}_1(X))} \tag{3}$$

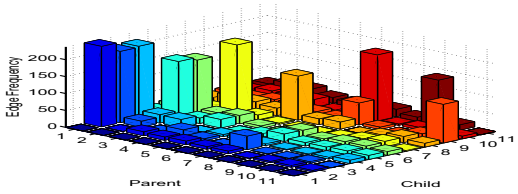
Figure 3. shows the histogram of learned edges using the presented approaches and also the generator network topology of data on Figure 3(a). In each graph on the horizontal plane is the adjacency matrix of the network topology, and the vertical columns represent the histogram edges. We consider the first



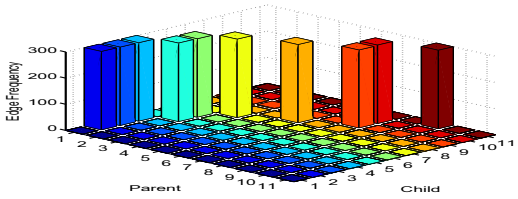
(a) Generator network topology



(b) Learned topologies with function f_1



(c) Learned topologies with function f_2, $\beta = 5$



(d) Learned topologies with function f_2, $\beta = 20$

Figure 3: Generator network and histogram of BN edges

attribute from the network as the class variable, so it is the root of the constructed tree. This means, there's no arc from any attribute to the class variable, hence it's column is 0 at each point. Figure 3(a). represents the generator network topology for the data, hence each edge appears once. The other graphs – on figures 3(b), 3(c), and 3(d) – represent the frequency of edges in the inferred network topologies.

Although there is a randomness introduced with importance sampling in our first approach, the generated structure is relatively stable through the iterations when learning with function f_1. The direct causal relations between the class variable and attributes are almost the same during the 300 tests simulations, differences can be observed only in the causal relations between the attributes, more precisely the differences are on the third level of the tree. There in approximately 100 cases – out of 300 – a single edge is placed differently compared to the generator network.

A bit more unstable structure (Figure 3(c)) can be observed when learning with function f_2 with parameter $\beta = 5$. This is due to the fact, that the separation between lower and higher values of mutual information is more sensitive for lower values of β. Figure 3(d). represents the learned structures for $\beta = 20$. One can see that in this case the structure is fully stable, which assures the former statement.

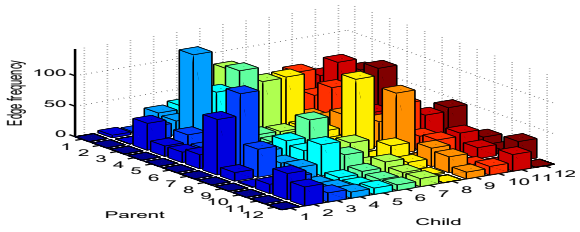
In next sections we will analyse the convergence of the learning process and the usage of the constructed network.

4 Results

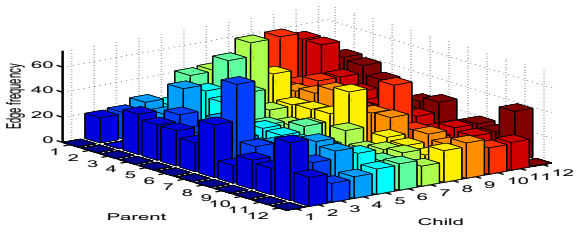
In this section we will present the learned topologies in case of real data. To fully settle the bypass problem, we have to perform the binarisation of data. This is due to the very low number of data samples in bypass database considering conditional probability distribution function estimation.

We made 300 test of the algorithm on the fully specified attributes from the database. Figure 4. presents the results of these testings. One can observe a high order of uncertainty when learning with function f_2 with $\beta = 5$. This is reduced by the increase of β to 20. Function f_1 highlights almost the same dependency relations as function f_2 with $\beta = 20$, but there is an annoying level of uncertainty in these relations.

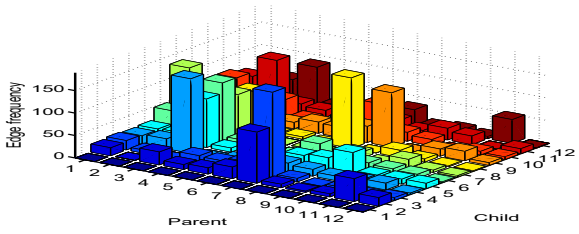
Further analysis of the algorithm meant to check stability of it on real data-set. For this reason Leave-One-Out method on attributes was used: we eliminated one attribute from the data, and tested the algorithm on the remaining ones. The results are showed on Figure 5., where Figure 5(a)., Figure 5(c).



(a) Histogram of edges with function f_1



(b) Histogram of edges with function f_2, $\beta = 5$



(c) Histogram of edges with function f_2, $\beta = 20$

Figure 4: Learned topologies of fully specified attributes from the database

and Figure 5(e). depicts the histogram of learned edges, and on Figure 5(b)., Figure 5(d). and Figure 5(f). the most frequent learned edges are drawn.

Figure 5(a). and Figure 5(b). depicts the histogram of learned edges respective the most frequent edges learned with function f_2, $\beta = 20$ for all fully specified attributes from the bypass database. One can see, that there are two crucial attributes in the database, namely the thirrd and seventh, which are central players in attribute dependence relations. Next sub-figures – Figure 5(c). and Figure 5(d). - elimination of 3rd. attribute, respective Figure 5(e). and Figure 5(e). - elimination of 3rd. attribute – depicts the learned BNs when eliminating these attributes.

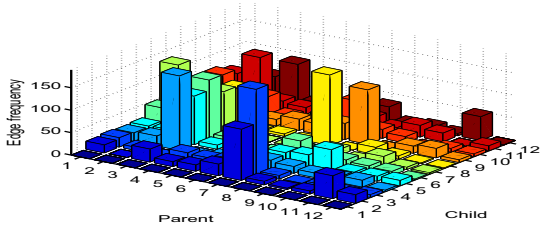
One can see that most of the dependence relations are stable, although there is a reorganisation between dependencies, when eliminating a crucial attribute. The most observable instability in dependence relations is that the edge $3 \rightarrow 12$ becomes an edge $2 \rightarrow 12$ when eliminating either the third attribute or the seventh attribute. But when analysing the dependency of $2 \rightarrow 12$, respective $3 \rightarrow 12$, one can see that their is not so much significant difference between the frequency of the two edges when learning on all attributes.

5 Conclusions

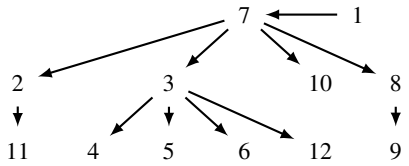
In this paper we presented a tree-like Bayesian network classifier algorithm developed for medical decision making problems and a stochastic algorithm to find the most appropriate structure of the network. We tried two functions for eliminating determinism from the algorithm, with the two functions f_1 and f_2, defined with eq. 2, respective eq. 3. Learned topologies with the presented algorithm and functions were presented both for artificial and real data. In this section we will present results considering the inference, and compare them with logistic regression and SVM.

Table 1. shows the results of efficiency of the presented algorithms compared to logistic regression. Comparing the first and second approach we described above the results are surprising. Although the high order of uncertainty in some cases, the results of efficiency are similar for all cases.

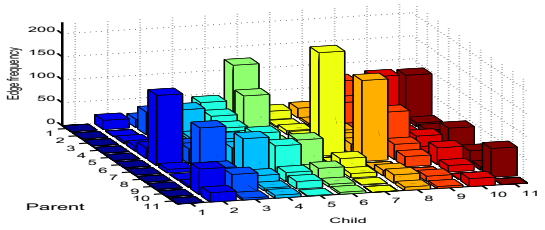
One can observe that the tree-like Bayesian networks constructed with the presented algorithm perform better than logistic regression, but Support Vector Machines with linear kernel obtain higher prediction accuracy than BN-s. It has to be mentioned, that although better accuracy on SVM, they don't allow interpretation of predictions, while BNs do, and interpretation in case of tree-like Bayesian networks,



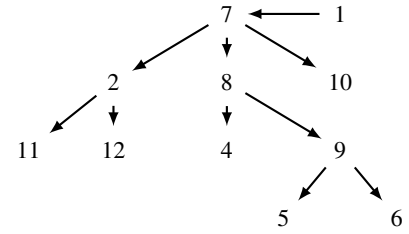
(a) Histogram of edges - all attributes



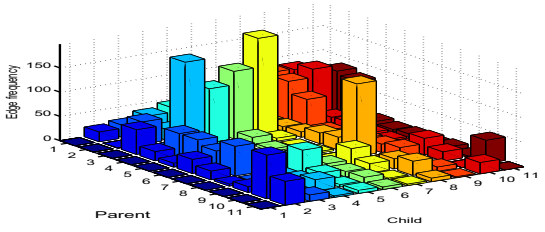
(b) Most frequent edges - all attributes



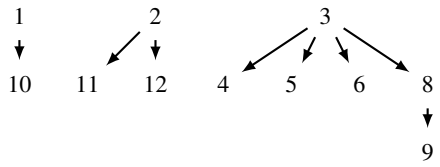
(c) Histogram of edges - LOO 3. attribute



(d) Most frequent edges - LOO 3. attribute



(e) Histogram of edges - LOO 7. attribute



(f) Most frequent edges - LOO 7. attribute

Figure 5: Leave-One-Out results on attributes

Method	**Accuracy**
Bayesian network - f_1	74.69%
Bayesian network - f_2, $\beta = 1$	74.25%
Bayesian network - f_2, $\beta = 5$	74.64%
Bayesian network - f_2, $\beta = 20$	75.71%
Logistic regression	63.50%
SVM with linear kernel	89.84%

Table 1: Efficiency of presented algorithms

constructed as above, can be done efficiently.

As for a summaryy of results it has to be mentioned that the learned structure by the algorithm is generally stable; the interpretation of the results is possible and partial observability is not a problem in case of prediction and interpretation.

Acknowledgements

We acknowledge the program committee of the International Conference on Computers, Communication and Control 2008 for the recommendation and also thank for the problem description and the medical database to Béla Vizvári from Department of Operations Research, Eötvös Loránd University, Budapest. This work was partially supported by the Romanian Ministry of Education and Research through grant 11-039/2007.

Bibliography

[1] Zs. Csizmadia, P.L.Hammer, B. Vizvári. Generation of artificial attributes for data analysis. Rutcor Research Report RRR 42-2004, Rutgers Center for Operations Research, Rutgers University, 2004.

[2] Zs. Csizmadia, B. Vizvári. Methods for the analysis of large real-valued medical databases by logical analysis of data. Rutcor Research Report RRR 42-2004, Rutgers Center for Operations Research, Rutgers University, 2004.

[3] Judea Pearl. *Causality: Modeling, Reasoning, and Inference*. Cambridge University Press, Cambridge, 2000.

[4] Beáta Reiz, Lehel Csató. Tree-like bayesian network classifiers for surgery survival chance prediction. In *Proceedings of International Conference on Computers, Communications and Control*, Vol. III, pp. 470-474, 2008.

[5] Kevin P. Murphy. Learning bayes net structure from sparse data sets. Technical report, Comp. Sci. Div., UC Berkeley, 2001.

[6] Jie Cheng, David A. Bell, and Weiru Liu. An algorithm for bayesian belief network construction from data, 1997.

[7] Jie Cheng, David A. Bell, and Weiru Liu. Learning belief networks from data: An information theory based approach. In *CIKM*, pages 325–331, 1997.

[8] Mieczyslaw A. Klopotek. Mining bayesian network structure for large sets of variables. In *ISMIS*, pages 114–122, 2002.

[9] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.

[10] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[11] David Heckerman and Christopher Meek. Models and selection criteria for regression and classification. Technical Report MSR-TR-97-08, Microsoft Research, 1997.

[12] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, November 2004.

[13] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *Information Theory, IEEE Transactions on*, 14(3):462–467, 1968.

[14] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

Lehel Csató obtained his BSc and MSc degrees at the Babeş–Bolyai University, Cluj–Napoca, and his PhD degree from the Neural Computing Research Group at the University of Aston, the United Kingdom. He was interested in the applications of machine learning techniques, specifically to apply non-parametric methods in Bayesian inference. His thesis investigated methods to approximate solutions of Bayesian regression using stochastic Gaussian processes, centring on sparse solutions that approximate the Gaussian processes. He is teaching at the Babes-Bolyai University and he is interested in applications of Bayesian techniques in modern data processing and probabilistic methods in robotics. He is heading the "Data Mining Research Group" at the same university. Web–page: `http://www.cs.ubbcluj.ro/~csatol`

Beáta Reiz obtained her BSc and MSc degrees at the Babeş–Bolyai University, Cluj–Napoca. Currently she is a PhD student in Hungary, University of Szeged and she is working in the Bioinformatics group of the Biological Research Center, from Szeged under the supervision of Sándor Pongor and János Csirik.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 73-81

Modeling of Errors Realized by a Human Learner in Virtual Environment for Training

Thanh-Hai Trinh, Cédric Buche, Ronan Querrec, Jacques Tisseau

Université Européenne de Bretagne, Ecole Nationale d'Ingénieurs de Brest, Laboratoire Informatique des Systèmes Complexes, Centre Européen de Réalite Virtuelle
Technopôle Brest-Iroise, 29238 Brest Cedex 3, France
E-mail: {trinh,buche,querrec,tisseau}@enib.fr

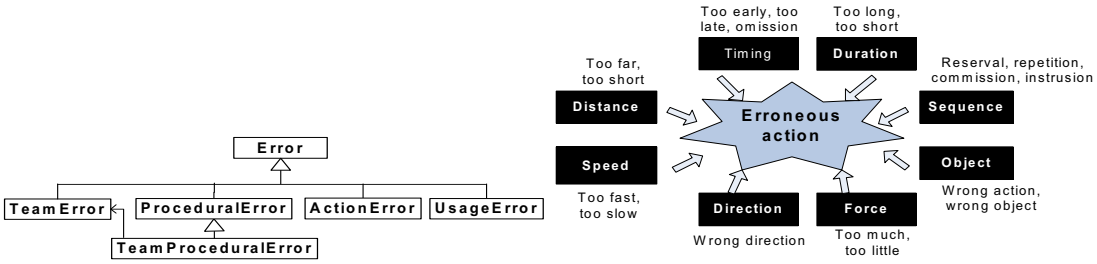
Abstract: This study focuses on the notion of erroneous actions realized by human learners in Virtual Environments for Training. Our principal objective is to develop an Intelligent Tutoring System (ITS) suggesting pedagogical assistances to the human teacher. For that, the ITS must obviously detect and classify erroneous actions produced by learners during the realization of procedural and collaborative work. Further, in order to better support human teacher and facilitate his comprehension, it is necessary to show the teacher why learner made an error. Addressing this issue, we firstly modeling the Cognitive Reliability and Error Analysis Method (CREAM). Then, we integrate the retrospective analysis mechanism of CREAM into our existing ITS, thus enable the system to indicate the path of probable cause-effect explaining reasons why errors have occurred.
Keywords: Intelligent tutoring system, Erroneous actions, Retrospective analysis

1 Introduction

In order to simulate procedural and collaborative work, we developed the model MASCARET (Multi-Agent System for Collaborative Adaptive and Realistic Environment for Training) where human learners and agents collaborate to realize a task [1]. Learners are gathered in team consisting of several predefined roles, every role contains a number of actions to be realized by learners with specific resources. During realization of the tasks, it is essential to take into account that human learners may make erroneous actions in comparing to their predefined correct procedure.

In [2], we have proposed a model of Intelligent Tutoring System (ITS) whose principal objective is to suggest pedagogical assistances to the teacher adapted to the simulation context and to the learner's behaviours (including erroneous actions). However, this works exclusively concerns errors detection and tagging. Once erroneous actions are detected in our existing ITS, it were be classified in different types (cf. Figure 1(a)) whose explications are based on a knowledge base on classical errors. In order to better support the teacher and facilitate his comprehension, it lacks a model that could explain reasons why the learner made an error.

Our approach bases on the Cognitive Reliability and Error Analysis Method (CREAM) in Human



(a) Errors types in ITS [2] (b) Dimensions of error modes [3]

Figure 1: Errors types and errors's phenotypes

Reliability Analysis field [3]. This approach proposed a classification scheme which makes a distinction between observations of errors (*phenotypes*, cf. Figure 1(b)) and its causes (*genotypes*) classified in

three categories: M(an), T(echnology) and O(rganization). For example, since the learner made a mistake about the order of actions, the erroneous action observed is in phenotype *Sequence* and that can be futher explained by some genotypes such as *Inattention* (Man related genotype), *Communication failure* (Technology related genotype), etc. The causal links between phenotype-genotype are represented using a number of consequent-antecedent links. Finally, the scheme could be associated with both a method of retrospective analysis (the search for causes) and a performance prediction method. However, in our goal of erroneous actions detection and then searching for the causes, we interested in human learner's performance analyses, in other words, in retrospective analyses.

Implementation of CREAM was object in the work of El-Kechaï [4][5] which firstly proposed a task model named METISSE in order to recognize learner's plans in Virtual Environments for Training (VET), then this model could be used to detect for erroneous actions according to classification of Hollnagel. Nevertheless, implementation of METISSE was not complete, and integration of CREAM into a really ITS was not performed.

In this paper, we will firstly propose an approach to model CREAM (Section 2). Next, in Section 3, we will present the integration of retrospective analysis mechanism of CREAM into our existing ITS as well as our evaluation.

2 Implementation of CREAM

2.1 Classification Scheme Representation

There are several graphic tools that permit to keep track of analyses processes such as CREAM Navigator developed by Serwy and Rantanen [7]. However, this navigator is completely closed in the sense that it does not maintain an explicit representation of possible errors modes and probable causes. For that, [4] proposed using a rules base for represent consequent-antecedent links, hence the search for the causes was executed by backward inferences. Limitation of this method obviously lies on the performance of inference mechanism, other problem maybe occurs in adding, removing another potential errors that will demand a considerable modification on the rules base. For our development, as suggested in [3], we intent to separate the analysis method (cf. Section 2.3 and 2.4) and the representation of errors modes using a group of four data files in format XML detailed below:

Questionnaire.xml : proposing to represent a list of questions from which we could evaluate the Common Performance Conditions (see Section 2.2 in following).

Phenotype.xml : proposing to maintain the phenotypes and its antecedents (cf. Figure 2).

```
<Phenotypes>
   <Phenotype name="Time/During" description="...">
      <GeneralAntecedents>
         <item>Inadequate plan</item>
         <item>Inattention</item>
      </GeneralAntecedents>
      <SpecificAntecedents>
         <item>Earlier omission</item>
      </SpecificAntecedents>
   </Phenotype>
   ...
</Phenotypes>
```

Figure 2: Representation of phenotypes

Genotype.xml : containing all possible causes classified in three groups (M,T,O), each group is then detailed into several categories. The important point is that this data file also represents relations between each consequent and its antecedents (cf. Figure 3).

Repartition.xml : proposing to determine repartition of specific antecedents (cf. Figure 4) in three factors (M,T,O) which serves to initialize the mass of each specific antecedent as a probable cause (cf. Section 2.4).

Finally, in considering that CREAM is naturally a flexible method and adaptable to different analysis contexts, this strategy of classification scheme representation permits customize the scheme without any modification on analysis method.

```
<Genotypes>
   <Group name="Man">
      <Category name="planning">
         <GeneralConsequent name="Inadequate plan" description="...">
            <GeneralAntecedents>
               <item>Distraction</item>
               <item>Excessive demand</item>
            </GeneralAntecedents>
            <SpecificAntecedents>
               <item>Error in goal</item>
               <item>Inadequate training</item>
            </SpecificAntecedents>
         </GeneralConsequent>
         ...
      </Category>
      ...
   </Group>
   ...
</Genotypes>
```

Figure 3: Representation of genotypes

```
<Repartition>
   <item name="Earlier omission" group="Man" description="..."/>
   <item name="Message misunderstood" group=Technology description="..."/>
   ...
</Repartition>
```

Figure 4: Repartition of specific antecedents in three factors (M,T,O)

2.2 Define the Common Performance Conditions (CPC's)

In CREAM, Hollnagel highlighted that the context strongly influence human actions. It is therefore essential to take into account the description of virtual environment in which the human learner is immersed. The objective is to determine how each factor (M,T,O) influences the training context. Here, we are inspired from the proposition presented in [5] using a predefined questionnaire which will be answered by the teacher before training session (cf. Figure 5). Next, each factor will be assigned one

```
<Questionnaire>
<Question name="The visual quality of the interface is it bad?" group="Technology" answer="Yes"/>
<Question name="Does the learner have concentration trouble?" group="Man" answer="No"/>
...
</Questionnaire>
```

Figure 5: Define the CPC's by questionnaire [5]

coefficient calculated using formula below:

$$Coefficient_{group\,i} = \frac{Number\,of\,Yes\,answers\,associated\,to\,group\,i}{Total\,number\,of\,Yes\,answers} \quad (1)$$

where *group i* is respectively in (Man, Technology, Organization). These values permit define the most probable factor leading to erroneous actions.

2.3 Modelling of Consequent-Antecedent Relations

One advantage of CREAM lies on its recursive analysis approach, rather than strictly sequential in compare with other traditional analysis methods. So that, it also conducts to a non-hierarchical data structure to connect the direct as well as indirect links: *(i)* between a phenotype and its antecedent; and *(ii)* between a consequent and its antecedents. Figure 6 shows our model to represent the connection between consequent-antecedent.

Here, we are going to construct a causal graph where we use the term *node* to point to either a consequent or an antecedent. Each node is described by its *name*; the group of errors modes that it is associated and its *category* in group; the *description* in text helps better explain the error's semantics in particular context. The boolean attribute *terminal* permit to identify if that is a terminal-cause or not. The most important is that, each node contains two lists: one includes its antecedents, other points to its consequents, in others words, they represent edges in/out one node in the causal graph. At last, each node must also

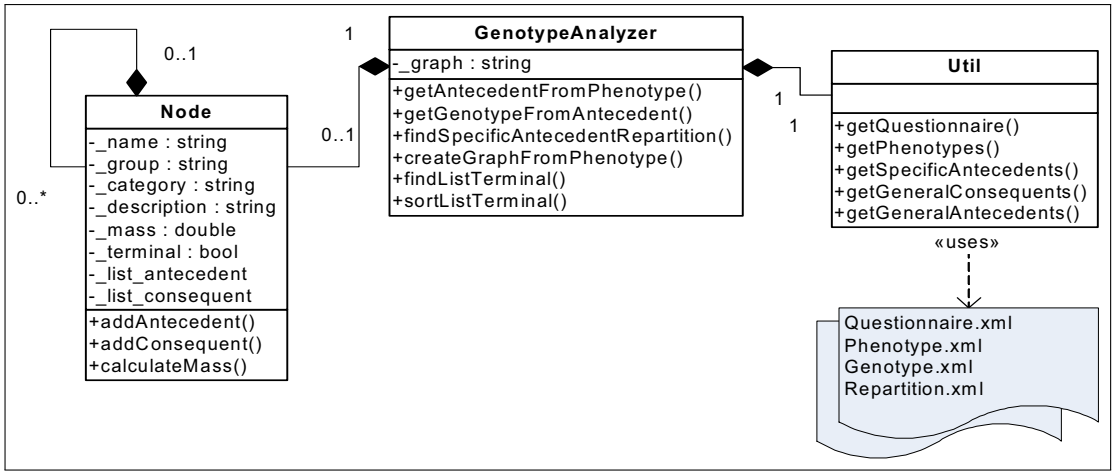


Figure 6: UML diagram for modeling consequent-antecedent links

include a value of *mass* which represent the certitude of choosing this node as a probable cause. The two methods *addAntecedent()* and *addConsequent()* serve for maintaining the two lists of antecedents and consequents of one node. Note that once a node calls the method *addAntecedent()* serving for adding a "parent" node like one of its antecedents, this node will also add itself to the consequents list of the "parent" node (using the method *addConsequent()* of the parent node), the value of the attribute *terminal* then will be set to false.

2.4 Search for the Causes

The retrospective analysis is executed by a *GenotypeAnalyzer* containing *graph* attribute which is initialized by pointing to the phenotype input (root node), then the analyzer calls accurate methods to find the root causes (the nodes with the attribute terminal having value false). This mechanism is presented below (cf. Algorithm 1).

Algorithm 1 Retrospective analysis

Require: Phenotype of erroneous action
1: **Initialization**: Construct the "root" node pointing to phenotype input
2: {**Step 1**: Finding antecedents of phenotype input}
3: Read from file *Phenotype.xml*, find all general antecedents of phenotype input
4: **for** each antecedent **do**
5: Add it into antecedents list of "root" node
6: **end for**
7: {**Step 2**: Construction the causal graph}
8: **for** each unvisited node in the graph **do**
9: Find its antecedents from file *Genotype.xml*
10: Add them to antecedents list
11: **end for**
12: Return **Step 2**. This recursive search terminates when the node selected is a specific antecedent node or a general consequent node without antecedents.

With this algorithm, we finally attain a causal network where each node is associated with its antecedents and consequents. The "leaves" are terminal nodes (or "root" causes) whose antecedents list is empty. In order to calculate the certitude of choosing each node as a probable cause, we inherit the proposition presented in [5] using Dempster-Shafer's evidence theory:

$$mass(a) = coefficient\,(g\,(a)) * \sum_{\forall c \in Cons(a)} \left(\frac{mass(c)}{\sum_{\forall i \in \{M,T,O\}} (coefficient(i) * n_{ic})} \right) \tag{2}$$

where:

- $mass(a)$: mass of antecedent a
- $g(a)$: group of a
- $Cons(a)$: consequents list of a
- $coefficient(i)$: coefficient of group i calculated in Formula 1
- n_{ic} : number of antecedents of c classified in group i

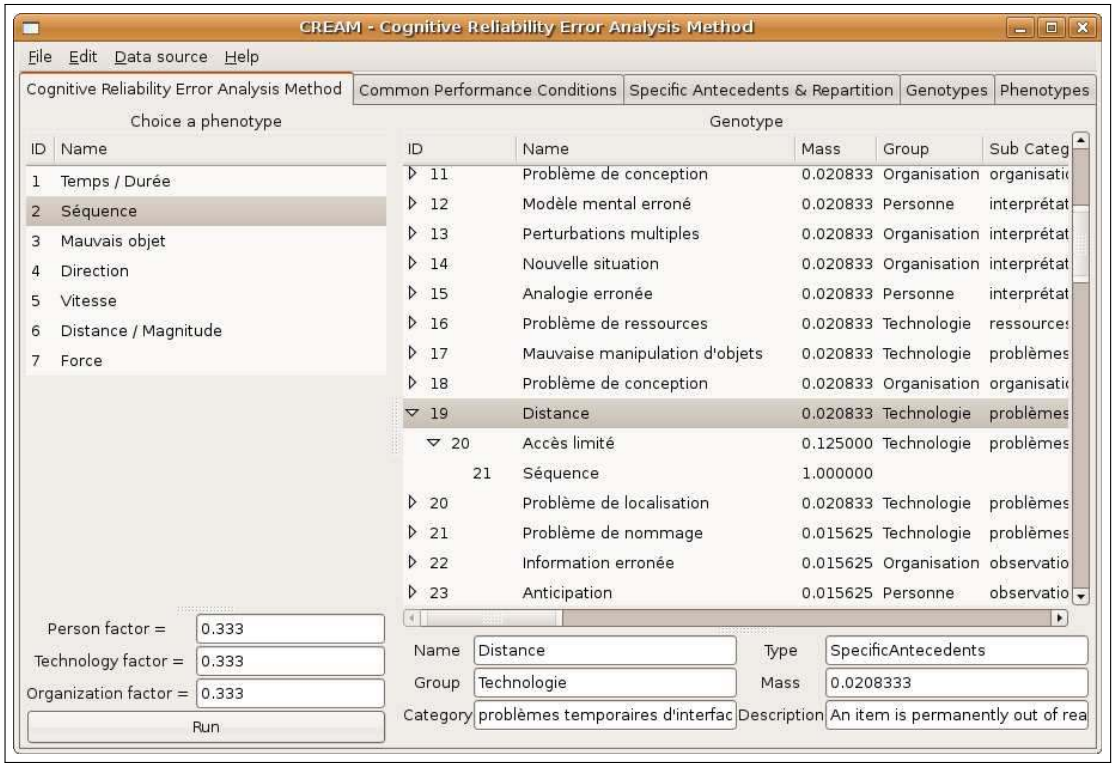


Figure 7: CREAM Explorer

Finally, the Figure 7 illustrates our tool - CREAM Explorer which was developped in this phase permitting to maintain the errors scheme, answer the questionnaire for define the CPC's and execute the retrospective analysis.

3 Integration of Retrospective Analysis into our existing ITS

3.1 Learner's Plans Recognition

In order to detect the erroneous actions realized by a human learner, it is indispensable to know:

(i) the learner's activities in the past;

(ii) his current action (in the meaning that the action has just been done);

(iii) the actions that the human learner intents to do in according to a predefined correct procedure.

Our existing ITS as proposed in [2] bases on the model MASCARET [1] where we used an multi-agent system to simulate collaboration between human learners and agents during their realization of tasks. Learners are gathered in team consisting of several predefined roles, every role contains a number of tasks associated eventually with accurate resources, every leaner also owns an epistemic memory containing all actions realized in the past, etc. Finally, we could retrieve from MASCARET following informations relating to learner's plan in VET:

- *action(s) before*: learner's action(s) in the past (note that, in MASCARET, every action is eventually associated with its accurate resource(s))
- *current action*: action has just been done by learner
- *action(s) correct (according to role)*: action(s) must be done by learner in his role(s)
- *action(s) correct (according to plan)*: action(s) may be done by learners in the context. Here, it is essential to make distinction betweens *action(s) correct according to role* and *action(s) correct according to plan*. In the first case, because the learner could play several roles, it represents all correct actions that the system expects from the learners. The second one concerns the cases where there are more than one learner in VET to realize together a mission. Therefore, in this case, it is possible that a leaner performs a correct action according to the plan but it is not correct in compare to his role.
- *next correct action(s) in the role*: next action(s) must be done by learner in his role(s)
- *full correct plan*: description of all accurate actions (associated with resources) in predetermined procedure that the learner must respect.

In next section, we present our mechanism for mapping erroneous actions detected by our existing ITS with Hollnagel's classification scheme of errors modes.

3.2 Classification of Erroneous Actions according to the Scheme of CREAM

Erroneous Actions in Phenotype "Sequence"
According to Hollnagel, performing an action at the wrong place in a sequence or procedure is a common erroneous action, and it is more realistic in our context of simulation of procedural and collaborative work. The *"Sequence"* problem consists of several specific effects: *Omission* (an action was not carried out); *Jump forward/ Jump backwards* (actions in a sequence were skipped/carried out again); *Repetition* (the previous action is repeated); *Reversal* (the order of two neighbouring action is reversed); *Wrong action* (an extraneous or irrelevant action is carried out). We present in following our mechanism to detect erroneous actions in phenotype *"Sequence"*:

Algorithm 2 Detection of erroneous actions in phenotype *Sequence*

```
1:  if current action exists in actions correct according to role then
2:      this is a correct action (phenotype Sequence does not occur)
3:  else
4:      if current action does not exist in actions correct according to plan then
5:          specific effect = "Wrong action"
6:      else
7:          if current action exist in last action before then
8:              specific effect = "Repetition"
9:          end if
10:         Compare the relative order of current action to the order of next correct action(s) in the role
            using the full correct plan
11:         if id current action < id correct action in role then
12:             specific effect = "Jump backwards and/or Omission"
13:         else
14:             specific effect = "Jump forward and/or Omission"
15:         end if
16:         if id current action = id correct action in role + 1 then
17:             specific effect = "Reversal"
18:         end if
19:     end if
20: end if
```

Erroneous Actions in Phenotype "Wrong object"
In [3], the author clarified that *"action at wrong object"* is one of the more frequent error modes, such as pressing the wrong button, looking at the wrong indicator, etc. In our context, during realisation of

collaborative work, it is possible that learner performs a correct action but on a wrong object. Therefore, the detection of erroneous actions in phenotype *"Wrong object"* must be implemented independently with the detection of phenotype *"Sequence"*. This phenotype is detailed into following specific effects: *Neighbour/Similar object* (an object that is proximity/similar to the object that should have been used); *Unrelated object* (an object that was used by mistake).

In order to detect erroneous actions in phenotype *"Wrong object"*, we use the same principle presented in the case of phenotype *"Sequence"* by using following informations retrieved from model MASCARET:

- *current resource*: resource associated with *current action*
- *resource(s) correct (according to role)*: resource(s) must be used by learner in his role(s)
- *resource(s) correct (according to plan)*: list of resource(s) associated with all action(s) in *action(s) correct according to plan*.

Our algorithm is detailed in following:

Algorithm 3 Detection of erroneous actions in phenotype *Wrong object*

```
1: if current resource exists in resource(s) correct according to role then
2:     this is a correct resource (phenotype Wrong object does not occur)
3: else
4:     if current resource does not exist in resources correct according to plan then
5:         specific effect = "Unrelated object"
6:     else
7:         specific effect = "Neighbour and/or Similar object"
8:     end if
9: end if
```

Erroneous Actions in Phenotype "Time/During"

The phenotype *Time/During* is divided in several specific effects: *Too early/ Too late* (an action started too early/too late); *Omission* (an action that was not done at all); *Too long/Too short* (an action that continued/was stopped beyond the point when it should have been). Hollnagel noted that the error modes of timing and duration refer to a single action, rather than to the temporal relation between two or more actions. In our context, the realization of tasks in model MASCARET is sequential, therefore, an action is considered to be too early when it was realized before several actions in plan; also, action(s) are considered to be omitted when they were not carried out.

Finally, in order to detect erroneous actions in phenotype *Time/Durring*, we propose that:

- action having specific effect *Jump forward* also has specific effect *Too early*
- action described by specific effect *Omission* (in error mode *Sequence*) will be considered as an action having specific effect *Omission* (in error mode *Time/During*)

3.3 Experiment & Results

In order to evaluate our integration of retrospective analysis into ITS, we take place in GASPAR application [6] whose objective aims at simulate aviation activities by virtual reality. The learners are immersed in virtual environment simulating the aircraft carrier in order to realize together the tasks. During the realization of these collaborative works, our ITS follows the learners and then apply the algorithms depicted above for detecting learner's erroneous actions. Next, for interpreting the causes of errors, we use the classification scheme of error modes proposed in [5] which were particularly adapted to VET. Table 1 and Table 2 respectively illustrate results of retrospective analysis for the phenotype Sequence and Wrong object.

We change coefficients of three factors (M,T,O) for evaluating how CPC's influence the analysis result. For each phase in analysis process, we select and display the most probable cause by ordering mass values.

Coefficient (M,T,O)	Causal links
(0.333 - 0.333 - 0.333)	1, Design failure (0.125) → Inadequate scenario (0.125) → Sequence
	2, Adverse ambient condition (0.125) → Inattention (0.125) → Sequence
	3, Long time since learning (0.042) → Memory failure (0.125) → Sequence
(1 - 0 - 0)	1, Other priority (0.2) → Memory failure (0.2) → Sequence
	2, Error in mental model (0.067) → Faulty diagnosis (0.2) →Sequence
	3, Erroneous analogy (0.067) → Faulty diagnosis (0.2) → Sequence
(0 - 1 - 0)	1, Equipment failure (0.1) → Access problems (0.5) → Sequence
	2, Distance (0.1) → Access problems (0.5) → Sequence
	3, Localisation problem (0.1) → Access problems (0.5) → Sequence
(0 - 0 - 1)	1, Noise (1) → Communication failure (1) → Sequence

Table 1: Causal links of phenotype *Sequence*

Coefficient (M,T,O)	Causal links
(0.333 - 0.333 - 0.333)	1, Access problems (0.125) → Wrong object
	2, Design failure (0.125) → Inadequate scenario (0.125) → Wrong object
	3, Adverse ambient condition (0.042) → Inattention (0.125) → Wrong object
(1 - 0 - 0)	1, Fatigue (0.1) → Performance variability (0.2) → Wrong object
	2, Virtual reality sickness (0.1) → Performance variability (0.2) → Wrong object
	3, Anticipation (0.05) → Wrong identification (0.2) → Wrong object
(0 - 1 - 0)	1, Access problems (0.5) → Wrong object
(0 - 0 - 1)	1, Noise (1) → Communication failure (1) → Wrong object

Table 2: Causal links of phenotype *Wrong object*

4 Conclusion & Future Work

In this paper, we proposed an approach to modelling the Cognitive Reliability and Error Analysis Method (CREAM). We separated the representation of classification scheme of erroneous actions and the analysis method; therefore, our description of errors modes is adaptable to different training context without any modification on analysis method. We started by defining the Common Performance Conditions, then the direct and indirect relations between consequent-antecedent are modelled using a non-hierarchical data structure. Finally, the most probable cause-effect links could be found using Dempster-Shafer's theory presented in [5].

In order to integrate the retrospective analysis described above into our existing ITS, we based on the model MASCARET to retrieve information concerning learner's plans and then detect erroneous actions. Finally, we presented our proposition to mapping erroneous actions with Hollnage's classification. The experimental results in GASPAR project are also presented. So that, in addition to the detection and tagging of erroneous actions, the ITS could furthermore indicate the path of probable cause-effect

explaining reasons that the errors occur.

In the future work, we will concentrate our attention on evaluation of MASCARET so that this model could permit to describe more complex tasks in taking into account other factors such as force, distance, speed, direction, etc. Hence, other different types of errors modes could be detected and then explained using the retrospective analysis.

Acknowledgement

This article is an extended version of our paper [8] published in Proceedings of the 3^{rd} International Conference on Virtual Learning (ICVL'08). The authors would like to thank the Scientific Committee of ICVL'08 (Chaired by Dr. Grigore Albeanu) that recommended the publishing of our extended work in IJCCC.

Bibliography

[1] R. Querrec, C. Buche , E. Maffre , and P. Chevaillier, Multiagents systems for virtual environment for training: application to fire-fighting, *International Journal of Computers and Applications (IJCA)*, pp. 25-34, June 2004.

[2] C. Buche and R. Querrec, Intelligent tutoring system for MASCARET, *Simon Richir and Bernard Taravel, editors, 7th Virtual Reality International Conference (VRIC'05)*, pp. 105-108, April 2005, Laval, France.

[3] E. Hollnagel, *Cognitive Reliability and Error Analysis Method*, Oxford: Elsevier Science Ltd, 1998.

[4] N. El-Kechaï and C. Després, A Plan Recognition Process, Based on a Task Model, for Detecting Learner's Erroneous Actions, *Intelligent Tutoring Systems ITS 2006*, pp. 329-338, June 2006, Jhongli, TaĂŻwan.

[5] N. El-Kechaï and C. Després, Proposing the underlying causes that lead to the trainee's erroneous actions to the trainer, *EC-TEL : European Conference on Technology Enhanced Learning*, pp. 41-55, September 2007, Crète, Grèce.

[6] N. Marion, C. Septseault, A. Boudinot and R. Querrec, GASPAR : Aviation management on an aircraft carrier using virtual reality, *Cyberworlds*, 2007.

[7] R.D. Serwy and E.M. Rantanen, CREAM Navigator, *http://www.ews.uiuc.edu/ serwy/cream/v0.6beta/*, version 0.6, September, 2007.

[8] T.H. Trinh, C. Buche and J. Tisseau, Modeling of errors realized by a human learner in virtual environment for training, 3^{rd} *International Conference on Virtual Learning ICVL 2008*, pp. 71-80, Octobre 31, Constanta, Romania.

Thanh-Hai Trinh, received his MSc in Artificial Intelligent & Multimedia at the Francophone Institute for Computer Science. He is currently a PhD student in CERV (Virtual Reality European Centre held at Brest, France). His current research interests are in the applications of multi-agent systems and artificial intelligent in virtual environements for training.
Cédric Buche is a Professor Assistant in Computer Science and works at the CERV. He works on the use of the behavior modeling agent applied to virtual environment for human learning. He is the leader of the ITS project in MASCARET.
Ronan Querrec is Professor Assistant in Computer Science and works at the CERV. His reasearch work is about virtual environment for training. In this theme, he works on the MASCARET project, a virtual environment meta-model.
Jacques Tisseau is Professor in Computer Science at the Engineer School of Brest (ENIB) where he leads the Computer Science for Complex Systems Laboratory (LISyC). His research focus on autonomous virtual entities, interaction with these entities and epistemology of virtual reality.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 82-91

Development Journey of QADPZ - A Desktop Grid Computing Platform

Monica Vlădoiu, Zoran Constantinescu

Monica Vlădoiu
Petroleum-Gas University of Ploieşti, Departament of Informatics
Bd. Bucureşti, Nr. 39, Ploieşti, Romania
E-mail: mmvladoiu@acm.org

Zoran Constantinescu
Zealsoft Ltd.
Str. Tg. Neamţ, Nr. 60, Bucureşti, Romania
E-mail: zoran@unde.ro

Abstract: In this paper we present QADPZ, an open source system for desktop grid computing, which enables users of a local network or Internet to share resources. QADPZ allows a centralized management and use of the computational resources of idle computers from a network of desktop computers. QADPZ users can submit compute-intensive applications to the system, which are then automatically scheduled for execution. The scheduling is performed according to the hardware and software requirements of the application. Users can later monitor and control the execution of the applications. Each application consists of one or more tasks. Applications can be independent, when the composing tasks do not require any interaction, or parallel, when the tasks communicate with each other during the computation. The paper describes both QADPZ functionality and the process of design and implementation, with focus on requirements, architecture, user interface and security. Some future work ideas are also presented.
Keywords: desktop grid computing, distributed and parallel computing.

1 Introduction

Grid computing and Peer-to-Peer (P2P) are both concerned with the pooling and coordinated use of resources within distributed communities, and are constructed as overlay structures that operate largely independently of institutional relationships [1]. The Grid is foreseen as a system that coordinates distributed resources using standard, open, general-purpose protocols and interfaces to deliver nontrivial qualities of service [1, 2]. Grid computing systems can be classified into two broad types: heavy-weight, feature-rich systems that provide access to large-scale, intra- and inter-institutional resources, such as clusters or multiprocessors, and Desktop Grids, in which cycles are scavenged from idle desktop computers. P2P networks are typically used for connecting nodes via largely ad-hoc connections. A pure P2P network does not have the notion of clients or servers but only equal peer nodes that simultaneously function as both "clients" and "servers" to the other nodes on the network [3].

This paper deals with QADPZ ['kwod 'pi: 'si:], an open source system for desktop grid computing, which enables users from a local network or Internet to share their resources [4, 5]. QADPZ (Quite Advanced Distributed Parallel Zystem) is a system for heterogeneous desktop grid computing that allows a centralized management and use of the computational resources of idle computers from a network of desktop computers. QADPZ users can submit compute-intensive applications to the system, which are then automatically scheduled for execution. Applications can be independent, when the composing tasks do not require any interaction, or they can be parallel, when the tasks communicate with each other during the computation. Thus, the system provides support for both task- and data-parallelism. Here are some important features of QADPZ [4]:

- native support for multiple operating systems: Linux, Windows, MacOS and Unix;
- support for legacy applications, which for different reasons could not be rewritten;
- object-oriented development framework that supports either low-level programming languages as C and C++, or high-level language applications (such as Lisp, Python, or Java), and that provides for using such applications in a computation;

- master worker-model that is improved with some refined capabilities: pushing of work units, pipelining, sending more work-units at a time, adaptive number of workers, adaptive timeout interval for work units, and the use of multithreading [6];
- a master can act as a client to another master. That makes it possible to create a distributed master, which consists of independent master nodes which communicate with each other, thus creating a virtual master;
- extended C/C++ API, which supports creation of lightweight tasks and parallel computing, using the message passing paradigm (MPI) [7];
- low-level optimizations: on-the-fly compression and encryption for communication. To increase performance, an experimental, adaptive compression algorithm, which can transparently choose from different algorithms, is also provided;
- efficient communication by using two different protocols (UDP and TCP/IP);
- autonomic computing characteristics: self-knowledge, self-configuration, self-optimization and self-healing [8].

2 Justification for a new desktop grid system

The idea of using the idle computational resources from existing desktop computers is not new, though the use of such distributed systems, especially in a research environment, has been limited. This is due to the lack of supporting applications, and challenges regarding security, management, and standardization. The need to develop QADPZ has arisen from the following main reasons:

o many existing systems were highly specialized in a very limited set of computationally challenging problems, and hence did not allow the execution of a general application. For example, SETI@home was programmed to solve one specific task: the analysis of data from telescopes [9, 10]. Similarly, distributed.net aimed to test the vulnerability of some particular encryption schemes [11];

o at the time of the development, the source code was generally not available, hence making difficult the extension or analysis of any new, non-standard application. Commercial systems such as Entropia, Office Grid and United Devices offered numerous features, but they were not free [4, 12]. On the other hand, some open source systems were available, e.g. XtremWeb [13], BOINC [14, 15], Condor [16], but they were limited in functionality;

o very few existing systems allowed specific considerations to be made wrt. challenges of computationally intensive applications, especially those of scientific computing and visualization [4]. Systems like BOINC and Bayanihan [12] allowed only task parallelism, where there was no communication between the running tasks during a computation. Most computationally intensive applications need such communication;

o most of the existing systems usually had a complicated deployment procedure, requiring high-level, privileged access to the desktop computers, which made very hard to use such systems on a larger scale, and also made further maintenance of the computers complicated - e.g. Condor and Globus Toolkit [12, 17, 18];

o many of today's networks are heterogeneous, thus requiring a distributed computing system with support for various architectures and different type of operating systems. The Java language provides the incentives for such requirements, and many Java based systems emerged: JXTA, Bayanihan, XtremWeb, Javelin [12]. There were very few systems supporting different architectures and operating systems in native mode, some of them being Condor and BOINC. There were also systems, which run only on one type operating system, either Windows or Unix, thus limiting their usability in heterogeneous environments - for instance, Entropia [12].

3 QADPZ Requirements

Given the reasons mentioned in the previous section, we have set up a set of requirements that a successful desktop grid computing system should satisfy to support computationally intensive applications. The overall goal of the system was to be friendly, flexible and suitable to a variety of needs. The main prerequisite has therefore been an open architecture that could evolve in pace with the needs and challenges of the real world.

Two sets of requirements for QADPZ have been specified: one for the system as a whole, mostly from a functional point of view, and another for the system interface. Additionally, a set of non-functional requirements that concern the development of the platform itself has been established. System requirements are concerned mainly with sharing and management of both resources and application jobs, in a heterogeneous environment. They also involve performance and usability of the system, as required by our conceptual model (extended master-worker). The system interface covers both user interfaces and programming interfaces [4, 6].

The system requirements are listed further on:

- *resource sharing*: idle computational cycles, storage space, specific data, etc. of the desktop machines which contribute to the system;
- *resource management*: efficient management of the available shared resources, which remain under the control of their owners via use policies and specific mechanisms;
- *job management*: users should be able to submit, monitor and control the execution of computational jobs on the system;
- *heterogeneity*: ability to work on a network of heterogeneous desktop computers, with different architectures (Intel, RISC, etc.) and different operating systems (UNIX, Windows, Mac OS, Linux);
- *simple installation and minimal maintenance;*
- *parallel programming support*: support for different parallel programming paradigms, for example both task- and data-parallelism, by using well known standards;
- *network support*: ability to work both in a LAN environment and in Internet;
- *communications*: the higher level communication protocol should rely on both TCP/IP and UDP/IP, this dual support increasing the efficiency;
- *autonomous features*: support for different autonomicity aspects: self-management, self-optimization, self-healing, self-configuration, and self-knowledge;
- *provide performance measurements*, which could be exploited for better usage of the available resources;
- *on-line/off-line support* for both batch (the user submits jobs which will be executed at a later time) and interactive applications (the user can inspect the partial result and interact with the execution of the application).

The interface requirements can be split up into two parts: first, the *user interfaces* that is the graphical interface, which the human users use to access the system. Using this interface, the users can either monitor or control the behavior of the system. The other interface is the *programming interface* (API), which allows different user applications to interact with the system. The interface requirements are enlisted beneath:

- *personalization*: different levels of access for various users, according to their personal skills and preferences;
- *job management interface*: a simple, platform independent, graphical user interface, to allow submission, monitoring and control of the different computational jobs;
- *resource sharing interface*: a simple, intuitive graphical user interface, which allows the control of shared resources.

The main non-functional requirements concern *object oriented programming*, for its well-known advantages, and *open source development*, which is a natural choice for modern research, as it encourages integration, cooperation and boosting of new ideas [19].

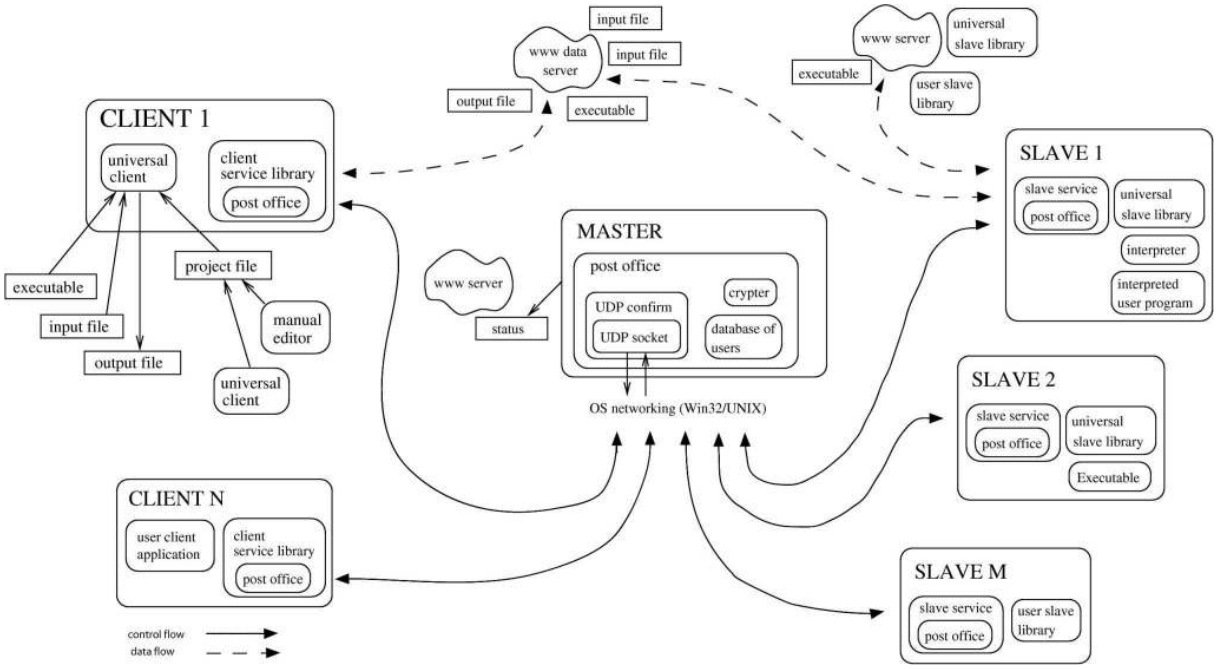


Figure 1: The QADPZ close-up architecture

4 QADPZ Architecture

The QADPZ system has a centralized architecture, based on the client-server model, which is the most common paradigm used in distributed computing. In our case, the server manages the available computational resources of the desktop computers. The client is a process that needs computational services in order to accomplish a certain work. It sends a request to the server, in which it asks for the execution of a concrete task that is covered by the services. Usually, the server carries out the task and sends back the result to the client. In our situation, the server has two parts: a single master, which accepts new requests from the clients, and multiple slaves, which handle those requests. The system consists of three types of entities: master, client, and slave (Figure 1).

The control and data flow in the system are separated. The data files (represented by binary, input, and output files) that are necessary to run the applications are not sent to the master. They are stored on one or more data servers. The smallest independent execution unit of the QADPZ is called a *task*. To facilitate the management, multiple tasks can be grouped into *jobs*. Different types of jobs can be submitted to the system: programs written in scripting languages (e.g. LISP, Java, Python), legacy applications and parallel programs (MPI). A job can consist of independent tasks, which do not require any kind of communication between them. This is usually called *task parallelism*. Jobs can also consist of parallel tasks, where different tasks running on different computers can communicate with each other. Inter-slave communication is accomplished using a MPI subset.

The current implementation of QADPZ considers only one central master node. This can be an inconvenience in certain situations, when computers located in different networks are used together. However, our high-level communication protocol between the entities allows a master to act as a client to another master, thus making possible to create a *virtual master*, consisting of independent master nodes, which communicate with each other.

4.1 Master

The main role of the master is to start and control the tasks, and to keep track of the availability, capabilities and configuration of the slaves. The *master* is responsible for managing the available resources and it has always an up-to-date overview of the system resources. It knows which slaves can accept jobs for execution and how to contact them. It schedules also the computational tasks submitted by any authorized user. Jobs are sent to the appropriate slave based on the hardware and software requirements from the job description. Tasks can be started, stopped, or re-scheduled by the master. Users create tasks that can be submitted to the master by using a *client*, which acts as an interface to the system. To make

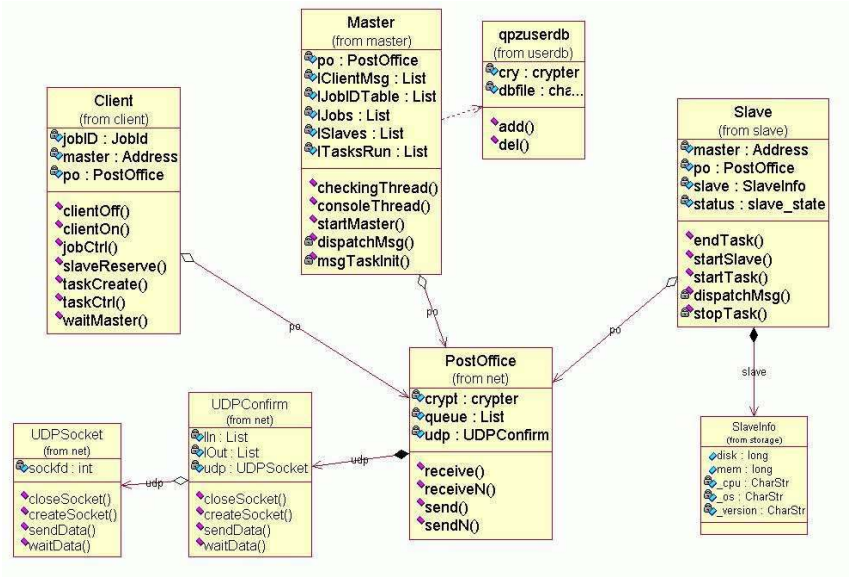


Figure 2: Simplified UML Diagram of QADPZ's architecture

this possible, the master keeps a database of authorized users (Figure 2).

4.2 Slave

Each computer contributing with computing resources to the system is called *a slave* and has two roles: first, it has to report the shared resources to the master. These are mainly computational resources (CPU cycles), but can also be storage space, input or output devices etc. The slave periodically sends to the master information about the system, which describes the hardware architecture of the slave (CPU type, CPU speed, physical memory, etc.), the software environment available on that architecture (operating system, available application or libraries), and the resources available on that slave (Figure 5). Secondly, the slave can accept computational jobs from the master. After accepting a computational request from the master, the slave downloads the corresponding binaries and data files for the task, executes the task, and uploads the result files after finishing. This can be done only when the slave is free, and any interactive, local user is not using the resources. The presence of a user logging into a slave computer is automatically detected and the task is killed or moved to another slave to minimize the disturbance to the regular computer users. The slave decides for itself whether or not to accept a computational job to be run (by setting some configuration parameters). The user can configure different times of day when the slave may accept computational jobs. It can also disable the slave at any time. The slave component runs as a small background process on the user's desktop. It starts automatically when the system starts. The program does not need any special privileges to run, which makes it very easy to install and control by an ordinary user. Below we present a simple example on how to create a computational application to be executed on a slave.

```
// SlaveDumb - simple example of how to create a computational job
#include "SlaveServ.h"
// callback functions for notification from the slave service

void taskStop ()
{ isTaskStop = 1;  DBUG_PRINT("taskStop");  }

void taskCtrl (const char *arg)
{ isTaskCtrl = 1;  DBUG_PRINT("taskCtrl arg=%s", arg);  }
// this is the exec loop on each task-thread
int taskExec (char *data, char *datares, char *userData)
{
  int isFinished = 0; DBUG_PRINT("task  started");
  // set callback functions
  q2adpz_slv_setcb_task_stop (taskStop);
  q2adpz_slv_setcb_task_ctrl (taskCtrl);
```

```
  DBUG_PRINT("input data '%s'", data);
  // start main task loop
  while (! isFinished) {
    //do some crunching of the data
    { ... if (ok) isFinished = 1; }
    //task needs to be stopped
    if (isTaskStop) { ... DBUG_PRINT("task stop executed"); break; }
    if (isTaskCtrl) {
        ... q2adpz_slv_task_status (task_ok, "task ctrl");
        DBUG_PRINT("info", ("task ctrl executed"); }
    //if crunching finished
    if (isFinished) {
        DBUG_PRINT("task finished res='%s'", datares); break; }
    } // while
}
```

4.3 Client

The client represents the interface for submitting jobs in the system. There are two execution modes for the client: a *batch mode* and an *interactive mode*. In the *batch mode*, a project file describes a job by specifying the required resources and how to start the tasks. This information is sent to the master, which is responsible for scheduling the tasks. The client can detach from the master and connect later for the results. Each project is described by using the XML language. In the *interactive mode*, the client remains connected to the master for the entire execution of the job. Also, the client can get direct connection to each of the slaves involved in the computation. The client has a lot of freedom over the creation and controlling of new tasks: it can dynamically create new tasks, send messages to the tasks already in execution, and receive feedback from the running tasks, either through the master node, or by means of direct communication with the slaves running the respective tasks. An example of a job description in XML is listed beneath.

```
<Job Name="executable_example">
  <Task ID="1" Type="Executable">
    <RunCount>3</RunCount>
    <DataPathPrefix>./datafiles/</DataPathPrefix>
    <FilesURL>http://www-data/qadpz/cgi-bin</FilesURL>
    <InputFile Constant="Yes">simple/source.txt</InputFile>
    <OutputFile>simple/dest.txt</OutputFile>
    <TaskInfo>
      <Memory Unit="MB">1</Memory>
      <Disk Unit="MB">1</Disk>
      <TimeOut>3600</TimeOut>
      <OS>Linux</OS>
      <CPU>i386</CPU>
      <URL>
        http://www-data/qadpz/app/lib/Linux/i386/libslv-app.so
      </URL>
      <Executable Type="File">simple</Executable>
    </TaskInfo>
    <TaskInfo>
      <Memory Unit="MB">1</Memory>
      <Disk Unit="MB">1</Disk>
      <TimeOut>3600</TimeOut>
      <OS>Win32</OS>
      <CPU>i386</CPU>
      <URL>
        http://www-data/qadpz/app/lib/Win32/i386/slv_app.dll
      </URL>
      <Executable Type="File">simple.exe</Executable>
    </TaskInfo>
    <TaskInfo>
      <Memory Unit="MB">1</Memory>
      <Disk Unit="MB">1</Disk>
      <TimeOut>3600</TimeOut>
      <OS>SunOS</OS>
      <CPU>sun4u</CPU>
      <URL>
        http://www-data/qadpz/app/lib/SunOS/sun4u/libslv-app.so
```

```
      </URL>
      <Executable Type="File">simple</Executable>
    </TaskInfo>
  </Task>
</Job>
```

4.4 Jobs, tasks and subtasks

The QADPZ users can submit, monitor, and control computing applications to be executed on the computers that share resources. Tasks can be binary programs, which can run on any of the sharing computers. A task comes in the form of an executable program, compiled for a specific architecture and operating system. For better performance, a task can be also in the form of a shared (dynamic) library, which can be more efficiently loaded by the slave program. As an alternative to native binary programs for a specific platform, a task can also be an interpreted or precompiled program. For example, it can be a compiled Java application or an interpreted program (e.g. Perl, Python), which further needs, respectively, a Java Virtual Machine or a specific interpreter, on the host computer.

Multiple tasks, which are related to each other, can be grouped into a job that is actually what a user submits to the system (see job life in Figure 3). A job can be composed of one or more tasks. Using jobs provides for easier structuring and management of the computational applications for both the user and the system. Each job is assigned uniquely to one user, however, a user can have multiple jobs submitted at the same time to the system. The tasks that correspond to a job can be independent or not at execution time. Tasks can further be divided into subtasks, consisting of finer work units that are executed within a task. Subtasks are used for interactive applications, which require permanent connection between a client and the slaves. They are usually generated at run-time at the client, and sent for execution to an already running task, which can solve them. The main reason for having subtasks is to improve the efficiency of smaller execution units without the overhead of starting a new task each time.

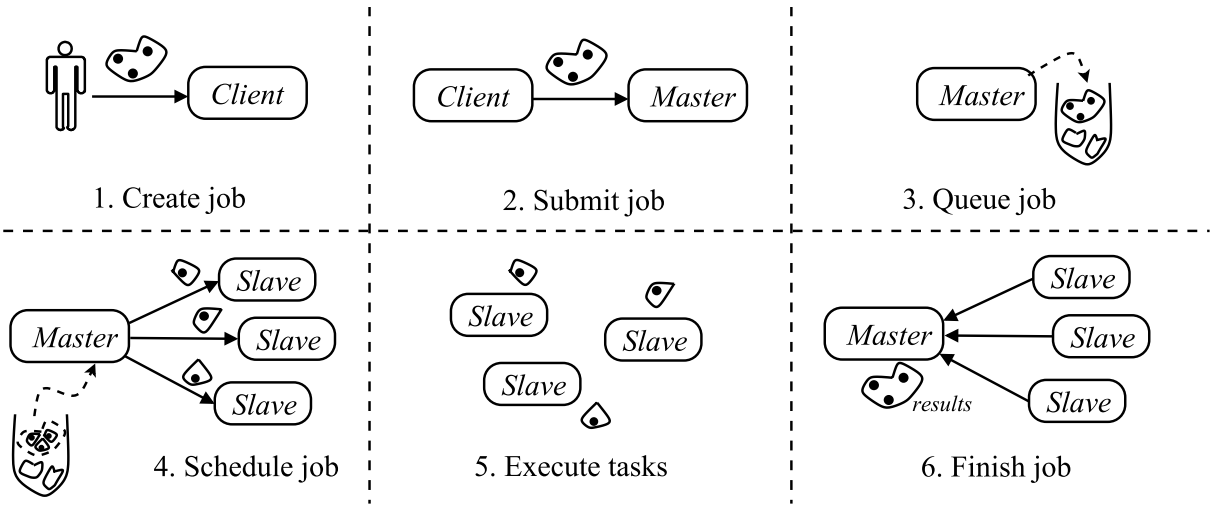


Figure 3: QADPZ job life

4.5 User interface

The QADPZ user interface provides for a user-friendly environment, in which the user can interact with the system. This interaction involves mainly the submission, the monitoring, and the management of the submitted computational applications, along with the resource monitoring and control. The first interface is the job-monitoring interface that is a web-based interface that provides detailed information about all existing jobs in the system. The user can browse the jobs, see their status, and view their component tasks. S/he can also easily create new jobs and tasks. Using this interface, each job can be stopped or deleted (Figure 4). The second interface is also web-based and provides information related to the resources in the system. Basically it gives a list of the slaves registered in the system and their current status (Figure 5). The owner of a desktop computer running a slave is given an interactive application,

which permits easy configuration of the slave. The user has complete control over the slave running on her computer.

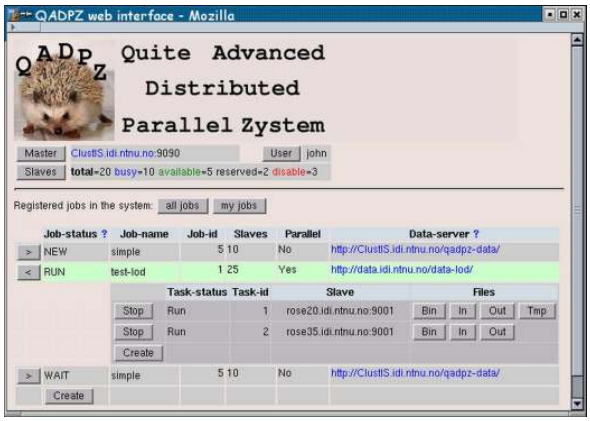


Figure 4: Job-monitoring web interface

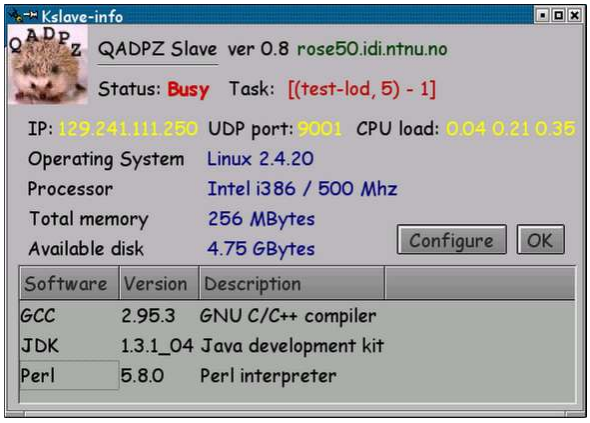


Figure 5: Slave information/configuration interface

5 Security

Because of the unreliability of the UDP protocol, which is our first option for the low-level communication protocol due to its benefits, it is not guaranteed that the execution tasks arriving to the slave computers are undoubtedly sent by the master. This is a serious security threat since it allows for a malicious hacker to submit any piece of code to the slave nodes (IPspoofing). For that reason, and on the cost of a decreased performance, all communication from clients to master and from master to slaves is encrypted and/or signed. Particularly, the data flow from client to master has to be authorized by the name and the password of a QADPZ user, and encrypted using a master public key. A master private key signs the data flow from master to slaves and the authenticity is verified using a master public key on slave nodes.

It is important to note that the data flow from slaves to master and from master to clients is neither encrypted nor signed, which means that a malicious hacker can monitor (packet sniffing) or alter (IP-spoofing) the data or control information arriving back to master or client nodes, and thus put the slave nodes and/or the master node out of operation, modify the resulted data that are submitted by the slaves, or do any other kind of harm to the computational process. In other words, the current QADPZ security scheme is designed to protect the security of the computers in the network, i.e. a malicious hacker cannot submit an alien piece of code to be executed instead of a user computational task. However, this scheme does not protect the QADPZ user data. We plan to provide optional data integrity in the future versions of the system.

Security of the system is handled in two ways. On the one hand, only registered users are allowed to submit applications for execution. This is done by using a user/password scheme, and allows a simple access control to the computational resources. The QADPZ system manages its own user database, completely independent of any of the underlying operating systems, thus simplifying users' access to the system. The QADPZ administrator can create new users by using some supporting tools. On the other hand, security involves the encryption of messages exchanged between various components of QADPZ. This is done by using public key encryption, and provides an additional level of protection against malicious attacks.

6 Conclusions and future work

The present paper reveals the development experience of QADPZ, a desktop grid computing environment. We summarized the main features of the system that make it a powerful platform for running computationally intensive applications. The reasons that have justified the endeavor of developing a new desktop grid platform are also presented. The QADPZ requirements have included all the core capabilities that a successful desktop grid system should provide [12]. We presented the detailed architecture of the system, along with some of the design details. When we started this work, our main goal was to build an easy to use, open source system that provides the complex functionality that users expect from

such a platform [4, 12]. It is worth mentioning that QADPZ has over a thousand users who have already downloaded it [20]. Many of them use it for their daily tasks and we have got valuable feedback from them [4].

Further on we present some future work ideas that aim to improve the QADPZ system:

- many areas of the QADPZ system are incomplete. For example, many large scale parallel problems require checkpointing: running a parallel application for hours or even days and loosing all results due to one failing node is unacceptable;
- data integrity is an important issue, especially in an open environment (Internet);
- improved support for user data security: computation results data can be encrypted and/or signed so that the user of the system can be sure the received data is correct;
- users could be provided with different scheduling algorithms to choose from, according to the type of their problem;
- more complete implementation of the MPI layer and development of a complete library of the collective communication routines;
- adding a set of transparent profiling tools for evaluating the performance of the different components, which is crucially important when running parallel applications;
- decentralizing the system by employing P2P services, which would permit to a group of users to form an ad-hoc set of shared resources; moving towards a P2P architecture;
- interconnection with a grid computing environment that must be decentralized, robust, highly available, and scalable [21], while efficiently mapping application instances to available resources in the system.

These future developments of QADPZ subscribe to the belief that the vision that motivates both Grid and P2P, i.e. that of "a worldwide computer within which access to resources and services can be negotiated as and when needed, will only become real if we are successful in developing a technology that combines important elements of P2P and Grid computing" [1].

Bibliography

[1] I. Foster, and A. Iamnitchi, *On death, taxes, and the convergence of peer-to-peer and grid computing*, in 2nd Int. Workshop on P2P Systems IPTPS 2003, pp. 118-128, 2003.

[2] I. Foster, C. Kesselman, *The grid: blueprint for a new computing infrastructure*, Boston: Morgan Kaufmann, 2004.

[3] J. I. Khan and A. Wierzbicki, Eds., *Foundation of Peer-to-Peer Computing*, Special Issue, Elsevier Journal of Computer Communication, Volume 31, Issue 2, Feb. 2008.

[4] Z. Constantinescu, *A Desktop Grid Computing Approach for Scientific Computing and Visualization*, PhD thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2008.

[5] QADPZ, [online] Available: http://qadpz.sourceforge.net. [Accessed August 1, 2008].

[6] M. Vladoiu, Z. Constantinescu, *An Extended Master-Worker Model for a Desktop Grid Computing Platform (QADPZ)*, in 3rd Int. Conference on Software and Data Technologies -ICSOFT 2008, pp. 169-174, 2008.

[7] Z. Constantinescu, J. Holmen, P. Petrovic, *Using Distributed Computing in Computational Fluid Dynamics*, in 15th Int. Conf. Parallel Computational Fluid Dynamics ParCFD-2003, pp. 123-129, 2003.

[8] Z. Constantinescu, *Towards an autonomic distributed computing environment*, in 14th Int. Workshop on Autonomic Computing Systems, 14th Int. Conf. on Database and Expert Systems Applications DEXA 2003, pp. 694-698, 2003.

[9] D. P. Anderson, J. Cobb , E. Korpela , M. Lebofsky, D. Werthimer, *SETIhome: an experiment in public-resource computing*, Communications of the ACM, vol. 45, no. 11, pp. 56-61, 2002.

[10] SETI@home [online] Available: setiathome.ssl.berkeley.edu [Accessed May 5, 2003].

[11] Distributed.net [online] Available: http://distributed.net. [Accessed May 5, 2008]

[12] M. Vladoiu, Z. Constantinescu, *A Taxonomy for Desktop Grids from Users Perspective*, in Int. Conference on Parallel and Distributed Computing - ICPDC 2008, World Congress on Engineering (WCE 2008), pp. 599-605, 2008.

[13] C. Germain, V. Neri, G. Fedak and F. Cappello, *XtremWeb: Building an Experimental Platform for Global Computing*, in 1st IEEE/ACM Workshop on Grid Computing Grid2000, pp. 91-101, 2000.

[14] D. P. Anderson, *BOINC: A System for Public-Resource Computing and Storage*, in 5th IEEE/ACM International Workshop on Grid Computing, pp. 365-372, 2004.

[15] BOINC - Open Source Software for Volunteer Computing and Grid Computing [online] Available: http://boinc.berkeley.edu. [Accessed November 25, 2007].

[16] J. Basney, M. Livny, *Managing network resources in Condor*, in Proc. of the 9th IEEE Symposium on High Performance Distributed Computing (HPDC9), pp. 298-299, 2000.

[17] Globus [online] Available: http://www.globus.org [Accessed May 15, 2008].

[18] I. Foster, and C. Kesselman., *Globus: A Metacomputing Infrastructure Toolkit*, Intl J. Supercomputer Applications, vol. 11, no. 2, pp. 115-128, 1997.

[19] J. Cassens, Z. Constantinescu, *Free Software: An Adequate Form of Software for Research and Education in Informatics?*, in LinuxTag 2003 Conference, pp. 5-10, 2003.

[20] Sourceforge, [online] Available: http://sourceforge.net [Accessed April 1, 2008].

[21] F. Berman, G. Fox, A.J.G. Hey, *Grid computing: making the global infrastructure a reality*, New York: J. Wiley, 2003.

Monica Vlădoiu got her MSc (1991) and PhD (2002) in the Department of Computer Science of The Polytechnic University of Bucharest, Romania. Since then, she has been with the Dept. of Informatics, Petroleum-Gas University of Ploieşti (UPG), Romania. Her main research interests include digital libraries, learning objects, multimedia databases, reflective and blended learning, desktop grid computing and e-society. She has published over 30 research papers concerning these topics and she has (co-) authored 3 books.

Zoran Constantinescu got his MSc (1997) in the Dept. of Computer Science of The Polytechnic University of Bucharest, Romania. Since then, he has been working both in the software engineering industry and in Higher Education. He got his doctoral degree in Computer Science (2008), from The Norwegian University of Science and Technology, Trondheim, Norway. His research interests include parallel and distributed computing, desktop grid computing, GPS systems and embedded systems. He has published over 20 research papers dealing with the above mentioned topics.

Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. IV (2009), No. 1, pp. 92-98

A Note on the Generative Power of Axon P Systems

Xingyi Zhang, Jun Wang, Linqiang Pan

Huazhong University of Science and Technology, Department of Control Science and Engineering
Key Laboratory of Image Processing and Intelligent Control
Wuhan 430074, Hubei, People's Republic of China
E-mail: xyzhanghust@gmail.com, junwangjf@gmail.com, lqpan@mail.hust.edu.cn

Abstract: Axon P systems are a class of spiking neural P systems. In this paper, the axon P systems are used as number generators and language generators. As a language generator, the relationships of the families of languages generated by axon P systems with finite and context-free languages are considered. As a number generator, a characterization of the family of finite sets can be obtained by axon P systems with only one node. The relationships of sets of numbers generated by axon P systems with semilinear sets of numbers are also investigated. This paper partially answers some open problems formulated by H. Chen, T.-O. Ishdorj and Gh. Păun.
Keywords: Membrane computing, SN P systems, Axon P systems

1 Introduction

The spiking neural P systems (in short, SN P systems) are a class of bio-inspired computing devices introduced in [6], which attempts to incorporate the idea of spiking neurons into the area of membrane computing. The resulting models are a variant of tissue-like and neural-like P systems, with specific ingredients and way of functioning inspired from spiking neurons. In SN P systems, the main "information-processor" is the neuron, while the axon is only a channel of communication without any other role – which is not exactly the case in neurobiology. So, recently, a special form of spiking neural P systems, called axon P systems, is introduced in [3], which corresponds to the activity of Ranvier nodes of neuron axon. Actually, axon P systems are a sort of linear SN P systems. Spikes are transmitted along the axon, to the left and to the right, from one node to another node, and an output is provided by the rightmost node. A symbol b_i is associated with a step when i spikes exit the system, in this way a string is associated with a computation. In [3], the language generating power of axon P systems under the above definition was investigated, and many open problems and research topics were formulated. In this paper, we continue the study of axon P systems, specifically, the number generative power and the language generative power are investigated, and in this context we answer some open problems formulated in [3].

SN P systems can be used as number generators (e.g., [2, 4, 6, 7]) or language generators (e.g., [1, 2, 5, 10, 12]). As a variant of SN P systems, axon P systems can also be used as number generators and language generators. As a number generator, we do not care whether or not the computation halts, but we only request that the output node spikes exactly twice during the computation, and the result of a computation is the number of steps elapsed between the two moments when the output node spikes. In this case, a characterization of the family of finite sets is given by axon P systems with one node. Also, the relationships of sets of numbers generated by axon P systems with semilinear sets of numbers are also investigated. As a language generator, each configuration is described by a corresponding string, and the result of a halting computation is defined as the strings associated with configurations where the system emits a spike. In this case, the relationships of the families of languages generated by axon P systems with finite and context-free languages are considered.

The paper is organized as follows. In Section 2, formal language theory prerequisites useful in the following sections are recalled. In Section 3, the definition of axon P systems and the problems considered in this paper are given. Axon P systems as number generators and language generators are investigated respectively in Section 4 and Section 5. Conclusions and remarks are drawn in Section 6.

2 Formal Language Theory Prerequisites

We assume the reader to be familiar with basic language and automata theory, as well as basic membrane computing [8] (for more updated information about membrane computing, please refer to [11]), so that we specify here only a few notations and basic definitions.

Let us start by mentioning the following convention: when comparing two generative or accepting devices, number zero is ignored (this corresponds to the usual convention in language theory of ignoring the empty string).

For an alphabet V, V^* denotes the set of all finite strings over V, with the empty string denoted by λ. The set of all nonempty strings over V is denoted by V^+. When $V = \{a\}$ is a singleton, then we write simply a^* and a^+ instead of $\{a\}^*$, $\{a\}^+$. The length of a string $x \in V^*$ is denoted by $|x|$. For a language $L \subseteq V^*$, the set $length(L) = \{|x| \mid x \in L\}$ is called the length set of L. The families of finite, regular, linear and context-free languages are denoted by FIN, REG, LIN, CF, respectively. The families of length sets of languages in FIN, REG, LIN and CF are denoted by $NFIN$, $NREG$, $NLIN$, NCF, respectively. The family of languages generated by oL systems is denoted by oL, and we add the letter E in front of oL if extended oL systems are used. We also denote by $SLIN_1$ the family of semilinear sets of numbers (the subscript indicates that we work with one-dimensional vectors, not with semilinear sets of vectors in general). It is known that the following equalities hold true: $NREG = NLIN = NCF = SLIN_1$ (see, e.g., [8]).

A regular expression over an alphabet V is defined as follows: (i) λ and each $a \in V$ is a regular expression, (ii) if E_1, E_2 are regular expressions over V, then $(E_1)(E_2)$, $(E_1) \cup (E_2)$, and $(E_1)^+$ are regular expressions over V, and (iii) nothing else is a regular expression over V. With each expression E we associate a language $L(E)$, defined in the following way: (i) $L(\lambda) = \{\lambda\}$ and $L(a) = \{a\}$, for all $a \in V$, (ii) $L((E_1) \cup (E_2)) = L(E_1) \cup L(E_2)$, $L((E_1)(E_2)) = L(E_1)L(E_2)$, and $L((E_1)^+) = L(E_1)^+$, for all regular expressions E_1, E_2 over V. Non-necessary parentheses are omitted when writing a regular expression, and also $(E)^+ \cup \{\lambda\}$ can be written as E^*.

A Chomsky grammar is given in the form $G = (N, T, S, P)$, with N being the nonterminal alphabet, T the terminal alphabet, $S \in N$ the axiom, and P is the finite set of productions. For regular grammars, the productions are of the form $u \to v$, for some $u \in N, v \in T \cup TN$ (in regular grammars, we also allow productions of the form $u \to \lambda$, but only when this is useful for simplifying the grammar: because of the convention that the empty string is not counted when comparing the languages generated by two grammars, such productions are not necessary in regular grammars).

3 Axon P Systems

We now introduce the axon P systems.

An axon P system of degree $m \geq 1$ is a construct of the form

$$\Pi = (O, \rho_1, \ldots, \rho_m),$$

where:

1. $O = \{a\}$ is the singleton alphabet (a is called spike);
2. $\rho_1, \ldots, \rho_m$ are (Ranvier) nodes, of the form

$$\rho_i = (n_i, R_i), 1 \leq i \leq m,$$

where:

a) $n_i \geq 0$ is the initial number of spikes contained in ρ_i;

b) R_i is a finite set of rules of the form $E/a^c \to (a^l, a^r)$, where E is a regular expression over a, $c \geq 1$, and $l, r \geq 0$, with the restriction that R_1 contains only rules with $l = 0$.

The nodes are arranged along an axon in the order $\rho_1, \ldots, \rho_m$, with ρ_m at the end of the axon; this means that node ρ_m is the output node of the system.

A rule $E/a^c \to (a^l, a^r) \in R_i$ is used as follows. If the node ρ_i contains k spikes, and $a^k \in L(E), k \geq c$, then the rule can be applied, and this means consuming (removing) c spikes from ρ_i (thus only $k - c$ spikes remain in ρ_i), and it sends l spikes to its left hand neighbor and r spikes to its right hand neighbor; the first node, ρ_1 does not send spikes to the left, while in the case of the rightmost node, ρ_m, the spikes sent to the right are "lost" in the environment. A global clock is assumed, marking the time for the whole system, hence the functioning of the system is synchronized.

If a rule $E/a^c \to (a^l, a^r)$ has $E = a^c$, then we will write it in the simplified form $a^c \to (a^l, a^r)$.

If several rules can be used at the same time, then the one to be applied is chosen non-deterministically.

During the computation, a configuration of the system is described by the number of spikes present in each node; thus, the initial configuration is $\langle n_1, \ldots, n_m \rangle$.

Using the rules as described above, one can define transitions among configurations. A transition between two configurations C_1, C_2 is denoted by $C_1 \Rightarrow C_2$. If $C_1 = \langle k_1, k_2, \ldots, k_m \rangle$, then $C_2 = \langle k'_1, k'_2, \ldots, k'_m \rangle$, where $k'_i = (k_i - c_i) + r_{i-1} + l_{i+1}$, i is the current node, c_i is number of spikes consumed in this node, and r_{i-1}, l_{i+1} are the numbers of spikes sent to this node by the neighboring nodes, $2 \leq i \leq m-1$. In the case of $i = 1$ or $i = m$, there is only one neighbor. Any sequence of transitions starting in the initial configuration is called a computation. A computation halts if it reaches a configuration where no rule can be used.

In this paper, we consider the following two ways of defining the result of a computation for axon P systems.

(i) Similar to [6] and [10], we do not care whether or not the computation halts, but we only request that the output node spikes exactly twice during the computation. Then, the number of steps elapsed between the two spikes is the number computed by the axon P system along that computation. We denote by $N_2(\Pi)$ the set of numbers computed in this way by the axon P system Π, and by $Spik_2AP_m(rule_k, cons_p)$ the family of sets $N_2(\Pi)$ generated by axon P systems with at most m nodes, at most k rules in each node, each rule consuming at most p spikes. As usual, any of these parameters is replaced by $*$ if it is not bounded.

(ii) As formulated in [3], a language is associated with a computation of axon P systems in the following way: for each node ρ_i we consider a symbol c_i and a configuration $\langle k_1, \ldots, k_m \rangle$ is described by the string $c_1^{k_1} \ldots c_m^{k_m}$; then, the result of a halting computation is defined as the strings associated with configurations where the system emits a spike. All these strings form the language generated by the system. We denote by $L(\Pi)$ the language generated in this way by the axon P system Π, and by $LAP_m(rule_k, cons_p)$ the family of languages $L(\Pi)$ with m, k, p having the same meaning as above.

4 Axon P Systems as Number Generators

In this section, we investigate the number generative power of axon P systems.

4.1 A Characterization of $NFIN$

In [6], it has been proved that SN P systems can characterize $NFIN$ by using only one neuron. For axon P systems, we have a similar result.

Theorem 1. $NFIN = Spik_2AP_1(rule_*, cons_*)$.

Proof. The inclusion $Spik_2AP_1(rule_*, cons_*) \subseteq NFIN$ can be easily proved: in each step, the number of spikes in an axon P system with only one node decreases by at least one, hence any computation lasts at most as many steps as the number of spikes present in the system at the beginning. Thus, axon P systems with only one node can only compute finite sets of numbers.

To prove the opposite inclusion $NFIN \subseteq Spik_2AP_1(rule_*, cons_*)$, let us take a finite set of numbers, $F = \{n_1, \ldots, n_k\}$, and we assume that we have $n_1 < n_2 < \ldots < n_k$.

An axon P system that generates F is shown in Figure 1.

Initially, the node contains $n_k + 1$ spikes, hence, only the rule $a^{n_k+1}/a \to (\lambda, a)$ can be used in the first step. It consumes one spike and immediately sends a spike to the environment. In the next step, we have to continue with rules $a^{n_k+1-t}/a \to (\lambda, \lambda)$, for $t = 1$, and then for the respective $t = 2, \ldots, n_1$. But for $t = n_1$ (in step $n_1 + 1$), there is another rule $a^{n_k+1-t} \to (\lambda, a)$ which can be used, non-deterministically chosen. If we choose the second rule, the system will send a spike to the environment again and then the computation halts. Therefore, the number n_1 will be generated. If we choose the first rule, the process will continue, and in a similar way the numbers $n_2, \ldots, n_{k-1}$ can be generated in turn. In step $n_k + 1$ (the last step) only the rule $a^{n_k+1-t} \to (\lambda, a)$ with $t = n_k$ can be used, therefore, the number n_k can be generated and this concludes the computation.

Consequently, all the numbers in F can be generated by the above axon P system, which concludes the proof. □

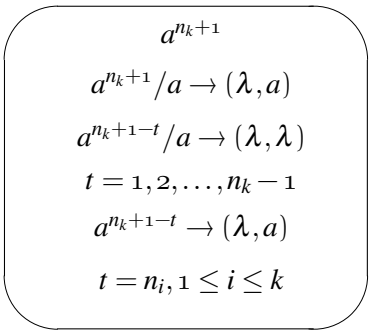


Figure 1: An axon P system generating a finite set of numbers

4.2 Relationships with Semilinear Sets of Numbers

As mentioned in Section 4.1, SN P systems with one neuron generate exactly the family of finite sets of numbers. Actually, SN P systems with two neurons also characterize the family of finite sets. So, the following results on axon P systems are unexpected: such systems with two nodes generate all semilinear sets of numbers.

Theorem 2. $SLIN_1 \subseteq Spik_2AP_2(rule_*, cons_*)$.

Proof. Consider a regular grammar $G = (N, T, S, P)$ with $N = \{A_1, A_2, \ldots, A_n\}, n \geq 1$, $S = A_n$, $T = \{b_1, b_2, \ldots, b_s\}$, and the productions in P are of the forms $A_i \to b_k A_j$, $A_i \to b_k$, $1 \leq i, j \leq n$, $1 \leq k \leq s$.

Then $length(L(G))$ can be generated by an axon P system as shown in Figure 2.

In the initial configuration we have $n+1$ spikes in node ρ_1 and $2n+1$ spikes in node ρ_2, therefore, in the first step, only the rule $a^{2n+1}/a \to (\lambda, a)$ can be used in node ρ_2, and the rule $a^{n+1}/a \to (\lambda, \lambda)$ is used in node ρ_1. Thus, a spike is sent to the environment and n spikes remain in node ρ_1 and $2n$ spikes in node ρ_2. In the next step, node ρ_2 fires by a rule $a^{n+i}/a^{n+i-j} \to (a^n, \lambda)$ or $a^{n+i} \to (\lambda, a)$ associated with a production $A_n \to b_k A_j$ or $A_n \to b_k$ from P, for $i = n$. If the second rule is used, another spike is sent to the environment and the computation halts. In this way, the generated number is 1. If the first rule is used, $2n - j$ spikes are consumed from node ρ_2. In this step node ρ_1 also fires and sends n spikes to node ρ_2. It will send n spikes back to node ρ_2 as long as it receives n spikes from node ρ_2.

Assume that in some step t, the rule $a^{n+i}/a^{n+i-j} \to (a^n, \lambda)$, for $A_i \to b_k A_j$, or $a^{n+i} \to (\lambda, a)$, for $A_i \to b_k$, is used, for some $1 \leq i \leq n$, and n spikes are received from node ρ_1.

If the first rule is used, then n spikes are sent to node ρ_1, $n + i - j$ spikes are consumed, and j spikes remain in node ρ_2. Then in step $t+1$, we have $n+j$ spikes in node ρ_2, and a rule for $A_i \to b_k A_j$ or $A_i \to b_k$ can be used. In step $t+1$ node ρ_2 also receives n spikes. In this way, the computation continues.

If the second rule is used, then no spike is sent to node ρ_1, all spikes in node ρ_2 are consumed, and n spikes are received from node ρ_1, which will remain in node ρ_2 forever. Moreover, in this step another spike is sent to the environment again. Then the computation halts.

We know that, if we use a production $A_i \to b_k A_j$ or $A_i \to b_k$ one time, then the length of a string generated by the grammar G increases by one, and this corresponds to the number of steps increasing by one by using the associated rule $a^{n+i}/a^{n+i-j} \to (a^n, \lambda)$ or $a^{n+i} \to (\lambda, a)$ one time in the axon P system. Moreover, the second spike is sent to the environment and the computation halts whenever a rule $a^{n+i} \to (\lambda, a)$ is used, which corresponds to a production $A_i \to b_k$ being used in G.

Therefore, the length set of all the strings in $L(G)$ can be generated, and the proof is complete. □

The inclusion in the previous theorem is proper.

Theorem 3. $Spik_2AP_2(rule_5, cons_6) - SLIN_1 \neq \emptyset$.

Proof. Let us consider the axon P system from Figure 3. In the initial configuration, only node ρ_1 contains 5 spikes, hence it fires in the first step by the rule $a(a^4)^+/a^4 \to (\lambda, a^{16})$ and sends 16 spikes to node ρ_2. In the next step, the rule $a \to (a, \lambda)$ or $a \to (\lambda, a^3)$ can be used in node ρ_1, non-deterministically chosen.

If we use the rule $a \to (a, \lambda)$, then we get a number of spikes of the form $4k+1$ in the second node, hence the first rule $a(a^4)^+/a^4 \to (a^4, \lambda)$ in node ρ_2 is applied as many times as possible, thus returning

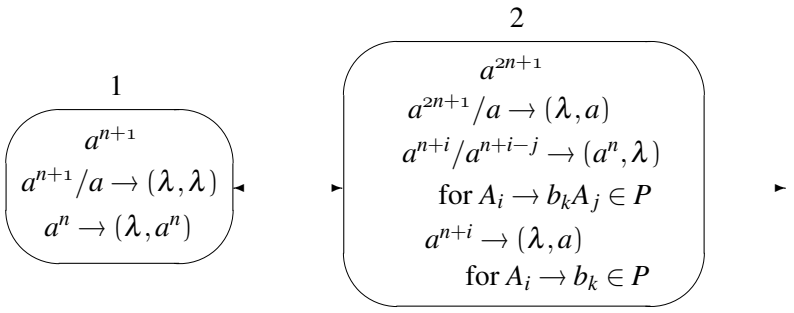


Figure 2: An axon P system generating a semilinear set

the spikes to node ρ_1. To end this returning, we have to use the rule $a \to (a, \lambda)$ in node ρ_2, which makes again the number of spikes from node ρ_1 be of the form $4k+1$ (note that no rule can be applied in any node when the number of spikes is multiple of 4). This process can be iterated any number of times, thus multiplying by 4 the number of spikes present in node ρ_1.

Assume that we have a configuration $\langle 4^n + 1, 0 \rangle$ in step t, for some $n \geq 1$; initially, this is the case, with $n = 1$. In node ρ_1, the rule $a(a^4)^+/a^4 \to (\lambda, a^{16})$ can be used as many times as possible until the node remains with only one spike. It moves all spikes to the second node, multiplied by 4. Therefore, in step $t + 4^{n-1} + 1$, we have a configuration $\langle 1, 4^{n+1} \rangle$. In the next step, node ρ_1 can also use the rule $a \to (\lambda, a^3)$ instead of $a \to (a, \lambda)$. Then the number of spikes from node ρ_2 is of the form $4k+3$ ($4^{n+1}+3$ spikes), hence the rule $a^3(a^4)^+/a \to (\lambda, a)$ should be applied in step $t + 4^{n-1} + 3$ and a spike is sent to the environment. At the same time, the number of spikes decreases by one and will be of the form $4k+2$ ($4^{n+1}+2$ spikes). Therefore, the rule $a^2(a^4)^+/a^4 \to (\lambda, \lambda)$ should be applied in step $t + 4^{n-1} + 4$. This rule does not change the form of the number of spikes and it remains in the form of $4k+2$, hence it is used as many times as possible. In this way, no spike is sent to the environment until the number of spikes from ρ_2 becomes 6 and this process needs $4^n - 1$ steps. Then, in the step $t + 4^{n-1} + 4^n + 3$, the rule $a^6 \to (\lambda, a)$ should be used and the second spike is sent to the environment. Therefore, the number of steps elapsed between the two spikes which were sent to the environment is 4^n and the computation halts.

Consequently, the set $\{4^n \mid n \geq 1\}$ can be generated by this system, which, obviously, is not a semilinear set. □

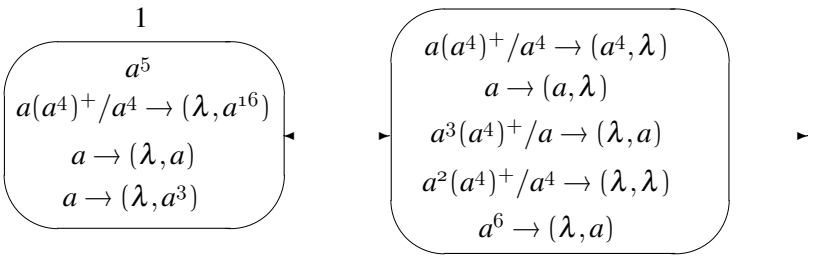


Figure 3: An axon P system generating a non-semilinear set $\{4^n | n \geq 1\}$

In [6], it has been shown that semilinear sets of numbers can be characterized by SN P systems with a bound on the number of spikes present in any neuron, but the number of neurons is not bounded. The following theorem also gives a characterization of semilinear sets of numbers, but here only two nodes are used in the axon P systems.

Theorem 4. *$SLIN_1 = Spik_2AP_2(rule_*, cons_*, bound_*)$, where $bound_*$ indicates that axon P systems have a bound on the number of spikes present in any node, but this bound is not specified.*

Proof. From Theorem 2, it is enough to prove the inclusion $Spik_2AP_2(rule_*, cons_*, bound_*) \subseteq SLIN_1$.
The proof is similar to the one of Lemma 9.1 in [6], so we do not recall it here. □

5 Axon P Systems as Language Generators

We now pass to considering the language generative power of axon P systems.

All strings generated by an axon P system in the way mentioned in Section 3 are of the form $c_1^{k_1}c_2^{k_2}\dots c_m^{k_m}$, where $\langle k_1,k_2,\dots,k_m\rangle$ is a configuration of the system. Therefore, if there is a bound on the number of spikes in any node of axon P systems, we can directly have the following result.

Remark 5. $LAP_n(rule_*,cons_*,bound_*) \subseteq FIN$, for $n \geq 1$.

Moreover, as a direct consequence of this restrictive way of defining the languages associated with axon P systems, we cannot find a characterization of finite languages for the general axon P systems. For instance, it is not difficult to find that the string $c_1c_2c_1$ cannot be generated by any axon P system.

5.1 Beyond *CF*

Theorem 6. *There exists at least a language $L \in E0L-CF$, which can be generated by axon P systems.*

Proof. Let us consider the language $L=\{c_1^{2n}c_2^{2n}c_3^{2n} \mid n \geq 1\}$; it is obvious that $L \in E0L-CF$.

We construct the axon P system whose initial configuration is shown in Figure 4.

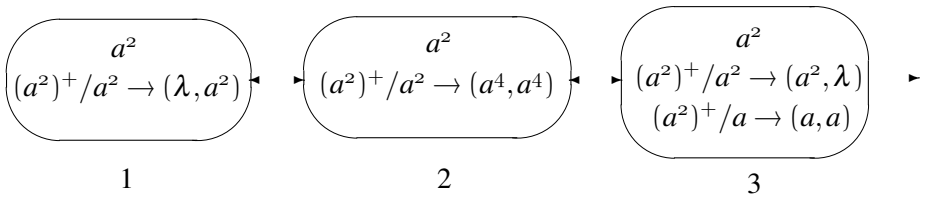


Figure 4: An axon P system generating the language $\{c_1^{2n}c_2^{2n}c_3^{2n} \mid n \geq 1\}$

Initially, each node of the system contains two spikes. Hence, each node can be activated, and in node ρ_3 both the rule $(a^2)^+/a^2 \to (a^2,\lambda)$ and the rule $(a^2)^+/a \to (a,a)$ can be applied, non-deterministically chosen. If the first rule is applied, then the two spikes in each node are consumed and no spike is sent to the environment. At the same time, each node accumulates 4 spikes (the 4 spikes in nodes ρ_1 and ρ_3 are received from node ρ_2; the 4 spikes in node ρ_2 are received from both node ρ_1 and node ρ_3, two from node ρ_1 and two from node ρ_3). In this way, all the nodes can be activated in the next step. If in node ρ_3 we continue using the rule $(a^2)^+/a^2 \to (a^2,\lambda)$, then in a similar way each node obtains 6 spikes. This process can be iterated until the second rule is applied. In this case, the number of spikes in each node increases by two in each step. Therefore, each node of the system accumulates $2n$ spikes in step n, for $n \geq 1$.

At any moment, we can also apply the second rule in node ρ_3. Assume that it is applied in step $n+1$, then node ρ_3 sends a spike to the environment, which means that the string $c_1^{2n}c_2^{2n}c_3^{2n}$ is generated by this system. At the same time, node ρ_1 accumulates $2n+2$ spikes, node ρ_2 accumulates $2n+1$ spikes, and node ρ_3 accumulates $2n+3$ spikes. In the next step, only node ρ_1 can be activated. Thus, it consumes two spikes and sends two spikes to node ρ_2. This process can be iterated until all the spikes in node ρ_1 are moved to node ρ_2, and then the computation halts.

Therefore, the language $\{c_1^{2n}c_2^{2n}c_3^{2n} \mid n \geq 1\}$ can be generated by the axon P system and this concludes the proof. □

6 Conclusions and Remarks

In this paper, the number generative power and the language generative power of axon P systems are investigated. As a variant of SN P systems, there remain many open problems and research topics about axon P systems to be considered. One important aspect is suggested by the research about SN P systems. For instance, as usual in SN P systems, an arbitrary delay can be considered in axon P systems. What about considering the spike trains (finite or infinite) themselves as the result of a computation in axon P systems, as investigated in [10]. The various applications for axon P systems also deserve to be considered.

As suggested by Gheorghe Păun in [9], a more general and probably more interesting problem is combining neurons and axons in a global model; maybe also astrocytes can be added, thus obtaining a

more complex model, closer to reality. Please refer to [3, 9] for more open problems and research topics related to axon P systems.

Acknowledgements

Comments and suggestions from Gheorghe Păun are greatly acknowledged. The project was supported by National Natural Science Foundation of China (Grant Nos. 60674106, 30870826, 60703047, 60503002, and 60533010), 863 Program of China (2006AA01Z104), Program for New Century Excellent Talents in University (NCET-05-0612), Ph.D. Programs Foundation of Ministry of Education of China (20060487014), Chenguang Program of Wuhan (200750731262), and HUST-SRF (2007Z015A).

Bibliography

[1] H.M. Chen, M. Ionescu, M. J. Pérez-Jiménez, R. Freund, and Gh. Păun, On String Languages Generated by Spiking Neural P Systems, *Fundamenta Informaticae*, Vol. 75(1–4), pp. 141–162, 2007.

[2] H. M. Chen, M. Ionescu, T.-O. Ishdorj, A. Păun, Gh. Păun and M. J. Pérez-Jiménez, Spiking Neural P Systems with Extended Rules, in: M. A. Gutiérrez-Naranjo, Gh. Păun, A. Riscos-Núñez, F. J. Romero-Campero, eds., *Fourth Brainstorming Week on Membrane Computing*, vol. I, RGNC Report 02/2006, Research Group on Natural Computing, Sevilla University, Fénix Editora, pp. 241–266, 2006.

[3] H. M. Chen, T.-O. Ishdorj and Gh. Păun, Computing Along the Axon, *Progress in Natual Science*, vol. 17(4), pp. 417–423, 2007.

[4] O. H. Ibarra, S. Woodworth, F. Yu and A. Păun, On Spiking Neural P Systems and Partially Blind Counter Machines, *Natural Computing*, Vol. 7(1), pp. 3–19, 2008.

[5] O. H. Ibarra and S. Woodworth, Characterizing Regular Languages by Spiking Neural P Systems, *International Journal of Foundations of Computer Science*, Vol. 18(6), pp. 1247–1256, 2007.

[6] M. Ionescu, Gh. Păun and T. Yokomori, Spiking Neural P Systems, *Fundamenta Informaticae*, Vol. 71(2–3), pp. 279–308, 2006.

[7] M. Ionescu, Gh. Păun and T. Yokomori, Spiking Neural P Systems with Exhaustive Use of Rules, *International Journal of Unconventional Computing*, Vol. 3(2), pp. 135–154, 2007.

[8] Gh. Păun, *Membrane Computing – An Introduction*, Springer-Verlag, Berlin, 2002.

[9] Gh. Păun, Twenty Six Research Topics about Spiking Neural P Systems, in: M. A. Gutiérrez-Naranjo, Gh. Păun, A. Romero-Jiménez, A. Riscos-Núñez, eds., *Fifth Brainstorming Week on Membrane Computing*, RGNC Report 01/2007, Research Group on Natural Computing, Sevilla University, Fénix Editora, pp. 263–280, 2007.

[10] Gh. Păun, M. J. Pérez-Jiménez and G. Rozenberg, Spike Trains in Spiking Neural P Systems, *International Journal of Foundations of Computer Science*, Vol. 17(4), 975–1002, 2006.

[11] The P System Web Page: http://ppage.psystems.eu

[12] X. Y. Zhang, X. X. Zeng and L. Q. Pan, On String Languages Generated by Spiking Neural P Systems with Exhaustive Use of Rules, *Natural Computing*, to appear.

Xingyi Zhang was born in China on June 6, 1982. He received his master degree in applied mathematics from Huazhong University of Science and Technology in 2006. Currently, his main research fields are formal language theory and its applications, unconventional models of computation, especially, membrane computing. He has published several scientific papers in international journals.

Linqiang Pan was born in Zhejiang, China on November 22, 1972. He got PhD at Nanjing University in 2000. Since 2004, he is a professor at Huazhong University of Science and Technology, China. His main research fields are graph theory and membrane computing.

Author index

Description

International Journal of Computers, Communications & Control (IJCCC) is a quarterly peer-reviewed publication started in 2006 by Agora University Editing House - CCC Publications, Oradea, ROMANIA.

Beginning with 2007, EBSCO Publishing is a licensed partner of IJCCC Publisher.

Every issue is published in online format (ISSN 1841-9844) and print format (ISSN 1841-9836).

Now we offer free online access to the full text of all published papers.

The printed version of the journal should be ordered, by subscription, and will be delivered by regular mail.

IJCCC is directed to the international communities of scientific researchers from the universities, research units and industry.

IJCCC publishes original and recent scientific contributions in the following fields:

- Computing & Computational Mathematics
- Information Technology & Communications
- Computer-based Control

To differentiate from other similar journals, the editorial policy of IJCCC encourages especially the publishing of scientific papers that focus on the convergence of the 3 "C" (Computing, Communication, Control).

The articles submitted to IJCCC must be original and previously unpublished in other journals. The submissions will be revised independently by minimum two reviewers and will be published only after end of the editorial workflow.

The peer-review process is single blinded: the reviewers know who the authors of the manuscript are, but the authors do not have access to the information of who the peer-reviewers are.

IJCCC also publishes:

- papers dedicated to the works and life of some remarkable personalities;
- reviews of some recent important published books

Also, IJCCC will publish as supplementary issues the proceedings of some international conferences or symposiums on Computers, Communications and Control, scientific events that have reviewers and program committee.

The authors are kindly asked to observe the rules for typesetting and submitting described in Instructions for Authors.

Thomson Reuters Subject Category of IJCCC:
AUTOMATION & CONTROL SYSTEMS
Category Description: Automation & Control Systems covers resources on the design and development of processes and systems that minimize the necessity of human intervention. Resources in this category cover control theory, control engineering, and laboratory and manufacturing automation.
COMPUTER SCIENCE, INFORMATION SYSTEMS
Category Description: Computer Science, Information Systems covers resources that focus on the acquisition, processing, storage, management, and dissemination of electronic information that can be read by humans, machines, or both. This category also includes resources for telecommunications systems and discipline-specific subjects such as medical informatics, chemical information processing systems, geographical information systems, and some library science.

Editorial Workflow

The editorial workflow is performed using the online Submission System.

The peer-review process is single blinded: the reviewers know who the authors of the manuscript are, but the authors do not have access to the information of who the peer-reviewers are.

The following is the editorial workflow that every manuscript submitted to the IJCCC during the course of the peer-review process.

Every IJCCC submitted manuscript is inspected by the Editor-in-Chief/Associate Editor-in-Chief. If the Editor-in-Chief/Associate Editor-in-Chief determines that the manuscript is not of sufficient quality to go through the normal review process or if the subject of the manuscript is not appropriate to the journal scope, Editor-in-Chief/Associate Editor-in-Chief *rejects the manuscript with no further processing*.

If the Editor-in-Chief/Associate Editor-in-Chief determines that the submitted manuscript is of sufficient quality and falls within the scope of the journal, he sends the manuscript to the IJCCC Executive Editor/Associate Executive Editor, who manages the peer-review process for the manuscript.

The Executive Editor/Associate Executive Editor can decide, after inspecting the submitted manuscript, that it should be rejected without further processing. Otherwise, the Executive Editor/Associate Executive Editor assigns the manuscript to the one of Associate Editors.

The Associate Editor can decide, after inspecting the submitted manuscript, that it should be rejected without further processing. Otherwise, the Associate Editor assigns the manuscript to minimum two external reviewers for peer-review. These external reviewers may or may not be from the list of potential reviewers of IJCCC database.

The reviewers submit their reports on the manuscripts along with their recommendation of one of the following actions to the Associate Editor: Publish Unaltered; *Publish after Minor Changes; Review Again after Major Changes; Reject* (Manuscript is flawed or not sufficiently novel).

When all reviewers have submitted their reports, the Associate Editor can make one of the following editorial recommendations to the Executive Editor: Publish Unaltered; Publish after Minor Changes; Review Again after Major Changes; Reject.

If the Associate Editor recommends *"Publish Unaltered"*, the Executive Editor/Associate Executive Editor is notified so he/she can inspect the manuscript and the review reports. The Executive Editor/Associate Executive Editor can either override the Associate Editor's recommendation in which case the manuscript is rejected or approve the Associate Editor's recommendation in which case the manuscript is accepted for publication.

If the Associate Editor recommends "Review Again after Minor Changes", the Executive Editor/Associate Executive Editor is notified of the recommendation so he/she can inspect the manuscript and the review reports.

If the Executive Editor/Associate Executive Editor overrides the Associate Editor's recommendation, the manuscript is rejected. If the Executive Editor approves the Associate Editor's recommendation, the authors are notified to prepare and submit a final copy of their manuscript with the required minor changes suggested by the reviewers. Only the Associate Editor, and not the external reviewers, reviews the revised manuscript after the minor changes have been made by the authors. Once the Associate Editor is satisfied with the final manuscript, the manuscript can be accepted.

If the Associate Editor recommends *"Review Again after Major Changes"*, the recommendation is communicated to the authors. The authors are expected to revise their manuscripts in accordance with the changes recommended by the reviewers and to submit their revised manuscript in a timely manner. Once the revised manuscript is submitted, the original reviewers are contacted with a request to review the revised version of the manuscript. Along with their review reports on the revised manuscript, the reviewers make a recommendation which can be "Publish Unaltered" or "Publish after Minor Changes" or "Reject". The Associate Editor can then make an editorial recommendation which can be "Publish Unaltered" or "Review Again after Minor Changes" or "Reject".

If the Associate Editor recommends rejecting the manuscript, either after the first or the second round of reviews, the rejection is immediate.

Only the Associate Editor-in-Chief can approve a manuscript for publication, where Executive Editor/Associate Executive Editor recommends manuscripts for acceptance to the Editor-in-Chief/Associate Editor-in-Chief.

Finally, recommendation of acceptance, proposed by the Associate Editor Chief, has to be approved by the Editor-in-Chief before publication.

Instructions for authors

Concurrent/Duplicate Submission

Submissions to IJCCC must represent original material.

Papers are accepted for review with the understanding that the same work has been neither submitted to, nor published in, another journal or conference. If it is determined that a paper has already appeared in anything more than a conference proceeding, or appears in or will appear in any other publication before the editorial process at IJCCC is completed, the paper will be automatically rejected.

Papers previously published in conference proceedings, digests, preprints, or records are eligible for consideration provided that the papers have undergone substantial revision, and that the author informs the IJCCC editor at the time of submission.

Concurrent submission to IJCCC and other publications is viewed as a serious breach of ethics and, if detected, will result in immediate rejection of the submission.

Preliminary/Conference Version(s)

If any portion of your submission has previously appeared in or will appear in a conference proceeding, you should notify us at the time of submitting, make sure that the submission references the conference publication, and supply a copy of the conference version(s) to our office. Please also provide a brief description of the differences between the submitted manuscript and the preliminary version(s).

Please be aware that editors and reviewers are required to check the submitted manuscript to determine whether a sufficient amount of new material has been added to warrant publication in IJCCC. If you have used your own previously published material as a basis for a new submission, then you are required to cite the previous work(s) and clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Any manuscript not meeting these criteria will be rejected. Copies of any previously published work affiliated with the new submission must also be included as supportive documentation upon submission.

Manuscript Preparing/Submission

The papers must be prepared using a LaTeX typesetting system. A template for preparing the papers is available on the journal website. In the template.tex file you will find instructions that will help you prepare the source file. Please, read carefully those instructions. (We are using MiKTeX 2.7).

Any graphics or pictures must be saved in Encapsulated PostScript (.eps) format.

Papers must be submitted electronically to the following email address: ccc.journal@gmail.com. You should send us the LaTeX source file (just one file - do not use bib files) and the graphics in a separate folder. You must send us also the pdf version of your paper.

The maximum number of pages of one article is 20. The publishing of a 12 page article is free of charge (including a bio-sketch). For each supplementary page there is a fee of 50 Euro/page that must be paid after receiving the acceptance for publication. The authors do not receive a print copy of the journal/paper, but the authors receive by email a copy of published paper in pdf format.

The papers must be written in English. The first page of the paper must contain title of the paper, name of author(s), an abstract of about 300 words and 3-5 keywords. The name, affiliation (institution and department), regular mailing address and email of the author(s) should be filled in at the end of the paper. Manuscripts must be accompanied by a signed copyright transfer form. The copyright transfer form is available on the journal website.

Please note: We will appreciate if the authors writes their own final version of the paper in LaTeX. But if the authors have difficulties with LaTeX and wish to send us their manuscript in Microsoft Word, the technical secretariat can do the transcription of the document. In this case, the paper can be sent in MS Word format with the following specifications: paper A4, font TNR 12p, single column. The graphics will be placed in the document but it has to be also attached separately in jpeg format.

Checklist:

1. Completed copyright transfer form.
2. Source (input) files.
    - One LaTeX file for the text.
    - EPS files for figures in a separate folder.
3. Final PDF file (for reference).