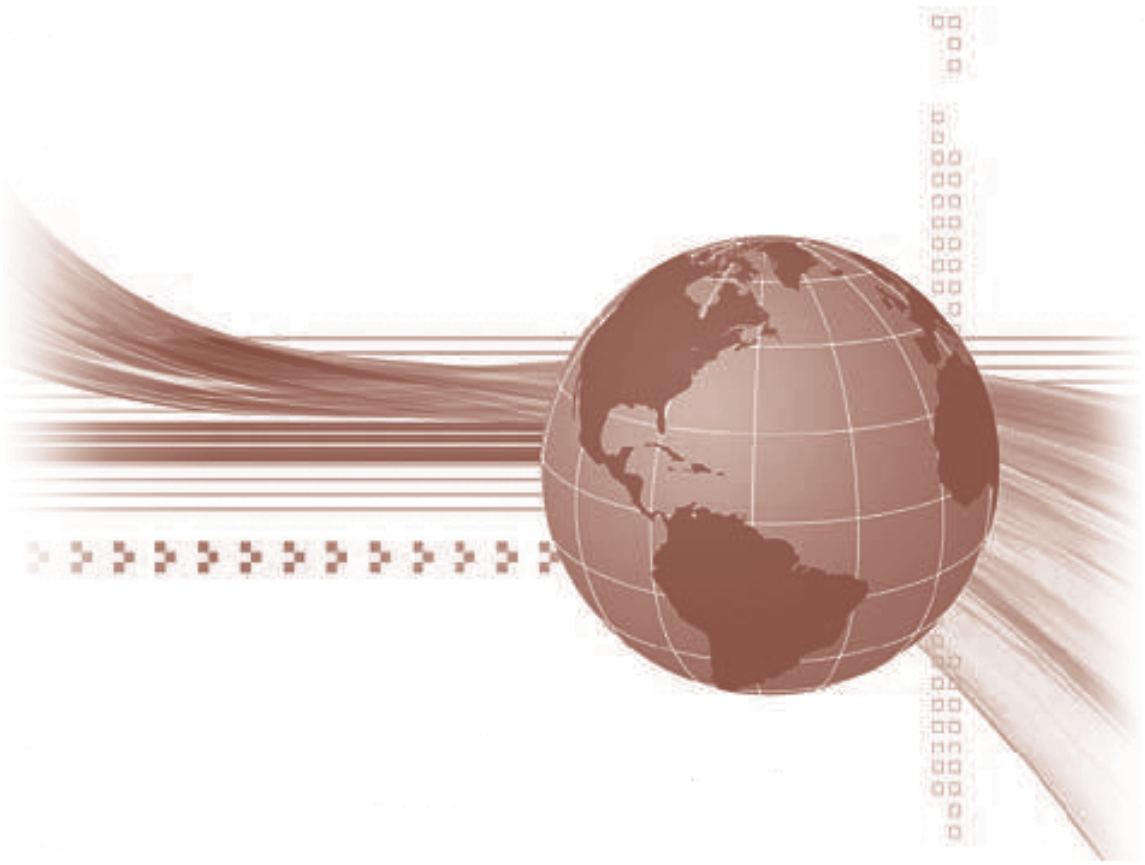
STUDIA UNIVERSITATIS
BABEŞ-BOLYAI



INFORMATICA

4/2010

Anul LV **2010**

S T U D I A
UNIVERSITATIS BABEŞ-BOLYAI

INFORMATICA

4

Redacția: M. Kogălniceanu 1 • 400084 Cluj-Napoca • Tel: 0264.405300

SUMAR – CONTENTS – SOMMAIRE

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

EDITORIAL: COMMON MISTAKES IN WRITING A SCIENTIFIC PAPER

MILITON FRENŢIU AND HORIA F. POP

1. Introduction

As editors of Studia UBB Informatica we have noticed that many papers contains some frequent errors. The best way to change this situation, to improve the quality of Studia papers, is to underline these frequent errors. And also, to offer a guide for writing papers for the Studia UBB Informatica journal, which we have done. This guide may be found at [18].

We have noticed that authors tend to pay less attention on details they should take care of before sending the paper to the journal. We have identified at least four types of errors: linguistic errors, scientific content errors, paper style errors, LaTeX related errors.

We underline from the beginning that the rules from this guide are well known, and we cite their sources [1–22]. But we consider useful to have these rules in a single place, with our specific preferences, in a form we expect to be obeyed by authors publishing in the Studia UBB Informatica journal.

2. The analysis of some recent manuscripts

Although the instructions for authors can be found on the Studia UBB Informatica web page, they are often not obeyed. We have analyzed 44 recent manuscripts, and we have centralized the frequent errors in Table 1.

One may notice that linguistic aspects are on the first place, which may be explained by the fact the authors are not English people. The scientific errors are also present with a large share. Lack of clarity, ambiguity, not enough rigor, often make the manuscripts unintelligible. Also, the wrong use of citations/references dominates the second category of errors. Many people do not give sufficient details on the references (volume, issue number, pages).

Although linguistic aspects are present in almost all papers, we consider the scientific errors more dangerous, difficult to eliminate. They differ from

2010 *Mathematics Subject Classification.* 97P99.

1998 *CR Categories and Descriptors.* K.7.0 [**Computing Milieux**]: The Computing Profession – *General*.

author to author; depend on their knowledge, intelligence, perseverance, and experience. They are unavoidably present in the authors first papers, and usually disappear in time.

	Errors	no	freq
A	Linguistic errors	38	86%
	long or wrong sentences or phrases	8	
	missing or unsuitable words	25	
	wrong grammar	21	
	ambiguous or unclear statements	22	
B	Scientific content errors	35	80%
	ambiguous statements	4	
	undefined concepts	10	
	unclear of incomplete reasoning	12	
	incomplete or missing presentation of known results	10	
	missing comparison with other works	7	
	unclear scientific contribution	8	
	missing references	5	
	wrong conclusion	5	
	unfinished paper	7	
C	Style errors	34	77%
	incomplete or nonstandard references	17	
	nonstandard abstract (long, unclear, ...)	7	
	missing keywords	6	
	missing citations	10	
	unsuitable title (too long, wrong choice)	5	
	non-standard/unsuitable paper structure	5	
D	Other types	26	59%
	missing punctuation	17	
	typing errors	5	
	missing letters inside words	8	
	uncited references	3	
	lack of or incomplete authors address	4	
	wrong format	3	
	LaTeX style and typesetting errors	21	
	TOTAL	44	100%

TABLE 1. The list of errors found in manuscripts

3. The important types of rules that must be obeyed by an author

The structure of a paper should contain a title, an abstract, an introduction, the main part with the authors original contribution, a conclusion, references and some additional information. These sections must be present in all papers! [8, 18, 19].

The **Abstract** should present the basic results obtained by the author and presented in the paper. It must clearly and concisely present these results, in no more than 250 words. No other information about existing results must be writen in the abstract, and no references are cited here.

The **Introduction** must state clearly the subject covered by the author and its relevance to the major topic of science. Also, existing results in the field (the state of the art), and their relations to the authors results should be presented, either here or in a separate background (**Related Work**) section [20]. These existing results must be cited and corresponding papers must figure as references.

The **Contribution** part must present clearly, completely, with sufficient detail and rigurosity, what has been done by the author. If there are more than one important results, each one may be presented in separate sections. All hypotheses, experiments, deductions, and results, and their interpretation must be described. The statements must be clear, long sentences must be avoided, and ambiguities eliminated. Figures and tables may be used for increasing understanding.

The **Conclusions** section exactly state the results, and must agree with what has been done in the paper. It should also discuss on the importance of the results and, possibly, present plans for future research.

The **References** part should contain all papers that was used by or influenced the author in writing that paper. All of them must be cited inside the paper refered by their numbers in the references list. References in the list must be arranged / ordered lexicographically.

A reference should contain all information needed to discover that paper; the journal or proceedings where it was published, the volume, number of issue, and pages inside. For clarity, an example of a reference - paper in a journal is [8], of an on-line paper [13], of a paper in a conference proceedings is [16], and of a book or monograph is [21].

The official style of Studia UBB Informatica has to be obeyed by all authors.

There is no space to give them here, therefore details about these style rules may be seen in a guide found on the Studia UBB Informatica web page [18].

4. LaTeX style and typesetting errors

There is a diverse array of LaTeX errors our authors make on a general basis. First and foremost, the papers should be (directly) written in LaTeX, and not in another WISYWIG editor and converted afterwards. The use of such conversion features indicate the lack of LaTeX knowledge of authors and their choice to let the redaction be concerned with the aspect issues of the paper. Not only that this indicates lack of respect, this is as well an unacceptable behavior.

The Studia UBB Informatica editors have prepared a simple example file to be used by the authors. Not all of them use this model. We have received LaTeX papers using other style files or, even, using the standard LaTeX files. The example LaTeX file and the Studia style file are not there for our fun, but as a rule to be obeyed by authors.

A frequent LaTeX error has been the improper use of references and citations. Both should be done using LaTeX bibliographical features. Instead, quite a lot of papers use simple lists for references and square brackets for citations. In case of need please read a good LaTeX documentation, manual or book [5].

Very frequent LaTeX errors are related to text justification, fonts and sizes. The paper title is sometimes very long. The same stands with authors name. The authors should use LaTeX syntax to produce a short title and short names for papers heading, such that the headers do not overlap with the page numbers. Authors should make sure their text is correctly justified and that there are no words, equations, tables or figures left outside the text frame, on the right side of the page. As well, the official text size and fonts used by the journal are not to be replaced. We had a number of papers whose authors have changed the fonts to other fonts, more to their like. Again, this is an unacceptable behavior.

Authors should make all effort to integrate the figures with their captions in the LaTeX document, using the standard LaTeX commands. As well, all numberings should be generated automatically. The papers should have automatic numbering features for sections, figures, tables, equations, references. In case of need, as always, you are sent to your preferred LaTeX documentation, manual or book [5].

5. The need to improve our manuscripts

From those 44 analyzed papers and reviews, 35 were returned to authors for eliminating the errors and improvement. Some of them could be easily sent to the journal without those errors! With a minor requirement: attention of the authors.

It would be very useful the authors will read those 21 suggestions made by Lertzmann [13]. One of them would be suitable to some of our authors: "Do not think to publish the first draft".

Reread it yourself, and correct all misspellings and ambiguous expressions. Analyze each sentence to clarify its meaning. If there are long phrases rewrite them by shorter sentences. Since some of our authors are at their first papers, maybe PhD students, and almost all are Romanians, ask some other person to help reading your manuscript before submitting it to the editor. S/he may be your supervisor, or a research team-mate, or just a friend. It is important, however, that the person helping you with cleaning an English paper to be not only an excellent English speaker, but an excellent professional in the scientific domain of the paper. Otherwise, the translator will do more damage than help, because the final responsibility for your manuscript stays with you, and not with your translator. Finally, just before sending the paper to the editor, read it again. Many authors [9, 10] suggest this possibility and with a little care on behalf of the authors, many errors will be discovered and corrected before the paper reaching the reviewers.

Another frequent error which can be easily eliminated with a little attention from the authors consists in obeying the Studia UBB Informatica style, and giving all the required information. We are reminding here the keywords and phrases, scientific AMS and ACM classification indices [1, 2], complete authors official postal addresses (not personal addresses), official email addresses (not Yahoo, GMail and so on), incomplete references, or missing citations should be given.

To help our future authors to improve their writing and to submit better manuscripts to our journal we have written a guide for them [18].

References

[1] Association of Computing Machinery, *The ACM Computing Classification System*, http://www.acm.org/about/class/1998

[2] American Mathematical Society, *The AMS Mathematics Subject Classification*, http://www.ams.org/msc

[3] S. Amonson, *Style in Scientific Writing*, Essays of an Information scientist, 3 (1977-78), pp. 4–13.

[4] R. Andone, I. Dzitac, *How to write a good paper in Computer Science and how will it be Measured by ISI Web of Knowledge*, Int. J. of Computers, Communication, and Control, 5 (4), 2010, pp.432–446.

[5] P. A. Blaga, H. F. Pop, *LaTeX2e*, Editura Tehnică, Bucharest, 1999.

[6] G. Bochmann, *How to do Research Paper in Computer Science*, (contains Writing Technical Articles by H. Schulzrinne), 2007, http://www.elg.uottawa.ca/∼bnochmann/dsrg/how-to-do-good-research/how-to-write-papers

[7] B. Buchberger, *Thinking, Speaking, Writing*, Springer Verlag, 2010, http://www.risc.jku.at/education/courses/ws2010/tsw
[8] R. Day, *How to write a scientific paper*, IEEE Trans. On Professional Commmunication, ASM News, 41, 7 (1975), pp. 486–494.
[9] R. Day, *How to Write and Publish a Scientific Paper*, Oryx Press, 1998.
[10] M. Ernst, *Writing a technical paper*, 2010, http://www.cs.washington.edu/homes/mernst /advice/write-technical-paper-html
[11] M. Frenţiu, H. F. Pop, *A Guide for Writing Scientific Papers*, 2010, http://www.cs.ubbcluj.ro/∼studia-i/guideWritingPapers.pdf
[12] P. Lange, *How to write a scientific paper for a peer-reviewed journal*, chapter 5 in "How to write and illustrate a Scientific Paper", Cambridge Press, UK, 2007.
[13] K. Lertzman, *Twenty one suggestions for Writing Good Science Paper*, Bulletin of the Ecological Society of America, 1996, http://bio.wiona.edu/delong/EcolLab/21
[14] S. Maloy, *Guidelines for Writing a Scientific Paper*, 2001, http://www.sci.sdsu.edu/ smaloy/MicrobialGenetics/topics/scientific-writing.pdf
[15] S. Maloy, *How to write a scientific paper and Word Usage in Scientific Writing*, http://www.sciencediversity.com/2010/how-to-write-a-scientific-paper-and-word-usage-in-scientific -writing
[16] H. F. Pop, M. Frenţiu, *Effort Estimation by Analogy based on Soft Computing Methods*, KEPT2009: Knowledge Engineering: Principles and Techniques, Selected Papers, Cluj University Press, Cluj-Napoca, 2009, pp. 239–246.
[17] H. Schulzrinne, *Common Bugs in Writing*, 2009, http://www.columbia.edu/∼hgs/etc/writing-bugs.html
[18] Studia UBB Informatica, *Information for Authors*, 2010, http://www.cs.ubbcluj.ro//*sim*studia-i/authors.php
[19] M. E. Tischler, *Scientific Writing Booklet*, http://www.biochem.arizona.edu/mark/SciWriting.pdf
[20] M. Vlada, *Metodologia conceperii, elaborării şi redactării lucrărilor ştiinţifice*, Conferinţa Naţională de Învăţământ Virtual, ediţia a VII-a, 2009, pp. 47–53.
[21] Zobel, *Writing for Computer Science*, The Art of Effective Communication, Springer Verlag, 1997.
[22] * * *, Principles of Science Writing, Scitext Cambridge, UK, http://www.scitext.com/writing.php

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogălniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: mfrentiu@cs.ubbcluj.ro, hfpop@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

A REINFORCEMENT LEARNING APPROACH FOR SOLVING THE MATRIX BANDWIDTH MINIMIZATION PROBLEM

GABRIELA CZIBULA[1], ISTVAN-GERGELY CZIBULA[1] AND CAMELIA-MIHAELA PINTEA[2]

Abstract. In this paper we aim at investigating and experimentally evaluating the reinforcement learning based model that we have previously introduced to solve the well-known matrix bandwidth minimization problem (*MBMP*). The *MBMP* is an *NP*-complete problem, which is to permute rows and columns of a matrix in order to keep its nonzero elements in a band lying as close as possible to the main diagonal. The *MBMP* has been found to be relevant to a wide range of applications including circuit design, network survivability, data storage and information retrieval. The potential of the reinforcement learning model proposed for solving the *MBMP* was confirmed by the computational experiment, which has provided encouraging results.

1. Introduction

The *MBMP* is an *NP*-complete problem, which has been found to be relevant to a wide range of applications including circuit design, network survivability, data storage and information retrieval.

Given a square matrix $\mathcal{A} = (a_{ij})_{n\times n}$, the matrix bandwidth minimization problem consists in finding a permutation of the rows and columns of $\mathcal{A}$ that keeps the nonzero elements in a band lying as close as possible to the main diagonal of $\mathcal{A}$. The problem can be easily stated in terms of graph optimization problem, considering a vertex for each row and a vertex for each column and connecting vertex i to vertex j through an edge either if $a_{ij} \neq 0$ or $a_{ji} \neq 0$. This way, a graph $G_{\mathcal{A}}$ is associated to the problem of minimizing the bandwidth of matrix $\mathcal{A}$ and the original problem is equivalent to the problem

Received by the editors: November 5, 2010.

2000 *Mathematics Subject Classification.* 65K10, 68T05.

1998 *CR Categories and Descriptors.* I.2.6[**Computing Methodologies**]: Artificial Intelligence – *Learning*; I.2.8[**Computing Methodologies**]: Problem Solving, Control Methods, and Search – *Heuristic methods* .

Key words and phrases. Combinatorial optimization, Matrix Bandwidth Minimization Problem, Reinforcement Learning.

of finding a labeling f of the vertices that minimizes the maximum difference between labels of adjacent vertices in $G_{\mathcal{A}}$. The matrix bandwidth minimization problem has been shown to be NP-complete [10].

The bandwidth minimization problem is relevant to a wide range of optimization applications. In solving large linear systems, Gaussian elimination can be performed much faster on matrices with a reduced bandwidth. Bandwidth minimization has also found applications in circuit design and saving large hypertext media [13]. Other practical problems are found in data storage, VLSI design, network survivability,industrial electromagnetics [5], finite element methods for approximating solutions of partial differential equations, large-scale power transmission systems, circuit design, chemical kinetics and numerical geophysics [6, 7].

Reinforcement Learning (RL) [4] is an approach to machine intelligence in which an agent can learn to behave in a certain way by receiving punishments or rewards on its chosen actions.

We have previously introduced in [8] a theoretical reinforcement learning based model for solving the *MBMP* problem. In this paper we detail our previous approach, also providing an experimental evaluation of it. The obtained results are good enough, indicating the potential of using RL techniques for solving the *MBMP*.

To our knowledge, excepting our proposal, a RL model for solving the $MBMP$ problem hasn't been reported in the literature, so far.

The rest of the paper is organized as follows. Section 2 briefly presents existing approaches for solving the Matrix Bandwidth Minimization Problem. The reinforcement learning model that we propose for solving the *MBMP* is introduced in Section 3. Section 4 provides an experimental evaluation of the RL approach and Section 5 contains some conclusions of the paper and future development of our work.

2. Related Work

Because of the importance of the bandwidth minimization problem, much research has been carried out in developing algorithms for it.

Cuthill and McKee propose in 1969 in [1] the first, best-known, stable and simple heuristic method for *MBMP*. It is the well-known CM algorithm, which used Breadth-First Search to construct a level structure of the graph. By labeling the vertex in the graph according to a level structure, good bandwidth minimization results are achieved in a short time. CM starts from a vertex with minimum degree and constructs a list of vertices that is the new vertices ordering. At each step, all the vertices that are adjacent to those already in list are appended, in ascending degree order. As the graph is connected, the

procedure stops when the list has $|V|$ positions filled, V being the number of vertices. The popular software package MatLab uses the command SYMRCM to find a permutation of vertices with good bandwidth, based on a variation of the Cuthill and McKee method.

The *GPS* algorithm proposed Gibbs, Poole and Stockmeyer in 1976 [14], is also based on level structure. Computational results show that the *GPS* algorithm is comparable to the *CM* algorithm in solution quality while being several times faster.

Marti et al. have used in [6] Tabu Search for solving the the *MBMP* problem. They used a candidate list strategy to accelerate the selection of moves in the neighborhood of the current solution. Extensive experimentation show that their Tabu Search outperforms the best-known algorithms in terms of solution quality in reasonable time.

A *GRASP* with *Path Relinking* method given by Pinana et al. in [7] has been shown to achieve better results than the Tabu Search procedure but with longer running times.

Lim et al. propose in [13] a Genetic Algorithm integrated with Hill Climbing to solve the bandwidth minimization problem. Computational experiments show that the proposed approach achieves the best solution quality when compared with the *GPS* algorithm, *Tabu Search*, and the *GRASP* with *Path Relinking* methods, being faster than the latter two heuristics.

A simulated annealing algorithm is presented in [15] for the matrix bandwidth minimization problem. The algorithm proposed by Tello et al. is based on three distinguished features including an original internal representation of solutions, a highly discriminating evaluation function and an effective neighborhood.

More recently, the Ant Colony Optimization (ACO) metaheuristic has been used in [17, 9] in order to solve the *MBMP*. The ant colony system was also hybridized in [9] with two local procedures which improve the system's performance.

3. A Reinforcement Learning Model for Solving *MBMP*

In this section we introduce the reinforcement learning model proposal for solving the *MBMP* problem. The theoretical model was previously introduced in [8] and will be detailed in this section and experimentally evaluated in Section 4.

3.1. **Reinforcement learning.**

Reinforcement learning is a synonym of learning by interaction [19]. During learning, the adaptive system tries some actions (i.e., output values) on its environment, then, it is reinforced by receiving a

scalar evaluation (the reward) of its actions. The reinforcement learning algorithms selectively retain the outputs that maximize the received reward over time. Reinforcement learning tasks are generally treated in discrete time steps.

At each time step, t, the learning system receives some representation of the environment's state s, it takes an action a, and one step later it receives a scalar reward r, and finds itself in a new state s'. The two basic concepts behind reinforcement learning are trial and error search and delayed reward [4].

One key aspect of reinforcement learning is a trade-off between *exploitation* and *exploration* [20]. To accumulate a lot of reward, the learning system must prefer the best experienced actions, however, it has to try (to experience) new actions in order to discover better action selections for the future.

3.2. **Our RL model.** Let us consider, in the following, that $\mathcal{A}$ is the symmetric matrix of order n whose bandwidth has to be minimized. The assumption that $\mathcal{A}$ is symmetric does not reduce the generality of the model.

We extend the set of vertices $\mathcal{V}$ from the graph $G_{\mathcal{A}}$ associated with the *MBMP* problem with a vertex denoted by s_0 and connected to all other vertices, i.e $\mathcal{V} = \{1, 2, ..., n\} \bigcup s_0$.

A general RL task is characterized by four components: a state space $\mathcal{S}$ that specifies all possible configurations of the system; the action space $\mathcal{A}$ that lists all available actions for the learning agent to perform; the transition function that specifies the possibly stochastic outcomes of taking each action in any state; and a reward function that defines the possible reward of taking each of the actions.

The RL task associated to the *MBMP* is defined as follows:

- The state space $\mathcal{S}$ (the agent's environment) consists of the extended set of vertices $\mathcal{V}$, i.e. $\mathcal{S} = \mathcal{V}$. The initial state si of the agent in the environment is s_0. A state $sf \in \mathcal{S}$ reached by the agent at a given moment after it has visited states $si, s_1, s_2, ...s_k$ is a *terminal* (final) state if the number of states visited by the agent in the current sequence is $n + 1$, i.e. $k = n$.
- The action space $\mathcal{A}$ is implicitly defined by a transition function between the states from $\mathcal{S}$. The transition function between the states is defined as $h : \mathcal{S} \to \mathcal{P}(\mathcal{S})$, where $h(s) = \{1, 2, ..., n\} \setminus \{s\}$, $\forall s \in \mathcal{S}$. This means that, at a given moment, from the state s the agent can move in any state from $\mathcal{S}$, excepting state s. We say that a state s' that is accessible from state s, i.e $s' \in h(s)$, is the *neighbor* (*successor*) state of s.

- The transitions between the states are equiprobable, the transition probability $P(s, s')$ between a state s and each neighbor state s' of s is equal to $\frac{1}{n}$.
- The reward function will be defined below (Formula (1)).

The *MBMP* formulated as a RL problem consists in training the agent to find a path $si, \pi_1, \pi_2, ..., \pi_n$ from the initial to a final state, i.e a permutation π of $\mathcal{V}$ that minimizes the matrix bandwidth. We denote by $\mathcal{A}^\pi = (a_{ij}^\pi)_{n \cdot n}$ the matrix obtained from the initial matrix $\mathcal{A}$ by permuting its lines and columns in the order $\pi_1, \pi_2, ..., \pi_n$.

It is known that the estimated utility of a state [3] in a reinforcement learning process is the estimated *reward-to-go* of the state (the sum of rewards received from the given state to a final state). So, after a reinforcement learning process, the agent learns to execute those transitions that maximize the sum of rewards received on a path from the initial to a final state.

As we aim at obtaining a permutation π of $\mathcal{V} = \{1, 2, ..., n\}$ that minimizes the matrix bandwidth, we define the reinforcement function as follows (Formula (1)):

- the reward received after a transition to a non terminal state is 0;
- the reward received after a transition to a final state π_n after states $si, \pi_1, \pi_2, ...\pi_{n-1}$ were visited is minus the bandwidth of matrix $\mathcal{A}^\pi$.

$$(1) \qquad r(\pi_k | si, \pi_1, \pi_2, ...\pi_{k-1}) = \begin{cases} 0 & if \;\; k <> n \\ -\max_{a_{ij}^\pi \neq 0} |i - j| & otherwise \end{cases},$$

where by $r(\pi_k | \pi_1, \pi_2, ...\pi_{k-1})$ we denote the reward received by the agent in state π_k, after it has visited states $\pi_1, \pi_2, ...\pi_{k-1}$.

Considering the reward defined in Formula (1), as the learning goal is to maximize the total amount of rewards received on a path from the initial to a final state, it can be easily proved that the agent is trained to find a permutation π of $\mathcal{V} = \{1, 2, ..., n\}$ that minimizes the bandwidth of matrix $\mathcal{A}$.

During the training step of the learning process, the agent will determine its *optimal policy* in the environment, i.e the *policy* that maximizes the sum of the received rewards.

3.2.1. *The training process of the MBMP agent.* For training the *MBMP* agent, the $TD(\lambda)$ [4] algorithm is used. It is a temporal-difference (TD) method [18] combined with eligibility traces to obtain a more general and efficient learning method. λ refers to the use of an *eligibility trace* [21].

The *eligibility trace* is one of the basic mechanisms used in reinforcement learning to handle delayed reward. An eligibility trace is a record of the occurrence of an event such as the visiting of a state or the taking of an action

[4]. By associating one of such traces to every possible action in every state, the following temporal credit assignment is implemented: "Earlier states/actions are given less credit for the current TD error".

We have used the backward view of the $TD(\lambda)$ algorithm and *accumulating eligibility traces*. The basic idea is that on each step, the eligibility traces for all states decay, and the eligibility trace for the one state visited on the step is incremented by 1.

The idea of the training process is the following:

- The agent starts with some initial estimates of the states' utilities.
- During some training episodes, the agent will experiment (using the ϵ-Greedy action selection mechanism) some (possible optimal) paths from the initial to a final state, updating the states' utilities estimations according to the $TD(\lambda)$ algorithm.
- During the training process, the states' utilities estimations converge to their exact values, thus, at the end of the training process, the estimations will be in the vicinity of the exact values.

It is proven that $TD(\lambda)$ converges with probability 1 to an optimal policy and utility function as long as all state-action pairs are visited an infinite number of times and the policy converges in the limit to the Greedy policy [22].

After the training step of the agent has been completed, the solution learned by the agent is constructed by starting from the initial state and following the *Greedy* mechanism until a solution is reached. From a given state i, using the *Greedy* policy, the agent transitions to an unvisited neighbor j of i having the maximum utility value.

Consequently, the solution of $MBMP$ reported by the RL agent is a permutation π of $\{1, 2, ..., n\}$ such that $U(\pi_1) \geq U(\pi_2) \geq ... \geq U(\pi_n)$. By U we have denoted the utility function whose values were learned during the training step.

4. Computational Experiment

An experiment for evaluating the RL model proposed in Section 3 will be presented in the following. The proposed system was implemented in Java, using as benchmark *can_24* instance from Harwell-Boeing sparse matrix collection [11].

The parameter values for our implementation are as follows.

- The number of training episodes is set to 12000.
- The learning rate $\alpha \in [0, 1]$ is set to 0.01.
- The parameter indicating the use of eligibility traces $\lambda \in [0, 1]$ is set to 1.

- The discount factor $\gamma \in [0, 1]$ used for decreasing the eligibility traces is set to 0.9.
- Regarding the ϵ parameter used for the *epsilon*-Greedy action selection mechanism during the training step, the following strategy was used: we have started with $\epsilon = 0.85$ in order to favor exploration, then after the training progresses ϵ is decreased until it reaches a value near 0, 10^{-5}, which means that at the end of the training exploitation is favorized.

Figure 1 depicts the training process of the $MBMP$ agent, illustrating how, at the end of the training, the bandwidth of the learned solution becomes stable.

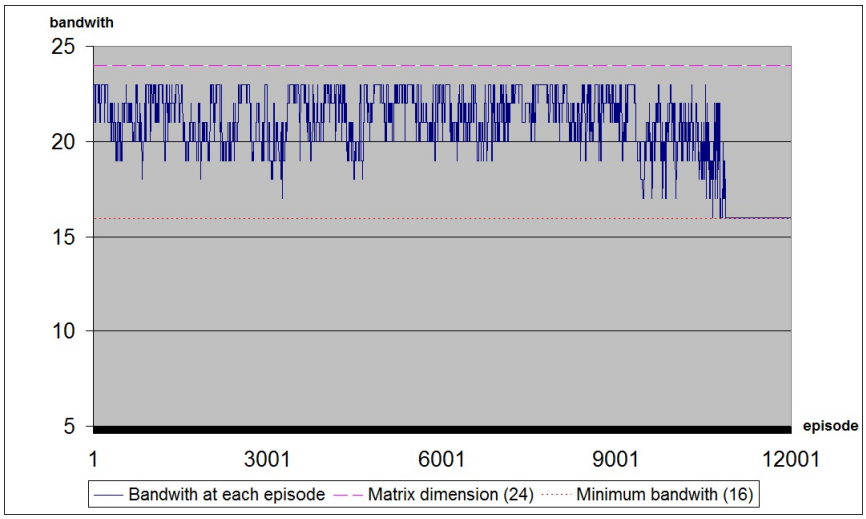


FIGURE 1. Results

We will compare in the following our approach with an approach that uses ant colony systems [16, 23]. The reason for this choice is that the ant systems can be related to reinforcement learning, as the pheromone by which the ants communicate can be viewed as a kind of reinforcement.

An ant colony system, ACS, was proposed in [9] for solving $MBMP$. The standard ACS was also hybridized with two local search mechanisms. For the benchmark *can__24*, ACS algorithm has reported a bandwidth 17. Compared with the standard ACS system, the solution reported our RL agent is better, but is worst than the solution reported by the hybrid ACS systems.

The results obtained by our approach are promising, but may be improved by further extensions of the proposed RL model.

5. Conclusions and Further Work

We have investigated in this paper a reinforcement learning based model for solving the matrix bandwidth minimization problem. The performed computational experiment has provided encouraging results, indicating the potential of using reinforcement learning techniques for the solving *MBMP*.

Further work will be made in order to improve the proposed RL model by investigating different reinforcement functions and by adding different local search mechanisms in order to increase the performance. We also aim at extending the proposed model to a distributed RL approach and at further evaluating it.

ACKNOWLEDGEMENT

This work was supported by CNCSIS - UEFISCSU, project number PNII - IDEI 2286/2008.

References

[1] Cuthill, E., McKee, J., *Reducing the bandwidth of sparse symmetric matrices*, Proceedings of the 1969 24th national conference, ACM Press, New York, NY, USA, 1969, pp. 157–172.

[2] Mitchell, T. M.: Machine Learning. New York: McGraw-Hill, 1997.

[3] Russell, S.J., Norvig, P., *Artificial intelligence. A modern approach*, Prentice-Hall International, 1995.

[4] Sutton, R., Barto, A., *Reinforcement Learning*, The MIT Press, Cambridge, London, 1998.

[5] Esposito, A., Catalano, M., S., Malucelli F., Tarricone, L., *Sparse Matrix Bandwidth Reduction: Algorithms, applications and real industrial cases in electromagnetics*, Advances in the theory of Computation and Computational Mathematics, Vol. 2, 1998, pp. 27–45.

[6] Marti, R., Laguna, M., Glover, F. and Campos, V., *Reducing the Bandwidth of a Sparse Matrix with Tabu Search*, European Journal of Operational Research, Vol. 135, No. 2, 2001, pp. 211–220.

[7] Pinana, E., Plana, I., Campos, V. Marti, R., *GRASP and Path Relinking for the Matrix Bandwidth Minimization*, European Journal of Operational Research, Vol. 153, Issue 1, 2004, pp. 200–210.

[8] Czibula, G., Crisan, G.C., Pintea, C.M., Czibula, I.G., Soft computing approaches on the Bandwith Problem, Proceedings of International Conference on Applied Mathematics (ICAM7), 2010, to be published.

[9] Pintea, C-M.,Crisan G-C., Chira C., *A Hybrid ACO Approach to the Matrix Bandwidth Minimization Problem*, HAIS **1** 2010, Springer LNCS (LNAI) 6076, 2010, pp. 405–412.

[10] Papadimitriou, C.H., *The NP-completeness of the bandwidth minimization problem*, Computing 16, 3, 1976, pp. 263-270.

[11] National Institute of Standards and Technology, Matrix Market, *Harwell-Boeing sparse matrix collection*, http://math.nist.gov/MatrixMarket/data/Harwell-Boeing/.

[12] Berry, M., Hendrickson, B., Raghavan, P., *Sparse matrix reordering schemes for browsing hypertext*, Lectures in Appl. Math. 32: The Mathematics of Numberical Analysis, 1996, pp. 99–123.

[13] Lim, A., Rodrigues, B., and Xiao, F., *Integrated genetic algorithm with hill climbing for bandwidth minimization problem*, Proceedings of the 2003 international Conference on Genetic and Evolutionary Computation, Lecture Notes In Computer Science., Springer-Verlag, Berlin, Heidelberg, 2003, pp. 1594–1595.

[14] Gibbs, N.E., Poole, W.G., Stockmeyer, P.K., *An algorithm for reducing the bandwidth and profile of sparse matrix*, SIAM Journal on Numerical Analysis **13(2)**, 1976, pp. 236–250.

[15] Rodriguez-Tello, E., Jin-Kao, H., Torres-Jimenez, J., *An improved Simulated Annealing Algorithm for the matrix bandwidth minimization*, European J. of Oper. Res., **185(3)**, 2008, pp. 1319–1335.

[16] Dorigo, M., Gambardella, L.M., *Ant Colony System: a cooperative learning approach to the traveling salesman problem*, IEEE Trans. on Evolutionary Computation, **1(1)**, 1997, pp. 53–66.

[17] Lim, A., Lin, J., Rodrigues, B., Xiao, F., *Ant Colony Optimization with hill climbing for the bandwidth minimization problem*, Appl. Soft Comput. , **6(2)**, 2006, pp. 180–188.

[18] Sutton, R.S., *Learning to predict by the methods of temporal differences*, Machine Learning **3**,1998, pp. 9–44.

[19] Perez-Uribe, A., *Introduction to Reinforcement learning*, 1998, http://lslwww.epfl.ch/∼anperez/RL/RL.html.

[20] Thrun, S.B., *The role of exploration in learning control*, *Handbook of Intelligent Control: Neural, Fuzzy, and Adaptive Approaches*, Van Nostrand Reinhold, New York, NY, 1992.

[21] Singh, S.P., Sutton, R.S., *Reinforcement learning with replacing eligibility traces*, Machine Learning **22**, 1996, pp. 123–158.

[22] Dayan P., Sejnowski, T.J., *TD(lambda) Converges with Probability 1*, Machine Learning, Volume 14, Number 1, 1994, pp. 295–301.

[23] Lupea, M., *Default reasoning by ant clony optimization*, Studia Univ. Babes-Bolyai, Informatica, **LIV**, Number 2, 2009, pp. 71–82.

[1]Babeş-Bolyai University, 400084 Cluj-Napoca, Romania; [2];George Coşbuc N. College, 400083 Cluj-Napoca, Romania;

E-mail address: gabis@cs.ubbcluj.ro, istvanc@cs.ubbcluj.ro, cmpintea@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

EVOLUTIONARY MULTIOBJECTIVE APPROACH FOR MULTILEVEL COMPONENT COMPOSITION

A. VESCAN AND C. GROŞAN

Abstract. Component-based Software Engineering (CBSE) uses components to construct systems, being a means to increase productivity by promoting software reuse and increasing software quality. The process of assembling component is called component composition. Components are themselves compositions of components. This give rise to the idea of composition levels, where a component on level i may be decomposed (using more components) at level $i+1$ or compositions at level $i+1$ serves as component at level i.

We are approaching the multilevel component composition problem. We formulate the problem as multiobjective, involving 4 objectives. The approach used is an evolutionary computation technique.

1. Introduction

Component-Based Software Engineering (CBSE) is concerned with designing, selecting and composing components [1]. As the popularity of this approach and hence number of commercially available software components grows, selecting a set of components to satisfy a set of requirements while minimizing a set of various objectives (as cost, number of used components) is becoming more difficult.

In this paper, we address the problem of automatic component selection. Informally, our problem is to select a set of components from available component set which can satisfy a given set of requirements while minimizing the number of used components. To achieve this goal, we should assign each component a set of requirements it satisfies.

In general, there may be different alternative components that can be selected, each coming at their own set of offered requirements. We aim at a selection approach that guarantees the optimality of the generated component

Received by the editors: October 22, 2010.

2010 *Mathematics Subject Classification.* 68T01, 68N19.

1998 *CR Categories and Descriptors.* D.2.13 [**Reusable Software**]: – *Reusable Software*; I.2.2 [**Artificial Intelligence**]: – *Automatic Programming*.

Key words and phrases. component, assembly, multilevel, automatic construction.

system. The compatibility of components is not discussed here, it will be treated in a future development.

Component selection methods are traditionally done in an architecture-centric manner. An approach was proposed in [6]. The authors present a method for simultaneously defining software architecture and selecting off-the-shelf components. Another type of component selection approaches is built around the relationship between requirements and components available for use. In [2] the authors have presented a framework for the construction of optimal component systems based on term rewriting strategies. Paper [4] proposes a comparison between a Greedy algorithm and a Genetic Algorithm. Various genetic algorithms representations were proposed in [9, 10, 8, 7].

All the above approaches did not considered the multi-level structure of a component-based system. They all constructed the final solution as a one level system, but components are themselves compositions of components. This give rise to the idea of composition levels, where a component on level i may be decomposed (using more components) at level $i + 1$ or compositions at level $i + 1$ serves as component at level i.

The motivation to propose this approach is two fold. Firstly, this paper presents a systematic approach to describe component-based systems having a multilevel structure. This offers a means to abstract details not needed in a certain level.

Secondly, the proposed evolutionary multiobjective approach provides a way of finding the "best" solution out of a set of solutions.

The paper is organized as follows: Section 2 presents a short introduction on components and their compositions. The proposed approach (that uses an evolutionary algorithm) is presented in Section 3. In Section 4 some experiments and comparisons are performed. We conclude our paper and discuss future work in Section 5.

2. Components and multi-level Compositions

A component is an independent software package that provides functionality via welldefined interfaces. The interface may be an export interface through which a component provides functionality to other components or an import interface through which a component gains services from other components. The purpose [1] of a component is to provide functionality that can be used in different contexts. This functionality is accessible through a components provides interface. Components may have multiple provide interfaces.

A component can depend on functionality offered by other components. The functionality that is required by a component forms a components requires interface. A component may also have multiple of these. A component

with a requires interface can be bound to any component that implements this interfaces. Thus, by specifying functionality in terms of interfaces, no dependencies on concrete components are introduced. This property makes components independently deployable. However, non-functional properties of components, such as performance characteristics, may yield such component dependencies. These are ignored in this article.

The "blackbox" nature of a component is important: that is, a component can be incorporated in a software system without regard to how it is implemented. In other words, the interface of a component should provide all the information that users need. Moreover, this information should be the only information they need. Consequently, the interface of a component should be the only point of access to the component.

Components are used as building blocks to form larger software entities. Assembling components [1] is called composition. Composition involves putting components together and connecting provided functionality to required functionality. Composition can be static or dynamic. With static composition the collection of components that form an application is statically known. With dynamic composition the composition of components is determined dynamically, e.g., at run-time. In this article we only consider static composition.

A graphical representation of our view of components is given in Figure 1. See details about component specification elements in [11]. Because of lack of space we only give a wordly description of component specification.

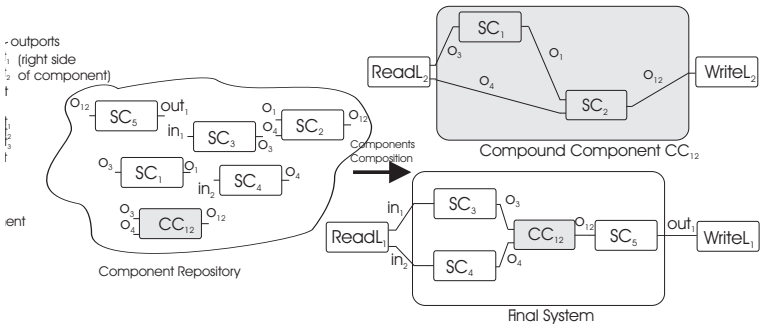


FIGURE 1. Components graphical representation and components assembly construction reasoning

There are two type of components: *simple component* - is specified by the inports (the set of input variables/parameters), outports (the set of output variables/parameters) and a function (the computation function of the component) and *compound component* - is a group of connected components in

which the output of a component is used as input by another component from the group.

In Figure 1 we have designed the compound component by fill in the box. We have also presented the inner side of the compound component: the constituents components and the interactions between them.

Two particular components are the source and the destination components: *the source component* has no inports and generates data provided as outports in order to be processed by other components and *the destination component* has no outports and receives data from the system as its inports and usually displays it, but it does not produce any output. The source component represents the "read" component and the destination component represents the "write" component.

Components are themselves compositions of components. This give rise to the idea of composition levels. In other words, in an hierarchical system, a subsystem of higher level components can be the infrastructure of a single component at a lower level. For example, in Figure 2 a higher level (component) is nested within the lower level. Any element at any level is both (if it is a compound component, gray fill color in the figure) a component in its own level and a subsystem at its adjacent higher (next) level. The first level represents the level of the final required system. Every compound component is decomposed into a subsystem that will be part of the next higher level.

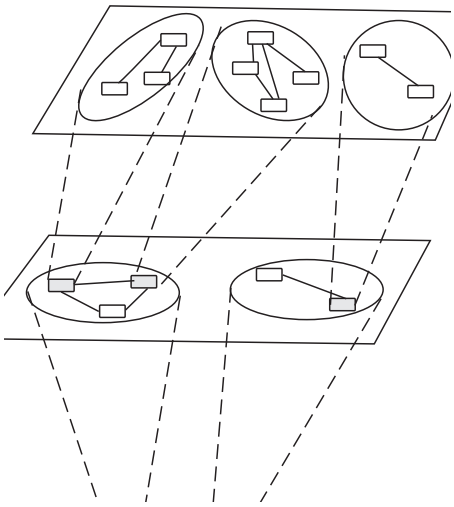


FIGURE 2. Hierarchical component-based system

2.1. **Problem formulation.** An informal specification of our aim is described next. It is needed to construct a final system specified by input data (that is given) and output data (what is required to compute). We can see the final system as a compound component and thus the input data becomes the required interfaces of the component and the output data becomes the provided interfaces, and in this context we have the required interfaces as provided and we need to provide the internal structure of the final compound component by offering the provided interfaces.

In Figure 3 all the above discussion is graphically represented.

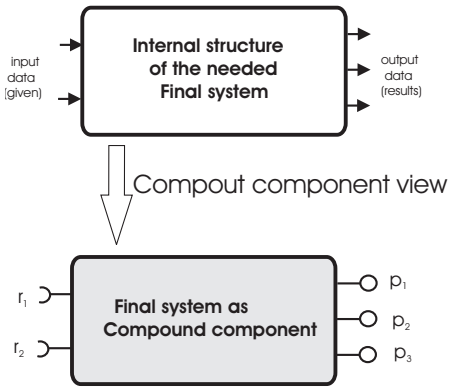


FIGURE 3. Graphycally representation of the problem formulation

A formal definition of the problem (seen as a compound component) is as follows. Consider SR the set of final system requirements (the provided functionalities of the final compound component) as $SR = \{r_1, r_2, ..., r_n\}$ and SC the set of components (repository) available for selection as $SC = \{c_1, c_2, ..., c_m\}$.

Each component c_i can satisfy a subset of the requirements from SR (the provided functionalities) denoted $SP_{c_i} = \{p_{i_1}, p_{i_2}, ..., p_{i_k}\}$ and has a set of requirements denoted $SR_{c_i} = \{r_{i_1}, r_{i_2}, ..., r_{i_h}\}$.

The goal is to find a set of components Sol in such a way that every requirement r_j $(j = \overline{1,n})$ from the set SR can be assigned a component c_i from Sol where r_j is in SP_{c_i} $(i = \overline{1,m})$, while minimizing the number of used components. All the requirements of the selected components must be satisfied by the components in the solution. If a selected component is a compound component, the internal structure is also provided. All the levels of the system are constructed.

3. Proposed approach description

Evolutionary algorithms are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. Inspired by Darwin's theory of evolution - Genetic Algorithms (GAs) are computer programs which create an environment where populations of data can compete and only the fittest survive, sort of evolution on a computer. They are well known suitable approaches for optimization problems.

The approach presented in this paper uses principles of evolutionary computation and multiobjective optimization [3]. First, the problem is formulated as a multiple objective optimization problem having 4 objectives: the number of used components, the number of new requirements, the number of provided interfaces and the number of the initial requirements that are not in solution. All objectives are to be minimized.

There are several ways to deal with a multiobjective optimization problem. In this paper the weighted sum method [5] is used. Let us consider we have the objective functions f_1, f_2,..., f_n. This method takes each objective function and multiplies it by a fraction of one, the "weighting coefficient" which is represented by w_i. The modified functions are then added together to obtain a single cost function, which can easily be solved using any method which can be applied for single objective optimization.

Mathematically, the new function is written as:

$$\sum_{i=1}^{n} w_i f_i, \ where\ 0 \leq w_i \leq 1\ and\ \sum_{i=1}^{n} w_i = 1.$$

In our case we have four objectives. Furthermore, the new function obtained by aggregating the four objectives can be written as:

$F(x) = \alpha \cdot f_1(x) + \alpha \cdot f_2(x) + \alpha \cdot f_3(x) + \alpha \cdot f_4(x)$.

The objectives (to minimize) are:

- the number of used components;
- the number of provided interfaces;
- the number of new added requirements;
- the number of initial requirements that are not in the solution.

3.1. Solution representation.

A solution (chromosome) is represented as a 4-tuple ($lstProv$, $lstComp$, $lstInitReq$, $lstNewReq$) with the following information:

- list of provided interfaces ($lstProv$);
- list of components ($lstComp$);
- list of initial requirements ($lstInitReq$);
- list of new requirements ($lstNewReq$).

The value of $i-th$ component represents the component satisfying the $i-th$ provided interface from the list of provided interfaces. A chromosome is initialized with the list of provided interfaces by the list of requirements of the final required system and with the list of initial requirements with the list of the requirements of the final required system (these will be provided as implicit, being input data of the problem/system). An example is given in what follows.

A valid chromosome using the components repository in Section 4 may be structured as follows: $Crom_0 = ((1,\ 2,\ 3),\ (9,\ 11,\ 11),\ (1,\ 2),\ (9,\ 10))$. This chromosome does not represent a solution, it is only an initialized chromosome without any applied genetic operator. The provided interfaces $(1,\ 2,\ 3)$ are offered by the components $(9,\ 11,\ 11)$. The set of initial requirements are: $(1,\ 2)$. By using a component we need to provide it's requirements: component 9 requires the 9-th new requirement and component 11 requires the 10-th new requirement.

The same chromosome after applying mutation operator has the internal structure:
$Crom_1 = ((1,\ 2,\ 3,\ 10, 9),\ (9,\ 11,\ 11,\ 12,\ 10),\ (1,\ 2),\ ())$. In order to provide the 10-th new requirement we have selected component 12, and for the 9-th new requirements the component 10 was chosen. No new requirements are added (the requirements of the new selected components are in the set of the initial requirements. A graphical visualization of the chromosome is given in Figure 4.

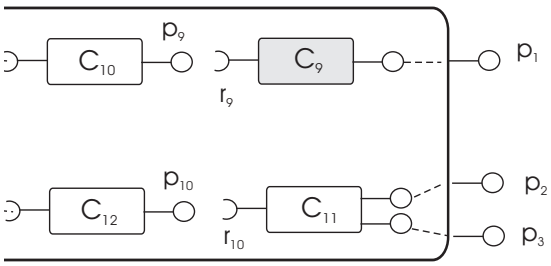


FIGURE 4. Chromosome representation $Crom_1 = ((1,\ 2,\ 3,\ 10, 9),\ (9,\ 11,\ 11,\ 12,\ 10),\ (1,\ 2),\ ())$

3.2. **Genetic operator: mutation.** The genetic operator used is mutation. Mutation operator used here consists in applying the following steps:

- randomly select a requirement form the list of new requirements;
- add the associated provided interface of the new requirement in the list of provided interfaces;

- add the component that satisfies the added provided interface (a component is randomly selected from the set of components that offer it);
- remove the required selected from the list of new requirements;
- add to the list of new requirements the requirements of the added component (if there are) to the list of components.

For instance, for the chromosome $Crom_{init} = ((1,\ 2),\ (21,\ 22),\ (4),\ (14,\ 15,\ 16))$ we can apply three mutations as follow:

(1) Mutation 1
- the selected new requirement is 16;
- add the associated provided interface:
  $Crom_{init} = ((1,\ 2,\ 16),\ (21,\ 22),\ (4),\ (14,\ 15,\ 16))$;
- the selected component to provide the 16-th requirements is component 23
  $Crom_{init} = ((1,\ 2,\ 16),\ (21,\ 22,\ 23),\ (4),\ (14,\ 15,\ 16))$;
- remove the satisfied requirement
  $Crom_{init} = ((1,\ 2,\ 16),\ (21,\ 22,\ 23),\ (4),\ (14,\ 15))$;
- add the requirements of the selected component - no new requirements.

(2) Mutation 2
- the selected new requirement is 15;
- add the associated provided interface:
  $Crom_{init} = ((1,\ 2,\ 16,\ 15),\ (21,\ 22,\ 23),\ (4),\ (14,\ 15))$;
- the selected component to provide the 15-th requirements is component 24
  $Crom_{init} = ((1,\ 2,\ 16,\ 15),\ (21,\ 22,\ 23,\ 24),\ (4),\ (14,\ 15))$;
- remove the satisfied requirement
  $Crom_{init} = ((1,\ 2,\ 16,\ 15),\ (21,\ 22,\ 23,\ 24),\ (4),\ (14))$;
- add the requirements of the selected component - no new requirements.

(3) Mutation 3
- the selected new requirement is 14;
- add the associated provided interface:
  $Crom_{init} = ((1,\ 2,\ 16,\ 15,\ 14),\ (21,\ 22,\ 23,\ 24),\ (4),\ (14))$;
- the selected component to provide the 14-th requirements is component 24
  $Crom_{init} = ((1,\ 2,\ 16,\ 15,\ 14),\ (21,\ 22,\ 23,\ 24,\ 24),\ (4),\ (14))$;
- remove the satisfied requirement
  $Crom_{init} = ((1,\ 2,\ 16),\ (21,\ 22,\ 23,\ 24,\ 24),\ (4),\ ())$;
- add the requirements of the selected component - no new requirements.

3.3. **Algorithm description.** In a steady-state evolutionary algorithm one member of the population is changed at a time. The best chromosome (or a few best chromosomes) is copied to the population in the next generation. Elitism can very rapidly increase performance of GA, because it prevents losing the best found solution to date. A variation is to eliminate an equal number of the worst solutions, i.e. for each "best chromosome" carried over a "worst chromosome" is deleted.

4. Experiments

A short and representative example is presented in this section. Starting for a set of three requirements and having a set of 29 available components, the goal is to find a subset of the given components such that all the requirements are satisfied.

The set of requirements $SR = \{r_1, r_2, r_3\}$ (view as provided interfaces $\{p_1, p_2, p_3$, see the discussion in Section 2.1) and the set of components $SC = \{c_0, c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8, c_9, ..., c_{29}\}$ are given as in Figure 5.

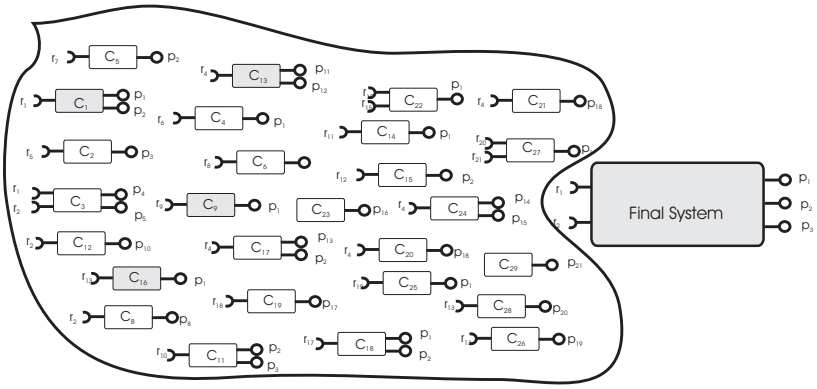


Figure 5. Components repository and final system

Figure 5 contains the components from the repository and the final system specification represented as a component (requirements of the (final) component are the input data of the problem and provided interfaces of the (final) component are the requirements of the problem, what should be provided by the final system). The compound components are depicted with fill grey color. There are many components that may provide the same functionality with different requirements interfaces.

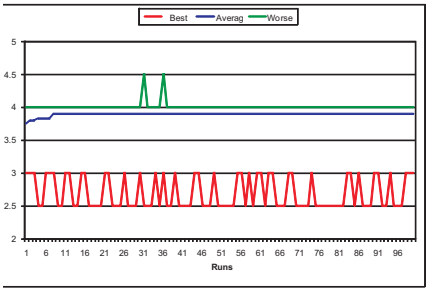


FIGURE 6. The evolution for independent runs of fitness function (best, worse and average)

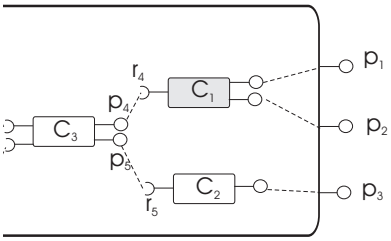


FIGURE 7. First level evolution and solution representation.

4.1. **Obtained results.** The parameters used by the evolutionary approach are as follows: population size 20, number of iterations 10, mutation probability 0.7. The value of α used while aggregating the objectives was set to 0.25 which gives the same importance to all objectives. The algorithm founded three levels, including the final system level (first level). For each level only one compound component was included.

The algorithm was run 100 times and the best, worse and average fitness values were recorded. The evolution of the fitness function for all 100 runs using the value $\alpha = 0.25$ is depicted in Figure 6 (first level), 8 (second level) and 10 (third level). Best, worse and average fitness value recorder for each run are presented. For each level the solution is also shown the representation in Figure 7.

A best solution (for the first level) was also found starting from the valid chromosome:
$Crom_{level_1} = ((1,\ 2,\ 3),\ (1,\ 1,\ 2),\ (1,\ 2),\ (4,\ 5))$. This chromosome does not represent a solution, it is only an initialized chromosome without any applied genetic operator. The provided interfaces $(1,\ 2,\ 3)$ are offered by the components $(1,\ 1,\ 2)$. The set of initial requirements are: $(1,\ 2)$. By using a component we need to provide it's requirements: component 1 requires the 4-th new requirement and component 2 requires the 5-th new requirement.

The same chromosome after applying mutation operator has the internal structure:
$Crom_{level_{1'}} = ((1,\ 2,\ 3,\ 4,\ 5),\ (1,\ 1,\ 2,\ 3,\ 3),\ (1,\ 2),\ ())$. In order to provide the 4-th new requirement we have selected component 3, and for the 5-th new requirements the same component was chosen. No new requirements are

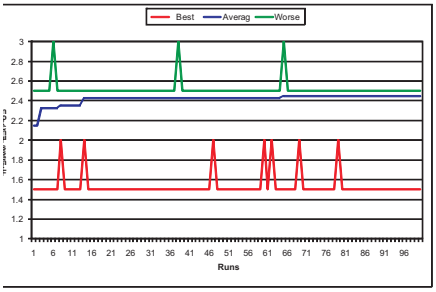


FIGURE 8. The evolution for independent runs of fitness function (best, worse and average)

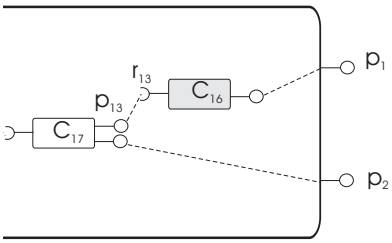


FIGURE 9. Level 2 solution representation of the chromosome with the best fitness value - $Crom_{level_2}$

added (the requirements of the new selected components are in the set of the initial requirements.

For the second level, the compound component C_1 was "constructed" by using other components. Figure 9 shows the internal structure of the found solution.

A best solution was also found starting from the valid chromosome: $Crom_{level_2} = ((1,\ 2),\ (16,\ 17),\ (4),\ (13))$. This chromosome does not represent a solution, it is only an initialized chromosome without any applied genetic operator. The provided interfaces $(1,\ 2)$ are offered by the components $(16,\ 17)$. The set of initial requirements contains only (4). By using a component we need to provide it's requirements: component 16 requires the 13-th new requirement.

The same chromosome after applying mutation operator has the internal structure: $Crom_{level_{2'}} = ((1,\ 2,\ 13),\ (16,\ 17,\ 17),\ (4),\ ())$. In order to provide the 13-th new requirement we have selected component 17. No new requirements are added (the requirements of the new selected components are in the set of the initial requirements.

For the third level, the compound component C_1 was "constructed" by using other components. Figure 9 shows the internal structure of the found solution.

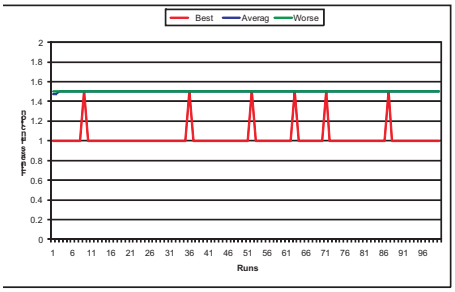


FIGURE 10. The evolution for independent runs of fitness function (best, worse and average)

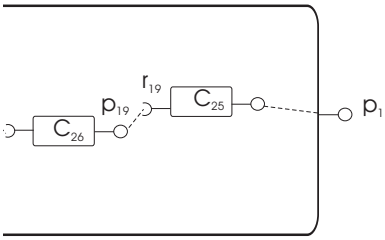


FIGURE 11. Level 3 solution representation of the chromosome with the best fitness value

A best solution was also found starting from the valid chromosome: $Crom_{level_3} = ((1), (25), (13), (19))$. This chromosome does not represent a solution, it is only an initialized chromosome without any applied genetic operator. The provided interfaces (1) are offered by the components (25). The set of initial requirements contains only (13). By using a component we need to provide it's requirements: component 25 requires the 19-th new requirement.

The same chromosome after applying mutation operator has the internal structure:
$Crom_{level_{3'}} = ((1), (25, 26), (13), ())$. In order to provide the 19-th new requirement we have selected component 26. No new requirements are added (the requirements of the new selected components are in the set of the initial requirements.

A comparison between the best fitness values using the first solution (Figure 7) and the second solution (Figure 12) is done.

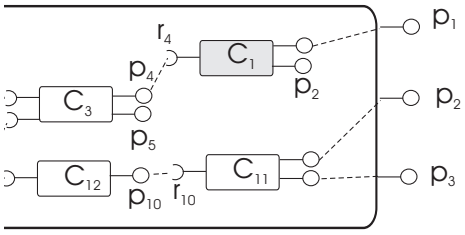


FIGURE 12. Level 1 solution representation of the chromosome with the best fitness value - second solution

The comparison is done using standard deviation and is presented in Figure 13. The same comparison between the worst fitness values are presented in Figure 14. Because the best obtained values using the two solutions are overlapping we could say that the same best solution may be obtained starting from the same components repository.

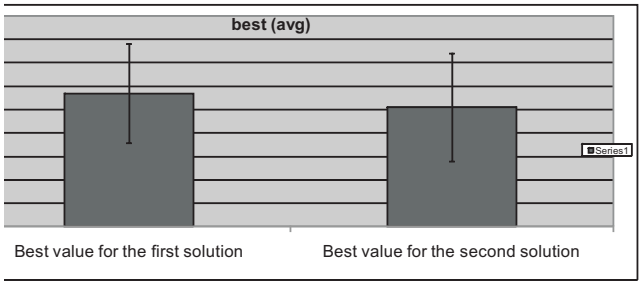


FIGURE 13. Boxplot giving the distribution of solution values achieved for 100 runs for the first (see Figure 7) solution for the level 1

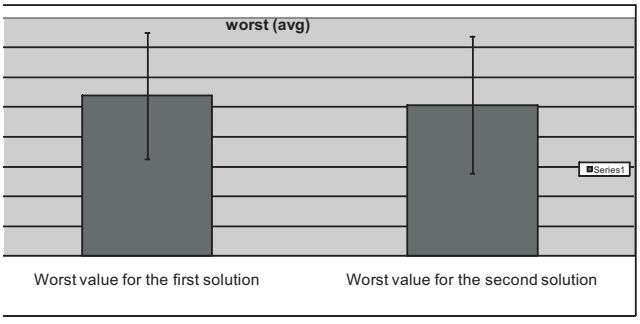


FIGURE 14. Boxplot giving the distribution of solution values achieved for 100 runs for the second (see Figure 12) solution for the level 1

4.2. **Discussion.** As evidents from the above results, the evolutionary algorithms are a suitable approach for CBSE problems. They are robust and fast, and, from the way the chromosome is constructed, the convergence to a solution is always guaranteed.

Another advantage of using evolutionary computation consist in the possibility of providing more than one solution, but a set of alternative feasible solutions.

By treating the problem as multiobjective we can incorporate all the problem constraints and analyze all the situations which can occur. The approach always converges to at least one solution of the system.

5. Conclusions

CBSE is the emerging discipline of the development of systems incorporating components. A challenge is how to assemble components effectively and efficiently. Multi-level component composition has been investigated in this paper. We have proposed a multiobjective evolutionary approach, 4 objectives being involved. For each level (a compound component) of the final system we have applied the algorithm and the solution corresponding to that level was obtained. For a level we have compared the best fitness values using two solution and because they are overlapping we could conclude that the same best solution may be obtained starting from the same components repository.

Some of the advantages of using an evolutionary algorithm are as follows:

- it obtain multiple solutions in a single run;
- it is fast and has a low computational complexity;
- it can be scaled to any number of components and requirements.

As future work we will consider dynamic modifications of the requirements of the final system, investigating: the use of the same representation as in the current paper, several ways to deal with the multiobjective optimization problem (the weighted sum method or Pareto principle), different ways of modifying the requirements, by adding new requirements or deleting existing ones.

Acknowledgement

This material is partially supported by the Romanian National University Research Council under award PN-II no. ID 2412/2009.

References

[1] I. Crnkovic, M. Larsson, *Building Reliable Component-Based Software Systems*, Artech House publisher, 2002.

[2] L. Gesellensetter, S. Glesner, *Only the Best Can Make It: Optimal Component Selection*, Electron. Notes Theor. Comput. Sci., vol. 176 (2), pp. 105–124, 2007.

[3] C. Grosan *A comparison of several evolutionary models and representations for multiobjective optimization*, ISE Book Series on Real Word Multi-Objective System Engineering, chapter 3, Nova Science, 2005.

[4] N. Haghpanah, S. Moaven, J. Habibi, M. Kargar, S. H. Yeganeh, *Approximation Algorithms for Software Component Selection Problem*, APSEC conference, pp. 159–166, 2007.

[5] Y. Kim, O.L. deWeck, *Adaptive weighted-sum method for bi-objective optimization: Pareto front generation*, in Structural and Multidisciplinary Optimization, vol 29 (2), pp. 149–158, 2005.

[6] E. Mancebo, A. Andrews, *A strategy for selecting multiple components*, SAC '05: Proceedings of the 2005 ACM symposium on Applied computing, pp. 1505–1510, 2005.

[7] A. Vescan, *Optimal component selection using a multiobjective evolutionary algorithm*, Neural Network World Journal, no. 2, pp. 201–213, 2009.

[8] A. Vescan, *A Metrics-based Evolutionary Approach for the Component Selection Problem*, the 11th International Conference on Computer Modelling and Simulation (UKSim 2009), pp. 83–88, 2009.

[9] A. Vescan, C. Grosan, *Two Evolutionary Multiobjective Approaches for the Component Selection Problem*, Proceedings of the Fourth International Workshop on Evolutionary Multiobjective Optimization Design and Applications, Kaohsiung, Taiwan, pp. 395–400, 2008.

[10] A. Vescan, C. Grosan, *A Hybrid Evolutionary Multiobjective Approach for the Component Selection Problem*, Proceedings of the 3rd International Workshop on Hybrid Artificial Intelligence Systems, Burgos, Spain, LNCS/LNAI 5271, pp. 164–171, 2008.

[11] A. Vescan, *Components ordered assembly construction based on temporal restraint*, Proceedings of the 3rd DoctoralWorkshop on Mathematical and Engineering Methods in Computer Science, Znojmo, Czechia, pp. 249–256, 2007.

Department of Computer Science, Faculty of Mathematics and Computer Science,, Babeş-Bolyai University, Cluj-Napoca, Romania

E-mail address: {avescan,cgrosan}@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

COMPLEX SYSTEMS AND CELLULAR AUTOMATA MODELS IN THE STUDY OF COMPLEXITY

CAMELIA CHIRA[(1)], ANCA GOG[(1)], RODICA IOANA LUNG[(2)], DAVID ICLANZAN[(1)]

Abstract. Complex systems consist of a large number of interconnected and mutually interacting components. The real world is full of examples of complex adaptive systems from ancient and modern cultures, biological and social systems to economies and ecosystems. The study of complex systems is crucial for a constructive assessment and understanding of essential aspects of everyday life. Computational approaches to the analysis of complexity are among the most important tools used for this purpose. The current paper aims to offer an overview of complex systems and their main properties of emergence, adaptability and self-organization. Cellular automata are reviewed as major computational techniques engaged for complex systems modelling. In a cellular automaton, simple rules give rise to complex emergent behaviours worth investigated in the context of many real-world complex models. Perspectives of promising research directions to be tackled are discussed.

1. Introduction

Complexity, emergence and self-organization represent essential aspects of today's world real systems. The study and in-depth analysis of these elements needs computational perspectives able to significantly impact the study of complexity and the solving process of dynamic complex problems. Complex systems research has developed and grown tremendously during the past two decades, being subject of studies in a great variety of fields including physics, biology, computer science, sociology and economics. There are three interrelated approaches to the modern study of complex systems: (i) study of the interactions that give rise to complex systems; (ii) models of complex systems; (iii) the process of formation of complex systems.

Received by the editors: November 5th, 2010.

2010 *Mathematics Subject Classification.* 68-02.

1998 *CR Categories and Descriptors.* A.1 General Literature [**INTRODUCTORY AND SURVEY**]; I.2.8 Computing Methodologies [**ARTIFICIAL INTELLIGENCE**]:Problem Solving, Control Methods, and Search – *Heuristic methods*.

Key words and phrases. complex systems, emergence, self-organization, cellular automata.

A complex system usually involves a large number of components. These components may be simple both in terms of their internal characteristics and in the way they interact. However, when the system is observed over a long time period and length scales, there may be phenomena that are not easily understood in terms of its simple components and their interactions. Complex systems are mainly characterized by emergence, complexity, self-organization, non-linearity, order/chaos dynamic and generic evolution. The current paper offers a survey on the main aspects related to the properties of complex systems and the major computational instruments that can be engaged in the study of these complex systems. Cellular Automata (CA) are reviewed in detail as important tools in the analysis of complex interactions and emergent systems. Furthermore, we discuss potential promising perspectives worth exploring in the analysis and understanding of complex behaviour in real systems.

The structure of the paper is as follows: complex systems are defined and their main properties of emergence and self-organization are detailed; CA are reviewed from definition and rule types to CA applications; some perspectives on future work and concluding remarks are drawn at the end of the paper.

2. Complex Systems

A system is a delineated part of the universe which is distinguished from the rest by an imaginary boundary [21]. A complex system is any system containing a large number of interacting entities (agents, processes, etc.) which are interdependent. The system behaviour cannot be identified by just considering each entity and combining them, but considering how the relationships between the entities affect the behaviour of the whole. This definition applies to systems from a wide range of scientific disciplines including thermodynamics, neural networks, evolution of cooperation, economic systems of interacting trading agents, urban growth, traffic systems and many others.

A system can be defined between two extremes of connectivity among its components: order and chaos. A complex system lies within this state-space of connectivity. At the ordered extreme, elements have no connections between them. Order is the result of no interactions (and therefore no dynamics). At the chaotic extreme, every element is directly connected to every other. This means that all information percolates and is modelled conventionally as a field. A complex system is an intermediate state between an ordered system and a chaotic system.

The main features of complex systems include emergence, self-organization, evolution and adaptability. Emergence occurs when the behaviour of a system cannot be reduced to the sum of the behaviour of the parts. Self-organization

is the process by which elements interact to create spatio-temporal patterns of behaviour that are not directly imposed by external forces.

The formation of complex systems, and the structural/functional change of such systems, is a process of adaptation. Evolution is the adaptation of populations through inter-generation changes in the composition of the population. Learning is a similar process of adaptation of a system through changes in its internal patterns. The conventional notion of evolution of a population based upon replication with variation and selection with competition continues to be central. However, additional concepts such as co-evolution, ecosystems, multiple niches and hierarchical or multilevel selection have become important.

3. Emergence and Complexity

Emergence is a concept widely used in sciences, arts and engineering. A short description would state that "emergence" is the notion that the whole is not the sum of its parts [38]. For example, an individual ant or an individual neuron do not exhibit special intelligence, but gathered together and properly connected, 'spontaneous intelligence' emerges. Actually, in nature, some of the most engaging and perplexing phenomena are those in which highly structured collective behaviour emerges over time from the interaction of simple subsystems. Flocks of birds flying in lockstep formation and schools of fish swimming in coherent array abruptly turn together with no leader guiding the group [16]. The emergence of order and organization in systems composed of many autonomous entities or agents is a fundamental process.

Emerging properties are fundamental and yet familiar. According to Holland [38], emerging phenomena in generated systems are typically persistent patterns with changing components, i.e. they are changeless and changing, constant and fluctuating, persistent and shifting, inevitable and unpredictable. True emergent properties are irreducible, they cannot be destroyed or decomposed - but they appear or disappear. Unforeseeable failures and unexpected faults in software or hardware systems are special, undesired forms of emergence. It is necessary to understand the process of emergence in complex systems in order to enhance their robustness. Thus the knowledge of different types of emergence is essential for understanding and mastering complex systems.

Several attempts to classify and formalize the concept of emergence have been made [26]. Formalization attempts using grammars, formal languages and mathematics are proposed in [44, 5, 19]. The ubiquitousness of emergence in various, very different fields prohibits - for the moment - a unified formalization that would fit every form of emergence encountered. However, common characteristics across these fields [88] are as follows: the micro-macro effect

is the most important characteristic referred in the literature (which refers to the properties, behaviours, structures or patterns appearing at a higher macro level that cannot be explicitly found at the lower, micro level); radical novelty - individuals at the micro-level have no explicit representation of the global behaviour; coherence or organizational closure; interacting parts - emergents arise from interaction between parts; dynamical - emergents arise as the system evolves in time; decentralized control; two-way link: micro-level parts give rise to an emergent structure which, from a macro-level, influences the parts; robustness and flexibility.

In [26] Fromm delineates four types of emergence as follows:

- *Type I* Simple/nominal emergence without top-down feedback, only "feed forward" relationships;
- *Type II* Weak emergence including simple (positive or negative) top-down feedback (Type II weak emergence may be stable or unstable);
- *Type III* Multiple emergence with many feedbacks appears in very complex systems with many feedback loops or complex adaptive systems with intelligent agents having a large amount of external influence during the process of emergence;
- *Type IV* Strong emergence is the form of emergence which is responsible for structures on a higher level of complexity which cannot be reduced, even in principle, to the direct effect of the properties and laws of the the elementary components.

Currently, emergence as phenomenon is studied in various fields. For example its philosophy is analyzed and defended in [40]. A general framework that allows the treatment of emergence without explicit reference to the specific underlying mechanism is presented in [34]. Current practical computational issues regarding implementing emergent behaviour are discussed in [15].

Application fields for studying the emergence phenomenon have become more complex: emergence of personality from the perspective of dynamical systems is formalized in [62]; emergence of chaos in complex dynamical networks is studied in [95]; emergence issues in managing complex adaptive systems used in natural resource management in [68] and many others.

In [93] the use of evolutionary computation is advocated for the automated, simulation-based design of organic computing systems [94] with emerging behaviour.

4. Self-organization and Adaptability

From cells, organisms and ecosystems to planets, stars and galaxies almost all systems found in nature show organization. Traditional scientific fields attempt to explain these features by referencing the particular micro

properties or laws characterizing the system components. However, following "the principle of self-organization" proposed by Ashby [3], the problem can also be approached from a general perspective using properties applicable to every collections of parts of a system, regardless of size or nature. In [3] Ashby notes that a dynamical system, independently of its type or composition, always tends to evolve towards a state of equilibrium, towards an attractor.

The mathematical modelling of systems with many degrees of freedom is very challenging. With the advent of (inexpensive) computer modelling, the scientific study of self-organizing systems and adaptivity has recently grown out from a variety of disciplines including thermodynamics, cybernetics, evolutionary biology. The computer simulation assisted approach is at the base of the new domain of "complex adaptive systems", which was pioneered in the 1980's by a number of researchers from the Santa Fe Institute in New Mexico. Through computer simulations, complexity theorists can study systems consisting of many interacting components, which undergo constant change, both autonomously and in interaction with their environment. The behaviour of such complex systems is typically unpredictable, yet exhibits various forms of adaptation and self-organization.

A defining characteristic of complex systems is their tendency to self-organize globally as a result of many local interactions without explicit pressure or involvement from outside the system. The field of self-organization seeks general rules about the dynamics and evolution of global patterns that might be used to predict future organization of the system, as a result to the changes made to the underlying components [41].

A complex system moves between order and disorder without becoming fixed in either state. Such a system is considered adapting when it responds to information by changing. Complex adaptive systems persists in spite of changes in the underlying components due to how the interactions between those components are realized and changed. From changes in local interactions the system itself engages in adaptation or learning [37].

Random or locally directed changes can instigate self-organization by promoting the exploration of new state space positions. The instability that may arise from a local change triggers some sort of stress upon the whole system, causing it to move along a trajectory to a new attractor which forms the self-organized state. Changes and fluctuations allow the system to escape one basin of attraction and to enter another one.

One could expect that a system perturbed from an equilibrium state should settle to a "minimally" stable state generated by some type of optimization process. In real life, these optimized states often are highly unstable, exhibiting catastrophic breakdown events or avalanches. For example, in traffic flow

the idealized state corresponds to a uniform flow of cars with all cars moving at maximum velocity possible. This idealized system is very fragile and unsustainable as traffic jams of all sizes might occur [65].

Complex systems are often situated at the delicate balanced edge between order and disorder in a self-organized critical state. Changes or mutations of a system may take it either towards a more static configuration or towards a more changeable one. If a particular dynamic structure is optimum for the system and the current configuration is too static, then the most changeable configuration will be most successful. If the system is currently too changeable then the most static mutation will be favoured. In this way the system can adapt in both directions to converge on the optimum dynamic characteristics.

Self-organized criticality provides a general mechanism for the emergence of complex behaviour in nature with granular piles [20], traffic [65], river networks [72] and braided rivers [82], the crust of the earth and many other systems operating in this state.

To summarize, the complex adaptive systems have the following properties:

- Local, non-linear interactions of the parts result in self-organization of the system as a whole. Kauffman [42] outlines the importance of the co-evolution of agents and their environments. As an agent changes, so does the environment (including other agents, and vice versa).
- The mixed condition between order and chaos of these systems gives them stability and flexibility simultaneously.
- Rather than striving for ideal but unstable states, complex systems are organized in a way that assures good functioning but also the ability to change.

5. Cellular Automata

Cellular Automata (CA) represent useful and important tools in the study of complex systems and interactions. Introduced by von Neumann more than fifty years ago [61] as formal models for organisms capable of self-reproducing, CA are simple models of computing with an extraordinary complex behaviour. A cellular automaton is a system evolving in discrete time steps with a discrete spatial geometry - usually a regular lattice. Other CA topologies such as networks or irregular graphs have been studied [83, 86, 87, 18]. The cellular automaton is specified in terms of rules that define how it changes and evolves in time. The emergent behaviour and computational complexity of a system can be analyzed and better understood based on CA dynamics.

The main features of CA are the following:

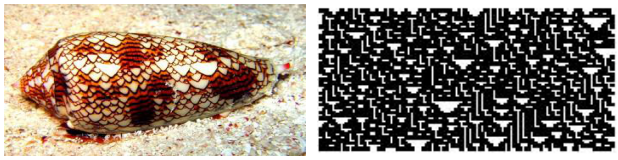


FIGURE 1. Similarities between the pattern growth of a specific shell and the dynamics of Rule 30 in 1D CA (from [71])

- *The state*: the value of each cell in CA (taken from a finite set of states). For example, cell state can have one of two values: 0 or 1 (extensively studied CA).
- *The neighbourhood*: the set of cells which interact locally. In 1D CA, a common neighbourhood of a cell refers to the so-called *first neighbours* $\{-1, 0, 1\}$ composed of the two neighbouring cells (one from the right and one from the left). In 2D CA, most used neighbourhoods include the von Neumann neighbourhood $\{(-1, 0), (1, 0), (0, -1), (0, 1)\}$ (the four neighbouring cells on top, right, left and down side) and Moore neighbourhood $\{-1, 0, 1\}^2$ (all eight neighbouring cells).
- *The transition function or the set of rules governing the evolution of CA*: how the state of a cell changes depending on its current value and that of the neighbourhood.

The global rule of a cellular automaton maps a configuration to the next time step configuration by applying the transition function uniformly in each cell. A space-time diagram is obtained by mapping the cellular automaton from one configuration to the next at all time steps [63]. It is important to emphasize that even simple rules (underlying local interactions only) give rise to complex emergent behaviour. Indeed, an important research direction refers to the analysis of CA space-time diagrams. The study of the underlying CA dynamics can potentially enable the understanding of emergent behaviour in complex systems and their computational capacity [30, 63].

CA are attractive models for many real-world complex systems from fields such as physics, biology and social sciences. For example, the natural pattern growth of a shell can be matched to the dynamics of 1D CA Rule 30 (see next subsection for a description of CA rule types) as illustrated in Figure 1 while the growth of crystals - like snowflakes patterns (see Figure 2) - can be modelled using 2D CA.

During the past few decades, CA have continuously attracted a great deal of interest from researchers and practitioners from various disciplines due to their simplicity and ability to generate highly complex behaviour. One significant property of CA is their capability to perform complex computation based

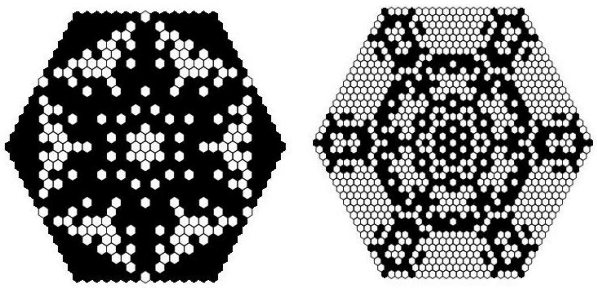


FIGURE 2. The pattern growth in snowflakes modeled with 2D CA (from [96])

on local information. Universality of computing and other theoretical aspects concerning the computational power of cellular automata are consequently considered of great importance [91].

The problem of finding CA rules able to generate a desired global behaviour is considered of great importance and highly challenging. The field of evolutionary computing offers the most promising models for addressing this inverse problem of global to local mapping [30]. The most studied problems include the density classification task [64, 50, 66, 33], the synchronization task [17, 47] and the discovery of structures such as gliders and glider guns, essential in the study of computationally universal cellular automata [75, 79].

5.1. **CA Rule Types.** Conway identified the first binary cellular automaton that supports universal computation, the well known Game of Life (or Life) [31]. This automaton evolves in an infinite two-dimensional grid of cells where each cell can be dead or alive. At each time step, the state of each cell is modified depending on the value of its eight neighbours (cells that are horizontally, vertically, or diagonally adjacent). The following simple rules are applied: any live cell with fewer than two live neighbours dies; any live cell with two or three live neighbours lives on; any live cell with more than three live neighbours dies; any dead cell with exactly three live neighbours becomes a live cell.

In [8] the authors have proved that this CA is able to simulate a Turing machine, meaning that it can compute everything that can be algorithmically computed. Life was one of the first major findings in CA field, being a great example of emergence and self-organization. Indeed, despite its very simple local rules, complex global behaviour and interesting patterns like still lives, oscillators and spaceships are evolved within Life. More interesting patterns have been further discovered: guns, which are able to create gilders and other

spaceships, puffers which leave a trail when moving or rakes, which create spaceships while moving through space.

After discovering the Game of Life, identifying other CA capable of universal computation has become a challenge for researchers. In [75, 76] evolutionary techniques are used in order to discover rules that give rise to CA able to simulate logic gates, as this could help identifying automata capable of universal computation. A similar evolutionary approach manages to identify a new cellular automaton that can implement any logic circuit and is a simulation of Life in [77, 78]. Gliders and glider guns have been further investigated in [79, 80, 81] by means of evolutionary techniques.

Wolfram has studied all 256 possible rules for 1D CA with two neighbours. The decimal interpretation of the binary number representing the rule gave the name of the resulting cellular automata. He identified several CA which exhibit an interesting complex behaviour. These CA have been classified into four categories [89]: Class 1 - CA which evolve to a homogeneous state; Class 2 - CA displaying simple separated periodic structures; Class 3 - CA which exhibit chaotic or pseudo-random behavior, and Class 4 - CA which yield complex patterns of localized structures and are capable of universal computation.

CA that exhibit Class 4 features include The Game of Life, HighLife [7], Life-3d [6], Rule 54, Rule 110 and Beehive Rule [92]. Wolfram conjectured that all CA belonging to Class 4 might be able to perform universal computation.

The 1D CA Rule 110 (see Figure 3) is one of the most intensively studied CA, due to its complex behaviour [90]. Its interesting behaviour is given by the evolution of gliders, periodic structures moving through time. Another important finding was the proof that Rule 110 automaton is capable of universal computation, emulating a universal Turing machine [91]. In 2003, Martinez shows that each glider without extensions proposed by Cook [14] can be obtained by collisions [48]. The computational complexity of Rule 110 is investigated in [60], where the prediction of Rule 110 automaton is proved to be a P-complete problem.

Rule 54 is investigated in [11, 35, 91, 1]. In [49] the author shows that all gilders produced in this automaton can be evolved by collisions, similar to Rule 110. The results obtained by this analysis are further used to evolve dynamical logic gates.

The most common approach used for updating the state of a cell is synchronous, where all cells are updated simultaneously. Taking into account the fact that the updating is not synchronized in most of the real complex systems that can be simulated using CA, the asynchronous application of rules has been recognized as an important issue and experimentally investigated in [13, 73, 84, 9]. In [22] the authors show that the behaviour induced by some rules is significantly modified by the asynchronous update of the cells, while

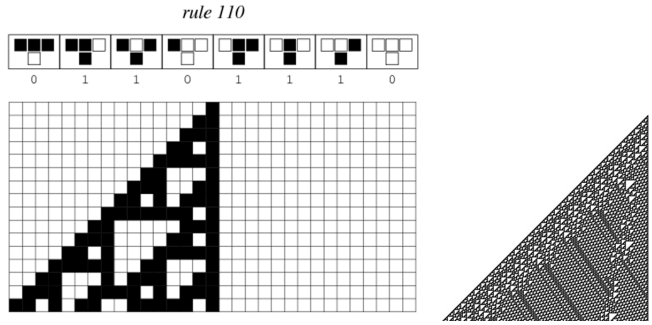


FIGURE 3. Space-time diagram of Rule 110 [91]

for some of them the CA dynamics do not change significantly. Asynchrony was induced by updating each cell with a given probability, called synchrony rate. This study is continued in [25] where directed percolation induced by asynchrony is identified. Such a phenomenon was also observed in the Game of Life [10].

The mathematical approaches to studying the behaviour induced by asynchrony in 1D CA can be found in [23, 24, 27, 28, 69]. 2D CA have been studied in [70, 71], where the authors show that the minority rule exhibits a complex behaviour when applied asynchronously. The importance of this rule and related findings might help improving understanding complex biological systems.

5.2. **Reversible CA.** In reversible CA, for each configuration there is exactly one past configuration. The importance of reversible CA is due to the fact that in such automata the information is conserved through time, making them suitable for the simulation of physical phenomena. This special class of CA has been studied starting with [59, 51, 2]. In [36] the authors investigate the local properties of cellular automata in order for them to be reversible. This study has been further developed in [52] by finding a way to detect the reversible behaviour by means of matrix representations. For this purpose, they are using the results obtained in [12] which show that any 1D CA can be transformed into an automaton of neighbourhood 2.

A method for calculating the ancestors of a configuration of states has been provided in [53, 56]. The ancestors are chosen to be non Garden-of-Eden sequences, which are sequences of states that can not be produced by any CA [51]. Two methods for calculating and classifying all possible reversible 1D CA of 3, 4, 5 and 6 states are proposed in [55]. Another type of invertible behaviour is investigated in [54, 57] and takes into account the spatial reversibility and not the temporal evolution.

5.3. **CA Applications.** The application areas of CA are many and diverse ranging from the simulation of complex systems in nature and society to modelling games and understanding social dynamics.

CA have been used to model various systems in biology (intracellular activity, cell-cell interactions, population of organisms, DNA sequences), chemistry (molecular systems, crystal growth, lattice gas automata), physics (dynamical systems, spin systems, reaction-diffusion systems), computer science (parallel processing architectures, von Neumann machines) and many others [30].

Due to the stability of CA dynamics [30], the most widely spread CA application refers to modelling dynamical physical systems. CA have been used to model the laws of physics as an alternative to differential equations [85]. Moreover, CA have been successfully engaged to model geography and urban growth [43], tumor-immune system interactions and tumor growth [45, 58, 32], controlling highly non-linear dynamical systems [4], image processing [74], traffic control [46], pattern recognition [39, 67, 29] etc.

6. Conclusions and Perspectives

The study of complex systems clearly needs to address essential aspects such as the emergence of complexity, the role of self-organization, cooperation and specialization and the relation between different complex adaptive systems. Basic properties of complex systems may include communication, cooperation, complexity, adaptation, feedback, growth, reproduction, spatial and temporal organization.

Computational approaches to the modelling and analysis of complex adaptive systems can potentially offer a better understanding of the underlying working mechanisms of many real-world systems. In this sense, the current paper offers an extensive review of cellular automata as supporting computational tools in the study of complexity. Although the volume of research material published in this area is very large and continuously growing, there are many issues to be tackled that can offer further insights into the understanding of complexity. Some of the perspectives worth exploring in the author's view include computer simulations using multi-agent modelling and evolutionary computing techniques and the investigation of complex network and cellular automata models for the analysis of complex systems.

Different types of emergence should be modelled by analyzing different interaction models within a population. A micro-level can be considered either at the individual/agent level or even within single entity individuals while a macro-level can be defined at group/population level. Different interaction models at the micro/macro level that induce emergent behavior can be studied using evolutionary computation.

Specific CA perspectives refer to the study of various CA topologies, the design of new CA types based on fuzzy neighbourhoods, the investigation of rule propagation models during CA evolution, the asynchronous application of (mixed) rules, the development of dynamic rules and neighbourhoods in CA, the investigation of various types of hybridizations within CA and novel natural computing models for CA rule detection.

Future scientific and technological developments in many fields will depend on the advances made to understand and harness complex systems behaviour. The study of complexity might facilitate the comprehension of the complex features involved in - for instance - the essential emotion-cognition-action interaction. Furthermore, this study may be really useful in dealing with the management of systems having a great number of components and dimensions. The development of universal methods for estimating and characterizing the behaviour of complex systems would be highly beneficial. A formalization would shed a light on how to (automatically) implement self-organization processes linked to applicative problems and might give us new insights about our surroundings.

7. Acknowledgments

This research is supported by Grant PN II TE 320, Emergence, auto-organization and evolution: New computational models in the study of complex systems, funded by CNCSIS, Romania.

References

[1] A. Adamatzky, *Computing in nonlinear media and automata collectives*, Bristol, Philadelphia: Institute of Physics Publishing, 2001.

[2] S. Amoroso, Y. Patt, *Decision procedures for surjectivity and injectivity of parallel maps for tessellation structures*, Journal of Computer ans System Sciences, 6 (1972), pp. 448-464.

[3] W.R. Ashby, *Principles of the self-organizing system*, Principles of Self-organization (1962), pp. 255-278.

[4] F. Bagnoli, S. El Yacoubi, R. Rechtman, *Synchronization and Control of Cellular Automata*, Cellular Automata - 9th International Conference on Cellular Automata for Research and Industry - ACRI (2010), pp. 188-197.

[5] Y. Bar-Yam, *A mathematical theory of strong emergence using multiscale variety*, Complex., 9 (2004), pp. 15-24.

[6] C. Bays, *Candidates for the game of life in three dimensions*, Complex Syst 1 (1987), pp. 373-400.

[7] D.I. Bell, *HighLife - an interesting variant of life*, 1994. Unpublished article, available on the internet: http://www.tip.net.au/d̃bell/.

[8] E. Berlekamp, J.H. Conway, R. Guy, *Winning Ways for your mathematical plays*, Academic Press, New York, 1982.

[9] H. Bersini, V. Detours, *Asynchrony induces stability in cellular automata based models*, Proceedings of the 4th International Workshop on the Synthesis and Simulation of Living Systems ArtificialLifeIV (Brooks, R. A, Maes, and Pattie, eds.), MIT Press (1994), pp. 382-387.

[10] H.J. Blok, B. Bergersen, *Synchronous versus asynchronous updating in the "game of life"*, Physical Review E 59 (1999), pp. 3876-9.

[11] N. Boccara, J. Nasser, M. Roger, *Particle like structures and their interactions in spatio-temporal patterns generated by one dimensional deterministic cellular automaton rules*, Phys Rev A; 44, 2 (1991), pp. 866-875.

[12] T. Boykett, *Comparism of Radius 1/2 and Radius 1 Paradigms in One Dimensional Reversible Cellular Automata*, http://www.algebra.uni-linz.ac.at/ tim/radiuscomp.ps.gz, Universidad Johannes Kepler, Linz, Austria, 1997.

[13] R.L. Buvel, T.E. Ingerson, *Structure in asynchronous cellular automata*, Physica D 1 (1984), pp. 59-68.

[14] M. Cook, *Introduction to the activity of rule 110* (copyright 1994-1998 Matthew Cook)

[15] S.B. Cooper, *Emergence as a computability-theoretic phenomenon*, Applied Mathematics and Computation, 215, 4 (2009), pp. 1351-1360.

[16] J.P. Crutchfield, *The calculi of emergence: Computation, dynamics, and induction*, Physica D, 75 (1994), pp. 11-54.

[17] R. Das, J. Crutchfield, M. Mitchell, J. Hanson, *Evolving globally synchronized cellular automata*, Proc. of Inter. Conf. on Genetic Algorithms (1995).

[18] C. Darabos, M. Giacobini, M. Tomassini, *Scale-Free Automata Networks are not Robust in a Collective Computational Task*, In El Yacoubi S., Chopard B. & Bandini S. (Eds.), Lecture Notes in Computer Science, Proceedings of Sixth International Conference on Cellular Automata in Research and Industry (ACRI), 4173 (2006), pp. 512 - 521.

[19] J. Deguet, L. Magnin, Y. Demazeau, *Elements about the emergence issue: A survey of emergence definitions*, ComPlexUs, 3 (2006), pp. 24-31.

[20] R.O. Dendy, P. Helander, *Sandpiles, silos and tokamak phenomenology: a brief review*, Plasma Physics and Controlled Fusion, 39:1947 (1997).

[21] P. Erdi ,T. Kiss, *The complexity of the brain: Structural, functional, and dynamic module*, In S.Wermter, J. Austin, D. J. Willshaw (Eds), Emergent Neural Computational Architectures Based on Neuroscience - Towards Neuroscience-Inspired Computing, volume 2036 of Lecture Notes in Computer Science, Springer (2001), pp. 203-211.

[22] N. Fates, M. Morvan, *An experimental study of robustness to asynchronism for elementary cellular automata*, Complex Systems 16 (2005), pp. 1-27.

[23] N. Fates, M. Morvan, N. Schabanel, E. Thierry, *Fully asynchronous behaviour of doublequiescent elementary cellular automata*, Theoretical Computer Science, 362 (2006), pp. 1-16.

[24] N. Fates, D. Regnault, N. Schabanel, E. Thierry, *Asynchronous behaviour of doublequiescent elementary cellular automata*, In Proceedings of LATIN'2006, LNCS 3887 (2006).

[25] N. Fates, *Directed percolation phenomena in asynchronous elementary cellular automata*, 7th International Conference on Cellular Automata for Research and Industry, Perpignan, France, LNCS 4173 (2006), pp. 667-675.

[26] J. Fromm, *Types and forms of emergence*, 2005, Cornell University Library, available from http://arxiv.org/pdf/nlin.AO/0506028.

[27] H. Fuks, *Non-deterministic density classiffcation with diffusive probabilistic cellular automata*, Phys. Rev. E, 66, 2 (2002)

[28] H. Fuks, *Probabilistic cellular automata with conserved quantities*, Nonlinearity, 17:1 (2004), pp. 159-173.
[29] N. Ganguly, P. Maji, S. Dhar, B. K. Sikdar, P. P. Chaudhuri, *Evolving Cellular Automata as Pattern Classifier*, ACRI 2002, LNCS 2493, Springer-Verlag Berlin Heidelberg (2002) pp. 56-68.
[30] N. Ganguly, B. K. Sikdar, A. Deutsch, G. Canright, P. P. Chaudhuri, *A Survey on Cellular Automata*, Technical Report, Centre for High Performance Computing, Dresden University of Technology (2003).
[31] M. Gardner, *Scientific American*, ISBN 0894540017, October 1970.
[32] P. Gerlee, A.R.A. Anderson, *An Evolutionary Hybrid Cellular Automaton Model of Solid Tumour Growth*, J Theor Biol., 246:4 (2007), pp. 583-603.
[33] A. Gog, C. Chira, *Cellular Automata Rule Detection Using Circular Asynchronous Evolutionary Search*, HAIS 2009, LNCS 5572 (2009), pp. 261-268.
[34] J. de Haan, *How emergence arises*, Ecological Complexity, 3, 4 (2006), pp. 293-301.
[35] J.E. Hanson, J.P. Crutchfield, *Computational mechanics of cellular automata: an example*, Physics D 103:1-4 (1997), pp. 169-89.
[36] G.A. Hedlund, *Endomorphisms and automorphisms of the shift dynamical system*, Mathematical Systems Theory, 3 (1969), pp. 320-375.
[37] J. Holland, *Hidden Order: How Adaptation Builds Complexity*, Helix Books. Addison-Wesley, Reading, Massachusetts, 1995.
[38] J. Holland, Emergence: From Chaos to Order, Oxford University Press, 1998.
[39] E. Jen, *Invariant Strings and Pattern Recognizing properties of 1D CA*. Journal of Statistical Physics, 43 (1986), pp. 243-265.
[40] J. Jost, N. Bertschinger, E. Olbrich, *Emergence*, New Ideas in Psychology, 28, 3 (2010), pp. 265-273.
[41] S. Kauffman, *The Origins of Order: Self-Organization and Selection in Evolution*, Oxford University Press, USA, 1 edition, 1993.
[42] S. Kauffman, *At home in the universe: The search for laws of self-organization and complexity*, Oxford University Press, Oxford, 1995.
[43] V. Kocabas, S. Dragicevic, *Assessing cellular automata model behaviour using a sensitivity analysis approach*, Computers, Environment and Urban Systems, 30 (2006), pp. 921-953.
[44] A. Kubik, *Toward a formalization of emergence*, Artif. Life, 9 (2002), pp. 41-65.
[45] D. G. Mallet, L. G. de Pillis, *A cellular automata model of tumor-immune system interactions*, Journal of Theoretical Biology, 239:3 (2006), pp. 334-350.
[46] S. Maerivoet, B.De Moor, *Cellular Automata Models of Road Traffic*, in Physics Reports, 419:1 (2005), pp. 1-64.
[47] A. S. Mariano, G. M. B. de Oliveira, *Evolving one-dimensional radius-2 cellular automata rules for the synchronization task*, AUTOMATA-2008 Theory and Applications of Cellular Automata, Luniver Press (2008), pp. 514-526.
[48] G.J. Martinez, H.V. McIntosh, J.C.S.T Mora, *Production of gliders by collisions in Rule 110*, Lecture Notes in Computer Science, 2801 (2003), pp. 175-182.
[49] G.J. Martinez, A. Adamatzky, H.V. McIntosh, *Phenomenology of glider collisions in cellular automaton Rule 54 and associated logical gates*, Chaos, Solitons & Fractals, 28:1 (2006), pp. 100-111.
[50] M. Mitchell, M. D. Thomure, N. L. Williams, *The role of space in the Success of Coevolutionary Learning*, Proceedings of ALIFE X - The Tenth International Conference on the Simulation and Synthesis of Living Systems (2006).

[51] E.F. Moore, *Machine models of self-reproduction*, In Arthur W. Burks, editor, Essays on Cellular Automata, University of Illinois Press (1970), pp. 204-205.

[52] J.C.S.T. Mora, S.V.C Vergara, G.J. Martinez, H.V. McIntosh, *Spectral properties of reversible one-dimensional cellular automata*, International Journal of Modern Physics C, 14:3 (2003), pp. 379-395.

[53] J.C.S.T. Mora, G. Juarez, H.V. McIntosh, *Calculating ancestors in one-dimensional cellular automata*, International Journal of Modern Physics C, Vol 15, No. 8 (2004), pp 1151-1169.

[54] J.C.S.T. Mora, G.J. Martinez, H.V. McIntosh, *Correspondence between the temporal and spacial behavior in reversible one-dimensional cellular automata equivalent with the full shift*, TUCS Gen. Publ., 32 (2004), pp. 54-63.

[55] J.C.S.T. Mora, S.V.C. Vergara, G.J. Martinez, H.V. McIntosh, *Procedures for calculating reversible one-dimensional cellular automata*, Physica D. v202 (2005), pp. 134-141.

[56] J.C.S.T. Mora, G.J. Martinez, H.V. McIntosh, *The inverse behavior of a reversible one-dimensional cellular automaton obtained by a single welch diagram*, Journal of Cellular Automata, 1:1 (2006), pp. 25-39.

[57] J.C.S.T. Mora, M.G. Hernandez, G.J. Martnez, S.V.C. Vergara, H.V. McIntosh, *Unconventional invertible behaviors in reversible one-dimensional cellular automata*, Int. J. Bifurcation Chaos Appl. Sci. Eng. 18, No. 12 (2008), pp. 3625-3632.

[58] J. Moreira, A. Deutsch, *Cellular Automaton Models of Tumor Development: A Critical Review*, Advances in Complex Systems, 5(2002), pp. 247-269.

[59] J. Myhill, *The converse of Moore's Garden-of-Eden theorem*, Proceedings of the American Mathematical Society, 14 (1963), pp. 685-686.

[60] T. Neary, D. Woods, *P-completeness of cellular automaton Rule 110*, Lecture Notes in Computer Science 4051 (2006), pp. 132-143.

[61] J. von Neumann, *The Theory of Self-Reproducing Automata*, Univ. of Illinois Press, 1966.

[62] A. Nowak, R.R. Vallacher, M. Zochowski, *The emergence of personality: Dynamic foundations of individual variation*, Developmental Review, 25(3-4) (2005), pp. 351-385.

[63] N. Ollinger, *Universalities in Cellular Automata. A (Short) Survey*, Journees Automates Cellulaires (2008), pp. 102-118.

[64] N. H. Packard, *Adaptation toward the edge of chaos. Dynamic Patterns in Complex Systems*, World Scientific (1988), pp. 293-301.

[65] M. Paczuski, K. Nagel, *Self-organized criticality and 1/f noise in traffic*, In D. E.Wolf, M. Schreckenberg, and A. Bachem, editors, Workshop — 1995 Oct: Julich; Germany, pub-WORLD-SCI:adr, World Scientific Publishing Co (1996), pp. 73-86.

[66] L. Pagie, M. Mitchell, *A comparison of evolutionary and coevolutionary search*, Int. J. Comput. Intell. Appl., 2:1 (2002), pp. 53-69.

[67] R. Raghavan, *Cellular Automata in Pattern Recognition*, Information Science, 70 (1993) pp. 145-177.

[68] C. Rammel, S. Stagl, H. Wilfing, *Managing complex adaptive systems - a co-evolutionary perspective on natural resource management*, Ecological Economics, 63, 1 (2007), pp. 9-21.

[69] D. Regnault, *Abrupt behaviour changes in cellular automata under asynchronous dynamics*, In Proceedings of 2nd European Conference on Complex Systems (ECCS), Oxford, UK, 2006.

[70] D. Regnault, N. Schabanel, E. Thierry, *Progresses in the analysis of stochastic 2D cellular automata: a study of asynchronous 2D Minority*, LNCS, vol. 4708 (2007), pp. 320-332.

[71] D. Regnault, N. Schabanel, E. Thierry, *On the analysis of "simple" 2D stochastic cellular automata*, Discrete Mathematics & Theoretical Computer Science; vol. 12:2 (2010), pp. 263-294.

[72] I. Rodriguez-Iturbe, A. Rinaldo, *Fractal river basins: chance and self-organization*, Cambridge Univ Pr, 2001.

[73] A. Roli, F. Zambonelli, *Emergence of macro spatial structures in dissipative cellular automata*, Proc. of ACRI2002: Fifth International Conference on Cellular Automata for Research and Industry, Lecture Notes in Computer Science, vol. 2493 (2002), pp. 144-155.

[74] P. L. Rosin, *Training Cellular Automata for Image Processing*, SCIA 2005, LNCS 3540, Springer-Verlag Berlin Heidelberg (2005) pp. 195-204.

[75] E. Sapin, O. Bailleux, J.J. Chabrier, *Research of a cellular automaton simulating logic gates by evolutionary algorithms*, In EuroGP03. Lecture Notes in Computer Science, 2610 (2003), pp. 414-423.

[76] E. Sapin, O. Bailleux, J.J. Chabrier, *Research of complex forms in the cellular automata by evolutionary algorithms*, In EA03. Lecture Notes in Computer Science, 2936 (2003), pp. 357-367.

[77] E. Sapin, O. Bailleux, J.J. Chabrier, P. Collet, *A New Universal Cellular Automaton Discovered by Evolutionary Algorithms*, In GECCO04. Lecture Notes in Computer Science, 3102 (2004), pp. 175-187.

[78] E. Sapin, O. Bailleux, J. Chabrier, P. Collet, *Demonstration of the Universality of a New Cellular Automaton*, International Journal of Unconventional Computing 3 (2007), pp. 79-103.

[79] E. Sapin, O. Bailleux, J. Chabrier, *Research of complexity in cellular automata through evolutionary algorithms*, Complex Syst 17:3 (2007), pp. 231-241.

[80] E. Sapin, L. Bull, A. Adamatzky, *A genetic approach to search for glider guns in cellular automata*, Proceedings of the IEEE Congress on Evolutionary Computation. IEEE Press (2007), pp. 2456-2462.

[81] E. Sapin, L. Bull, *Searching for Glider Guns in Cellular Automata: Exploring Evolutionary and Other Techniques*, In N. Monmarche et al. (eds) Artificial Evolution: Proceedings of the 8th International Conference on Evolution Artificielle. Springer (2007), pp. 255-265.

[82] V.B. Sapozhnikov, E. Foufoula-Georgiou, *Experimental evidence of dynamic scaling and indications of self-organized criticality in braided rivers*, Water Resources Research, 33, 8 (1997), pp. 1983-1991.

[83] R. Serra, M. Villani, *Perturbing the regular topology of cellular automata: Implications for the dynamics*, Proceedings of the 5th International Conference on Cellular Automata for Research and Industry (2002), pp. 168-177.

[84] B. Schonfisch, A. de Roos, *Synchronous and asynchronous updating in cellular automata*, BioSystems 51 (1999), pp. 123-143.

[85] T. Tofoli, *Cellular Automata as an Alternative to (rather than an approximation of) Diferential Equations in Modeling Physics*, Physica D, 10 (1984), pp. 117-127.

[86] M. Tomassini, M. Giacobini, C. Darabos, *Evolution of small-world networks of automata for computation*, Lecture Notes in Computer Science, Proceedings of Parallel Problem Solving from Nature, PPSN 04, 3102 (2004), pp. 672 - 681.

[87] M. Tomassini, M. Giacobini, C. Darabos, *Evolution and dynamics of small-world cellular automata*, Advances in Complex Systems, 15, 4 (2005), pp. 261 - 284.
[88] T. de Wolf, T. Holvoet, *Emergence versus self-organisation: Different concepts but promising when combined* Springer-Verlag (2005), pp. 1-15.
[89] S. Wolfram, *Undecidability and Intractability in Theoretical Physics*, Phys. Rev. Lett., 54 (1985), pp. 735-738.
[90] S. Wolfram, *Theory and Applications of Cellular Automata*, World Scientific Press, Singapore 1986.
[91] S. Wolfram, *A new kind of science*, Champaign, IL: Wolfram Media, Inc., 2002.
[92] A. Wuensche, *Self-reproduction by glider collisions; the beehive rule*, In: Pollack et al., editors. Int J. Cambridge: MIT Press (2004), pp. 286-291.
[93] R. Wuertz, J. Branke, H. Schmeck, *Evolutionary design of emergent behavior*, In Organic Computing, volume 21 of Understanding Complex Systems, Springer Berlin / Heidelberg (2008), pp. 123-140.
[94] R.P. Wuertz, *Organic Computing*, Springer (2008).
[95] H.F. Zhang, R.X. Wu, X.C. Fu, *The emergence of chaos in complex dynamical networks*, Chaos, Solitons and Fractals, 28, 2, (2006), pp. 472-479.
[96] http://demonstrations.wolfram.com/SnowflakeLikePatterns/

[(1)]Department of Computer Science, Babes-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: {cchira,anca}@cs.ubbcluj.ro
E-mail address: david.iclanzan@gmail.com

[(2)]Faculty of Economics and Business Administration, Babes-Bolyai University, Kogalniceanu 1, 400084 Cluj-Napoca, Romania
E-mail address: rodica.lung@econ.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

NONMONOTONIC SKEPTICAL CONSEQUENCE RELATION IN CONSTRAINED DEFAULT LOGIC

MIHAIELA LUPEA

Abstract. This paper presents a study of the nonmonotonic consequence relation which models the skeptical reasoning formalised by constrained default logic. The nonmonotonic skeptical consequence relation is defined using the sequent calculus axiomatic system. We study the formal properties desirable for a "good" nonmonotonic relation: supraclassicality, cut, cautious monotony, cumulativity, absorption, distribution.

1. Introduction

Default logics [10] represent a simple but a powerful class of nonmonotonic formalisms. These logical systems capture and model *defeasible inference*, a type of inference which permits that in the light of new information, already derived conclusions, called *beliefs*, to be retracted. The corresponding reasoning process is not a monotonic one: the set of derived conclusions does not increase by adding new premises.

The family of default logics represent information using a *default theory* (D, W), containing a set W of first-order formulas, called *facts* (explicit information) and a set D of inference rules, called *defaults* (implicit information). These special inference rules model laws that are true with a few exceptions. According to [7] a default has the syntax: $d = \frac{\alpha:\beta}{\gamma}$, where α, β, γ are formulas of first-order logic, α is the *prerequisite*, β is the *justification* and γ is the *consequent*. The default $d = \frac{\alpha:\beta}{\gamma}$ can be applied and thus derive γ if α is believed and it is consistent to assume β.

The differences among the variants (classical, justified, constrained, rational) of default logic are caused by the semantics of the defaults. The defaults extend a given set of facts obtaining one or more sets called *extensions* which

Received by the editors: August 10, 2010.

2010 *Mathematics Subject Classification.* 68T15,68T27,68T37.

1998 *CR Categories and Descriptors.* I.2 [**Artificial Intelligence**]: Logics in computer science – *default logics, nonmonotonic reasoning*;

Key words and phrases. default logics, nonmonotonic consequence relation, skeptical reasoning, sequent calculus.

contain the *nonmonotonic consequences* (beliefs). The reasoning process can be viewed as a process of inferring consequences of both explicit and implicit content of the knowledge base (default theory).

The extensions represent possible belief sets of an agent reasoning about the initial default theory. A *credulous reasoning* perspective means that an agents' beliefs belong to at least one extension. *Skeptical consequences* are more robust beliefs because they belong to all extensions of a theory.

The nonmonotonic reasoning can be approached from an abstract point of view, like deductive reasoning was studied using the classical inference relation/operation [1, 3, 6, 11]. The properties of a nonmonotonic inference relation/operation specific to default logics characterise the influence of the set of facts (explicit content of the knowledge base) on the nonmonotonic reasoning process.

In the papers [2, 4, 5, 8] the credulous/skeptical nonmonotonic reasoning modeled by different nonmonotonic logics, including default logics, is described using sequent calculi-based axiomatic systems.

From all versions of default logic we have chosen to study constrained default logic [9] because it has the most desirable computational properties (semi-monotonicity, commitment to assumption) regarding the application of defaults. We will prove that this version of default logic also satisfies important formal properties regarding the influence of facts on the reasoning process.

Based on the *skeptical constrained default sequent calculus* axiomatic system introduced in [5], in this paper we define the *nonmonotonic consequence relation* which models the *skeptical reasoning* formalised by constrained default logic. The formal properties desirable for a "good" nonmonotonic relation: *consistency preserving, supraclassicality, cut, cautious monotony, cumulativity, absorption, distribution*, are studied.

2. Nonmonotonic inference relations

The authors of [1, 6, 11] classified and enumerated the properties of a nonmonotonic inference operation: pure conditions, relations with the classical consequence relation, interaction with logical connectives.

A nonmonotonic inference relation denoted by "$S \mid\sim \varphi$", is defined between a set S of formulas (the premises of the inference) and a formula (the consequence of the inference).

In the following we will express the above properties in a relational manner, corresponding to a nonmonotonic (inference) consequence relation.

The *pure conditions* contain the properties inherited from the classical consequence relation ($\vdash$): *reflexivity, transitivity*, and specific properties: *cut, cautious monotony, cumulativity*, which try to weaken the monotony property

of the deductive classical relation (if $S \vdash \varphi$ and $S \subseteq V$ then $V \vdash \varphi$) which is given-up in a nonmonotonic reasoning.

- **Reflexivity**: if $\varphi \in S$ then $S \mid\sim \varphi$.
  All formulas of S are nonmonotonic consequences of S.
- **Transitivity**: if $S \mid\sim \varphi$ and $\{\varphi\} \mid\sim \psi$ then $S \mid\sim \psi$.
- **Cut**: if $S \mid\sim \varphi$ and $S \cup \{\varphi\} \mid\sim \psi$ then $S \mid\sim \psi$.
  Adding to a set S a formula which is already a nonmonotonic consequence of S does not lead to any increase in inferential power.
- **Cautious monotony**: if $S \mid\sim \varphi$ and $S \mid\sim \psi$ then $S \cup \{\varphi\} \mid\sim \psi$.
  Cautious monotony is the converse of *cut* and has the meaning: the information derived during the reasoning process and added to the set of premises does not decrease the set of nonmonotonic consequences.
- **Cumulativity**: cut + cautious monotony:
  if $S \mid\sim \varphi$ then:
  $S \mid\sim \psi$ if and only if $S \cup \{\varphi\} \mid\sim \psi$.
  Cumulativity permits the addition of lemmas to the set of premises without affecting the inferential process.
- **Reciprocity**:
  if $\forall \varphi \in V : S \mid\sim \varphi$ and $\forall \varphi \in S : V \mid\sim \varphi$ then:
  $S \mid\sim \psi$ if and only if $V \mid\sim \psi$.

The following properties characterise the relationships between the nonmonotonic inference relation ($\mid\sim$) and the classical inference relation, operation ($\vdash$, $Th(S) = \{\varphi | S \vdash \varphi\}$) and also define the interactions with the logical connectives in classical logic.

- **Supraclassicality**: if $S \vdash \varphi$ then $S \mid\sim \varphi$.
  A monotonic classical consequence of a set S of premises is also a nonmonotonic consequence of the same set S.
- **Distribution**: if $S \mid\sim \varphi$ and $V \mid\sim \varphi$ then $Th(S) \cap Th(V) \mid\sim \varphi$.
- **Left logical equivalence**:
  if $Th(S) = Th(V)$ then: $S \mid\sim \varphi$ if and only if $V \mid\sim \varphi$.
- **Right weakening**: if $S \mid\sim \varphi$ and $\{\varphi\} \vdash \psi$ then $S \mid\sim \psi$.
  Right weakening is a weak transitivity.
- **Subclassical cumulativity**:
  if $S \subseteq V$ and $(\forall \varphi \in V : S \vdash \varphi)$ and $S \mid\sim \psi$ then $V \mid\sim \psi$.
  This property is a "weak" monotony: if we add only formulas monotonically derived from the premises to the set of premises, the number of nonmonotonic consequences may increase.
- **Left absorption**: $S \mid\sim \psi$ if and only if $V \vdash \psi$, when $\forall \varphi \in V : S \mid\sim \varphi$.
- **Right absorption**:
  $S \mid\sim \psi$ if and only if $V \mid\sim \psi$,when $\forall \varphi \in V : S \vdash \varphi$.

- **Full absorption**: left absorption + right absorption.
- **Right "and"**: if $S \mid\sim \varphi$ and $S \mid\sim \psi$ then $S \mid\sim \varphi \wedge \psi$.
- **Left "or"**: if $S \cup \{\varphi\} \mid\sim \lambda$ and $S \cup \{\psi\} \mid\sim \lambda$ then $S \cup \{\varphi \vee \psi\} \mid\sim \lambda$.
- **Conditionalization**: if $S \cup \{\varphi\} \mid\sim \psi$ then $S \mid\sim \varphi \rightarrow \psi$.
- **Proof by cases**: if $S \cup \{\varphi\} \mid\sim \psi$ and $S \cup \{\neg\varphi\} \mid\sim \psi$ then $S \mid\sim \psi$.

The following relations between the above properties hold:

1. supraclassicality + cumulativity $\Longrightarrow$ full absorption;
2. reflexivity + reciprocity $\Longleftrightarrow$ cumulativity;
3. left absorption $\Longrightarrow$ right "and", right weakening;
4. right absorption $\Longrightarrow$ left logical equivalence, subclassical cumulativity;
5. distribution + supraclassicality + absorption $\Longrightarrow$ left "or", proof by cases, conditionalization;

The monotonicity is given-up in a defeasible reasoning and according to [1, 7, 11] the following properties are natural and desirable for a nonmonotonic consequence relation: *consistency preserving + supraclassicality + cumulativity + distribution.*

3. Skeptical constrained default sequent calculus

A specific sequent calculus, based on classical sequent/anti-sequent calculi enhanced with residues is used to express the skeptical constrained default reasoning [5].

For a set D of defaults we define the set of *residues* and the set of *justifications of D with respect to the set C* of formulas, using the classical anti-sequent calculus (metasybol: $\not\Rightarrow$) as follows:

$Res_D^C = \{\frac{\alpha}{\gamma} | \frac{\alpha:\beta}{\gamma} \in D, C \cup \{\beta, \gamma\} \not\Rightarrow false\}$,

$Justif_D^C = \{\beta | \frac{\alpha:\beta}{\gamma} \in D, C \cup \{\beta, \gamma\} \not\Rightarrow false\}$.

The residues corresponding to the applied defaults are monotonic rules and are used to reduce the nonmonotonic reasoning process modeled by constrained default logic into a monotonic one according to the following theorem.

Theorem 1[5]: Let $\Delta = (D, W)$ be a default theory. (E, C) is a *constrained extension* of Δ if $E = Th^{res}(W, Res_D^C)$ and $C = Th(Th^{res}(W, Res_D^C) \cup Justif_D^C)$, where $Th(\cdot)$ is the classical consequence operator and $Th^{res}(\cdot, R)$ is the consequence operator of the propositional formal system enhanced with the set R of residues. E is the *actual extension* embedded in the *reasoning context* C.

The **sequent/anti-sequent rules for residues** use the same metasymbols $(\Rightarrow, \not\Rightarrow)$ like classical logic:

(Re1) $\frac{\Gamma \Rightarrow \Psi}{\Gamma, \frac{\alpha}{\gamma} \Rightarrow \Psi}$; (Re2) $\frac{\Gamma \Rightarrow \alpha \quad \Gamma, \gamma \Rightarrow \Psi}{\Gamma, \frac{\alpha}{\gamma} \Rightarrow \Psi}$; (Re3) $\frac{\Gamma \not\Rightarrow \alpha \quad \Gamma \not\Rightarrow \Psi}{\Gamma, \frac{\alpha}{\gamma} \not\Rightarrow \Psi}$; (Re4) $\frac{\Gamma, \gamma \not\Rightarrow \Psi}{\Gamma, \frac{\alpha}{\gamma} \not\Rightarrow \Psi}$.

Let $\Delta = (D, W)$ be a default theory. A *skeptical constrained default sequent* has the syntax: $Constr; (W, D); Res \longmapsto U$. The set U of formulas is called *succedent*. The *antecedent* contains $Constr$ (a set of constraints expressed using the modalities: M-possibility and L-necessity), the default theory (W, D) and Res (the set of residues corresponding to the applied defaults).

The skeptical reasoning formalized by constrained default logic is described in [5] using the *skeptical constrained default axiomatic system*:

$Sk_{\Delta}^{cons} = \left(\Sigma_{Sk_{\Delta}}^{cons}, F_{Sk_{\Delta}}^{cons}, A_{Sk_{\Delta}}^{cons}, R_{Sk_{\Delta}}^{cons}\right)$, where $\Delta = (D, W)$ and:

$\Sigma_{Sk_{\Delta}}^{cons}$ is the alphabet;

$F_{Sk_{\Delta}}^{cons}$ contains all classical sequents/anti-sequents enhanced with residues and all skeptical constrained default sequents as below.

$A_{Sk_{\Delta}}^{cons}$ = the axioms (all classical basic sequents and anti-sequents).

$R_{Sk_{\Delta}}^{cons}$ - the classical sequent/anti-sequent rules, the rules for residues (Re1, Re2, Re3, Re4) and the sequent rules for skeptical constrained logic(S1, S2, S3) from below.

Sequent rules for skeptical constrained default logic:

(S1) $\frac{Constr^M \cup W \not\Rightarrow false \quad W \cup Res \Rightarrow U}{Constr; (W,D); Res \longmapsto U}$, $Constr^M = \{\alpha | M\alpha \in Constr\}$;

(S2) $\frac{Constr \cup \{M(\beta \wedge \gamma)\}; (W,D); Res \cup \{\frac{\alpha}{\gamma}\} \longmapsto U \qquad Constr \cup \{L\neg(\beta \wedge \gamma)\}; (W,D); Res \longmapsto U}{Constr; (W, D \cup \{\frac{\alpha:\beta}{\gamma}\}); Res \longmapsto U}$

(S3) $\frac{W \cup \{\beta \wedge \gamma | \frac{\alpha:\beta}{\gamma} \in D\} \not\Rightarrow \delta}{Constr \cup \{L\delta\}; (W,D); Res \longmapsto U}$

The above rules are based on the properties: semimonotonicity, commitment to assumption and the fact that the nonmonotonic reasoning process modelled by constrained default logic is guided by a maximal consistent reasoning context.

Theorem 2[5]: A formula X is a skeptical constrained default consequence of the default theory $\Delta = (D, W)$ if and only if the skeptical constrained default sequent $\emptyset; (W, D); \emptyset \longmapsto X$ is true (can be reduced to basic sequents/anti-sequents using Sk_{Δ}^{cons}).

4. Nonmonotonic skeptical default consequence relation for constrained default logic

Based on Theorem 2 we define the nonmonotonic skeptical constrained default consequence relation $|\sim_s^c$ and we will study its formal properties.

Let $\Delta = (D, W)$ be a default theory and X a formula:
$(D, W) |\sim_s^c X$ if the sequent $\emptyset; (W, D); \emptyset \longmapsto X$ is true using Sk_{Δ}^{cons}.

The formal properties satisfied by the nonmonotonic skeptical constrained default consequence relation $|\sim_s^c$ are established in the following theorem.

Theorem 3:

(1) Let $\Delta = (D, W)$ be a default theory with no restrictions imposed. The formal properties satisfied by $|\sim_s^c$ are as follows:
- **Consistency preserving**: the skeptical default reasoning based on a consistent set of facts will not introduce contradictions.
- **Reflexivity**: if $X \in W$ then $(D, W) |\sim_s^c X$.
  The facts are skeptical default consequences of the theory Δ.
- **Supraclassicality**: if $W \vdash X$ then $(D, W) |\sim_s^c X$.
  Constrained default logic extends the classical logic from the inferential point of view.
- **Cut**:
  if $(D, W) |\sim_s^c X$ and $(D, W \cup \{X\}) |\sim_s^c Y$ then $(D, W) |\sim_s^c Y$.
- **Full absorption**= *left absorption + right absorption*:
  - left absorption:
  $(D, W) |\sim_s^c X$ if and only if $V \vdash X$, when $\forall Y \in V, (D, W) |\sim_s^c Y$.
  - right absorption:
  $(D, W) |\sim_s^c X$ if and only if $(D, W) |\sim_s^c X$, when $\forall Y \in V, W \vdash Y$.
  This property is specific to the logical approaches used to formalize the nonmonotonic reasoning and it is not satisfied by procedural approaches.
- the properties derived from absorption: **right weakening, right and, left logical equivalence, subclassical cumulativity**.

(2) Let $\Delta = (D, W)$ be a default theory with D containing only defaults free of prerequisites $\frac{:\beta}{\gamma}$. The properties from (1) are satisfied and also:
- **Cumulativity**: **cautious monotony + cut**:
  if $(D, W) |\sim_s^c X$ then:
  $(D, W) |\sim_s^c Y$ if and only if $(D, W \cup \{X\}) |\sim_s^c Y$.
  This property is an alternative of monotony in nonmonotonic formalisms and permits the use of lemmas (nonmonotonic consequences already derived) in the reasoning process.

(3) Let $\Delta = (D, W)$ be a default theory with D containing only normal defaults free of prerequisites $\frac{:\beta}{\beta}$. The properties from (1), (2) are satisfied and also:
- **Reciprocity**:
if $\forall X \in V, (D, W) \mid\sim_s^c X$ and $\forall X \in W, (D, V) \mid\sim_s^c X$ then:
$(D, W) \mid\sim_s^c Y$ if and only if $(D, V) \mid\sim_s^c Y$.
- **Distribution**: if $(D, W) \mid\sim_s^c X$ and $(D, V) \mid\sim_s^c X$ then $(D, Th(W) \cap Th(V)) \mid\sim_s^c X$.
- the derived properties: **proof by cases, left or, conditionalization**.

Proof: The above properties are easily proved using the sequent/antisequent rules for constrained default logics (S1, S2, S3), for residues (Re1, Re2, Re3, Re4) and the rules for classical logic.

The following examples present some negative results regarding the properties of nonmonotonic skeptical default consequence relation: *distribution* and *cumulativity*.

Example 1: Let $D = \{d_1 = \frac{a:c}{c}, d_2 = \frac{\neg a:c}{c}\}$ be a set of normal defaults with prerequisites. The property "proof by cases" is not satisfied, and thus neither distribution.
Using Sk^{cons} we prove that $(D, \{a\}) \mid\sim_s^c c$, $(D, \{\neg a\}) \mid\sim_s^c c$, but $(D, \emptyset) \not\mid\sim_s^c c$.

$$\dfrac{\dfrac{\dfrac{}{\{a,c\} \not\Rightarrow false} \quad \dfrac{}{\{a, \frac{a}{c}\} \Rightarrow c}}{\{Mc\};(\{a\},\{d2});\{\frac{a}{c}\} \longmapsto c}\, S1 \qquad \dfrac{\dfrac{}{\{a,c\} \not\Rightarrow \neg c}}{\{L\neg c\};(\{a\},\{d2});\emptyset \longmapsto c}\, S3}{\emptyset;(\{a\},\{d1{=}\frac{a:c}{c}, d2{=}\frac{\neg a:c}{c}\});\emptyset \longmapsto c}\, S2$$

The above up-side-down binary tree represents the reduction of the skeptical default sequent: $\emptyset; (\{a\}, D); \emptyset \longmapsto c$, to two basic anti-sequents and a true sequent:$\{a, \frac{a}{c}\} \Rightarrow c$ which can be reduced further with $Re2$ to two basic sequents. Thus we have proved: $(D, \{a\}) \mid\sim_s^c c$.

In a similar manner we can prove that c is also a skeptical default consequence of the default theory $(\{\neg a\}, D) : (D, \{\neg a\}) \mid\sim_s^c c$.

If we try to reduce the skeptical default sequent: $\emptyset; (\emptyset, D); \emptyset \longmapsto c$, we remark that the residues rule from the sequent $\{\frac{a}{c}\} \Rightarrow c$ (if we apply first d_1) cannot be applied because there are no facts (see the following reduction tree). Similarily the default d_2 cannot be apply.

$$\dfrac{\dfrac{\dfrac{}{\{a,c\} \not\Rightarrow false} \quad \dfrac{}{\{\frac{a}{c}\} \Rightarrow c}}{\{Mc\};(\emptyset,\{d2\});\{\frac{a}{c}\} \longmapsto c}\, S1 \qquad \dfrac{\dfrac{}{\{a,c\} \not\Rightarrow \neg c}}{\{L\neg c\};(\{a\},\{d2\});\emptyset \longmapsto c}\, S3}{\emptyset;(\emptyset,\{d1{=}\frac{a:c}{c}, d2{=}\frac{\neg a:c}{c}\});\emptyset \longmapsto c}\, S2$$

The initial skeptical default sequent cannot be reduced to basic sequents/anti-sequent and thus $(D, \emptyset) \not\sim_s^c c$.

We conclude that the "proof by cases" (a particular case of distribution) property is not satisfied and neither distribution.

Example 2 [6]: shows that the cautious monotony and cumulativity are not satisfied in skeptical reasoning formalized by constrained default logic for normal default theories having defaults with prerequisites.

Let $D = \{d_1 = \frac{:a}{a}, d_2 = \frac{a:b}{b}, d_3 = \frac{b:\neg a}{\neg a}\}$ be a set of normal defaults. $E1 = Th(\{a, b\})$ is the unique extension of $(D, \emptyset)$. We have that $(D, \emptyset) \sim_s^c a$ and $(D, \emptyset) \sim_s^c b$.

The default theory $(D, \{b\})$ has as constrained default extensions $E1 = Th(\{a, b\})$ and $E2 = Th(\{\neg a, b\})$. We remark that $a \notin E1 \cap E2$.

If we consider b as a lemma, adding it to the initial set of facts will decrease the set of nonmonotonic skeptical consequences of the new default theory: $(D, \{b\}) \not\sim_s^c a$.

Thus the cautious monotony is not satisfied and neither cumulativity.

Like in the previous example we can use Sk^{cons} axiomatic system to prove $(D, \emptyset) \sim_s^c a$, $(D, \emptyset) \sim_s^c b$ and $(D, \{b\}) \not\sim_s^c a$.

5. Conclusions

The nonmonotonic consequence relation which models the skeptical reasoning formalised by constrained default logic emphasises properties that characterises the reasoning process from an abstract point of view. Using the default sequent calculus axiomatic system for expressing the skeptical default reasoning we have studied properties inherited from classical logics and some specific properties.

According to the results from Section 4 we can conclude:

- For general default theories, the nonmontonic skeptical consequence relation extends (*supraclassicality, reflexivity*) and absorbs (*absorption*) the classical consequence relation. Adding a new fact, which is already a nonmonotonic consequence, to the default theory, does not lead to any increase in inferential power (*cut*).
- The normal default theories with defaults free of prerequisites represent a special class of theories, which have associated a nonmonotonic skeptical inference relation that satisfies the desirable properties: *absorption, cumulativity, distribution* and all the properties derived from them: *right weakening, right and, left logical equivalence, subclassical cumulativity, proof by cases, left or, conditionalization.*

- The existence of prerequisites of the defaults imposes an order in the application of the defaults and it is the cause of the lack of the properties: cautious monotony, cumulativity and distribution for the corresponding nonmonotonic skeptical inference relation.

All these properties are useful from the theoretical point of view and also to increase the efficiency in the computational process of obtaining the nonmonotonic skeptical constrained consequences of a default theory.

References

[1] Antoniou, G., "Nonmonotonic reasoning", MIT Press, 1998.
[2] Bonatti, P., Olivetti, N., "Sequent Calculi for Propositional Nonmonotonic Logics", ACM Trans. Comput. Log., 2002, pp. 226–278.
[3] Lupea, M., "Nonmonotonic inference operations for default logics",ECIT - Symposium on Knowledge-based Systems and Expert Systems, Iasi, Romania, 2002, pp. 1-12.
[4] Lupea, M., "Axiomatization of credulous reasoning in default logics using sequent calculus",10-th International Symposium SYNASC 2008, IEEE Computer Society, pp.49-55.
[5] Lupea, M.,"Skeptical reasoning in constrained default logic using sequent calculus, KEPT 2009, Knowledge Engineering Principles and Techiques, Studia Universitatis, Seria Informatica, Babes-Bolyai, special issue, pp.231-234.
[6] Makinson, D: General Patterns in non-monotonic reasoning. D.Gabbay editor. Handbook of Logic in Artificial Intelligence and Logic programming, vol III, Oxford, University Press, 1994, pp. 35-105.
[7] Marek, W., Truszczynski, M., "Normal form results for default logics", Nonmonotonic and Inductive logic, LNAI Vol. 659, Springer Verlag, 1993, pp. 153–174.
[8] Milnikel, R.S., "Sequent calculi for skeptical reasoning in predicate default logic and other nonmonotonic logics", Kluwer, 2004, pp. 1–40.
[9] Schaub, T.H., "Considerations on default logics", Ph.D. Thesis, Technischen Hochschule Darmstadt, Germany, 1992.
[10] Schaub, T.H," The family of default logics", Reasoning with actual and Potential Contradictions, Handbook of Defeasible Reasoning and Uncertainty Management Systems, P.Besnard, A. Hunter, D. M. Gabbay and P. Smets, Eds. Kluwer Academic Publishers, Norwell, MA, 1998, pp.254–378.
[11] Stalnaker, R.C., "What is a non-monotonic consequence relation", Fundamenta Informaticae, 21, 1995,pp 7–21.

Babeş-Bolyai University, Cluj-Napoca, Romania
E-mail address: lupea@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

WSWRAPPER — A UNIVERSAL WEB SERVICE GENERATOR

FLORIAN BOIAN, DIANA CHINCEŞ, DAN CIUPEIU, DAN HOMORODEAN, BEATA JANCSO, AND ADINA PLOSCAR

Abstract. The need for distributed software applications is increasing day by day. Having to choose from a large variety of libraries, and to learn what each is capable of and how to use them is time consuming and overall can decreases the productivity of an engineering team. We created the WS Wrapper as a unified library on top of existing language-specific libraries, to transparently solve all dependencies, and to provide the developer with a solution that can be used in a distributed application without having to know what happens behind the scenes.

1. Introduction

Looking to the software industry nowadays, you can see that distributed applications have become a requirement. Before even thinking how a project can be developed, a good designer needs to decide what technologies should be used. In recent years, the one technology that has gained popularity is web services [1, 7, 9, 10], but even though it provides a very good design to a project, in most situations (programming languages) you can find that these are very hard to use. We created a web-services wrapper on top of existing libraries that unifies these technologies, such that they can be easily used from several programming languages. This wrapper will be available in: C# [4, 14, 20], Java [19, 11, 18], PHP [16, 19, 9] and Python [13, 17, 15] for all types of web-services: XML-RPC, SOAP, REST.

In the following sections, we are presenting the state of art in web services, what a user must do to use all types of web-services in the previously mentioned

Received by the editors: November 5, 2010.

2010 *Mathematics Subject Classification.* 68N25, 68U35.

1998 *CR Categories and Descriptors.* C.2.4 [**Computer Systems Organization**]: Computer-Communication Networks – *Distributed systems*; D.2.12 [**Software**]: Software Engineering – *Interoperability*; H.3.5 [**Information Systems**]: Information Storage and Retrieval – *On-line Information Services*; I.2.2 [**Computing Methodologies**]: Artificial Intelligence – *Automatic Programming*.

Key words and phrases. Web services, system specification, program generation, frameworks.

programming languages, and how this wrapper simplifies the process. We are also giving simple examples, just to illustrate how different it is to implement web-services in the four chosen programming languages.

The main advantage of the WS Wrapper is the simple and uniform access to the underlying web services.

2. Web Service Models

2.1. **XML-RPC.** A remote procedure call is call made by a program, over the network, to a procedure executing on a remote machine connected. Let's say that program B has implemented a *date* function and program A has not. Instead of just implementing it, assuming this would take a lot of time, program A decides to call program B's function. The function call is encoded with XML and sent over HTTP requests.

Of course, this is a very simple example. Let's say that program B has a more complex function, which needs arguments, such as the summing of two numbers.

The data types known by the XML-RPC [7] protocol are as follows: array, base64, boolean, date/time, double, integer, string, struct, nil. We would like to point these out, just to see the differences between the three types of web-services and the need for unification.

For demonstrating how XML-RPC works in the chosen programming languages, we have selected the following libraries to be used:

- C# - XmlRpcCS [20];
- Java – Apache XML-RPC [12];
- PHP – xmlrpc distribution by Dumbill [19];
- Python - xmlrpclib [17].

In C#, for using the mentioned library, we had to compile all the source files in the first place. After this, the library had to be loaded into memory, and from this moment the flow of creating a new web service is similar to the one in Java, meaning that all the public methods of a specific class are exposed as web-methods. Classes are exposed as web-services through XmlRpcServer.The client creates a XmlRpcRequest object, which is used for calling web service methods.

In Java, the client creates a XmlRpcClient object, an object of parameters that is passed to the method, and calls it. On the server side, a WebServer object is created, and to this object the user may add Java classes that will expose all their public methods as web methods. We give all the names details in this article, just to outline how many differences there are in using web-services.

In PHP, an xmlrpc_server object needs to be created, and pass to it the list of web-methods that will expose. Each web method needs to have a name, an implementation and a signature. Each web method needs to decode and encode its arguments. On the client side, we have to create an xmlrpc_client object and send xmlrpcmsgs through it.

Last, but not least, in Python a SimpleXMLRPCServer needs to be constructed, which will register the web-methods that will be exposed. To call the web-methods, the Python client needs to construct an xmlrpclib.ServerProxy and call the methods using the getattr function.

It is evident from the above, that using XML-RPC in these four programming languages can be quite different, and we have only pointed out the high level differences.

2.2. **SOAP.** SOAP[10] is another protocol which can be used in exchanging information between two applications, by calling web-service methods. SOAP sends Envelopes, which are XML messages having a well-defined structure. The "low" level protocol used for sending these messages is HTTP, same as it is for XML-RPC. One of the main features in using this protocol in implementing web-services is that, using SOAP, one can publish the web service description, using WSDL (Web-Service Description Language). [1,10]

So basically, the flow in this situation is:

- Program A wants to call a web-method from program B;
- Program A looks at the WSDL of program B;
- Program A chooses the method;
- Program A calls the method using SOAP envelops;
- Program A receives the answer in an envelope.

It is very simple to develop SOAP servers in C#[4]. You just need to create an asmx file and place this in IIS. The strange thing here is that three very similar things need to be done so that the service is deployed:

- The first line in the file must be a WebService tag;
- The web-service class needs to be annotated with WebService;
- The web-service class needs to extends WebService.

Implementing SOAP clients in C# just requires the wsdl executable, which is shipped with the .NET framework. The URL of the WSDL needs to be given to this executable as a command line argument, and the C# stubs are generated. After this, all the files need to be compiled and grouped in a library which will be used by the client at runtime.

In Java[11], for implementing SOAP web-services we need the Apache Cxf library. Similarly, but simpler than C#, Java classes will just need to

be annotated with WebService.Each method that will be exposed as a web-method will have the WebMethod annotation. After doing so, an Endpoint needs to be defined, action which will actually publish the web-service. Using the apt executable, the needed .class files for the web-service will be created.

Client-side, wsimport generates the needed stubs by specifying the web-service's WSDL. As the C# wsdl executable, wsimport generates source files, too, which afterwards need to be compiled and added in the CLASSPATH.

Using SOAP in PHP requires NuSOAP [16],which is one of the first libraries that were developed in this programming language. The basic flow is that a nusoap_server is created, to which the web-methods are registered. The client constructs a nusoap_client,and uses it to call the methods that were registered with the server.

Python's list of dependencies and requirements for running a SOAP server and client are a bit longer than the other's programming languages [13]. We need to mention that the end-user would have to manually download the exact version of libraries. This being said, the list of dependencies is: Python 2.6, PyXML 0.8.4, lxml 2.2.2, soaplib, pytz, and suds 0.3.9.

For the SOAP server, each web method is decorated with @soapmethod. The arguments of this decorator have to be a list of types that are accepted as method arguments plus the return type of the method. The Python client uses the suds library and just creates a suds.Client object, used afterwards for web-method calls.

As in the case of XML-RPC, we can see the same pattern for SOAP. Even though we could group these techniques in two by similarity: (Java, C#) and (Python, PHP), there is a great difference even between the elements in the same group.

2.3. **REST.** REST (Representational State Transfer)[9] is a style of software architecture. The best known implementation of a system conforming to REST is the World Wide Web. A RESTful web-service is a web-service implemented using HTTP and the principles of REST. Web-services that are implemented using REST are seen as a collection of resources, which can be accessed through HTTP methods.

Implementing REST web-services in C#[14] is quite simple. As for SOAP, all you need is the *asmx* file containing the web-service and web-methods, file that will be placed in IIS. The class needs to extend HTTPRequest and ProcessRequest method will decide which method will be called for each type of requests (GET, PUT, POST, DELETE). For the REST client, it just has to create a HTTPWebRequest object which will be used to call the web-methods.

Using the RESTeasy [18] JBoss distribution, for exposing REST Java web-services, firstly the class that will expose web-methods needs to be annotated with @Path, indicating the path with which the web-service can be accessed. Each web method has four annotations: @Path, indicating the path for the method, @Produces, indicating what type of data the method produces, @Consumes, indicating the input data type for the method, and the HTTP method. The Java REST client is a straightforward HTTPURLConnection, setting to it the content type and the HTTP request method.

The best PHP solution that can be used to implement REST web-services are the wrappers implemented by DaSilva[3], which are RestClient and RestServer objects. The RestServer allows the user to add methods that will be exposed as web-methods, and the RestClient allows the user to call any of the web-methods, using one of the available HTTP-methods: GET, POST, PUT, DELETE.

CherryPy[13] is the REST Web Service library for Python we chose to use out of several existing libraries. These REST Web Service libraries often are difficult to use, and in the majority of cases it is easier to implement a new Python module for REST Web Services. The Python server part needs to extend the http.server.BaseHTTPRequestHandler module, providing implementation for the four HTTP methods (GET, POST, PUT, DELETE). For the Python client, things are even simpler, meaning it just has to open an http.client.HTTPConnection and call the methods using the right HTTP-method.

We can notice the huge differences in programming languages and libraries for this type of web-service, too.

3. WS-Wrapper

Knowing the current situation, the difficulty to implement web-services in different programming languages, we thought of a solution that would simply wrap all the web-services types in these four programming languages.

As the necessity of such a solution night not be evident, we present in the following the main motivation for this work. In [5, 6, 8] we presented some preliminary considerations on particular platforms. The first reason that could come to our mind was to simplify the process one needs to follow when starting writing web-services. The WS-Wrapper not only would offer the user a nice and simple interface, which he can use to write web-services clients and servers, but would also make sure that the system he is using is compliant, meaning it has installed all the required libraries.

One of the main features of this project will be, as we have mentioned in the previous paragraph, checking that the system on which web-services are

ran is compliant. For each web-service type (XML-RPC, SOAP and REST) and each programming language, there will be scripts/installers that will make sure that all the packages required for the web-services are available, and if not it would download/install them.

The web services wrapper that we present exposes two application interfaces that work seamlessly together to provide the same high level of abstraction on each of the twelve possible language-service combinations. First interface contains the wrapper classes for the unified web service and web client. This interface is shown in Figure 1. The Client interface is presented in Figure 2. Some details of Figures 1 and 2 need to be exposed:

- serviceType and clientType are one of the integer constants defined in class.
- operationDescriptor is one of:
  - Class::methodname – for referencing a public method in a class
  - globalFunction – where it is permitted by the language
- addOperations mapping all public methods from class (a shortcut foe avoid more addOperation)
- HTTPmethod will use only for REST_service
- start starting the service
- parameters are passing into an associative array: Map Java, Hashtable C#, array PHP, Dictionary Python. All inherit a generic WSAnyType.

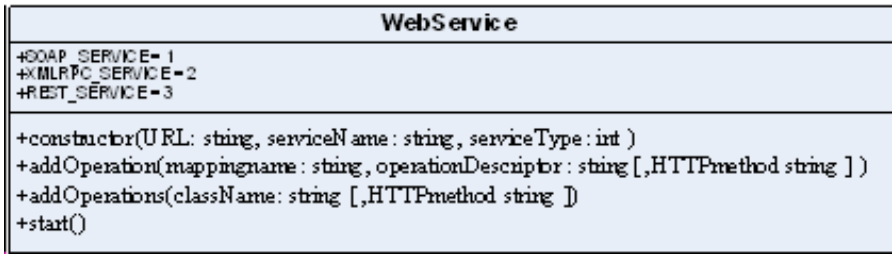


FIGURE 1. The Service interface of the WS Wrapper

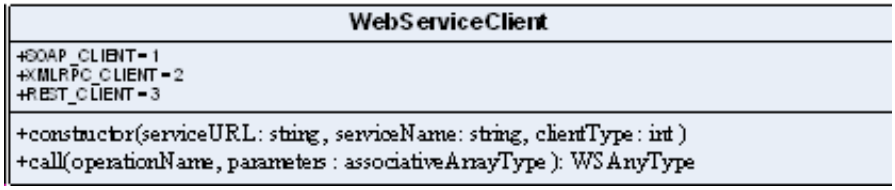


FIGURE 2. The Client interface of the WS Wrapper

The WSAnyType belongs to the other interface of classes we have been talking about and will be discussed later. So a web service object will be obtained by calling the constructor of the WebService class and specifying the URL of the service is mapping itself to, then giving a service name, and at last chosing one of the three constants (or public attributes) in the WebService class to obtain the desired type of web service. The class provides mapping methods for three types of operation mapping: mapping of a class method, mapping of a global function (where language permits it), and, for heavy services where service operations are defined in several classes, there is a method that adds to service exposed operations all public methods in a class.

The web service client is just a thin wrapper over consecrated web service clients presented in previous sections of this article. The client wrapper object is obtained in the same fashion as the web service wrapper, and the actual calling of an operation is done through the call() method, as explained in the diagram.

The second interface of objects that makes up WS Wrapper is the wrapper types' hierarchy. This hierarchy was designed to comply with all three web service types: REST, SOAP and XMLRPC. Also it is compatible with all four programming languages chosen and takes advantage of object oriented paradigm present in different forms in each of them. The diagram of this hierarchy can be observed in Figure 3. Two remarks:

- *getSoapName*() returns the actual name for SOAP case, for example "xsd:int" for an integer value. Analogously, getXmlRpc() returns "int" or "i4" in the XML-RPC case. No method is necessary in the case REST.
- wrap and unwrap methods are static and implemented in the base class for all structs. This means that all wrapping and unwrapping is already done for end user and adding a new struct type is as simple as declaring a class with need attributes, setters/getters and a constructor with suitable parameters. Every implementation of WSStruct in the four languages makes associations between struct class attributes and values in associative array based on the declaration of the actual class. For example class Person, declaring a struct with a person's data, is given.

4. An example

We have chosen to show the difference in implementing a simple web-service, Exec[2], in the old-fashioned way and using the WS Wrapper. This service exposes three methods:

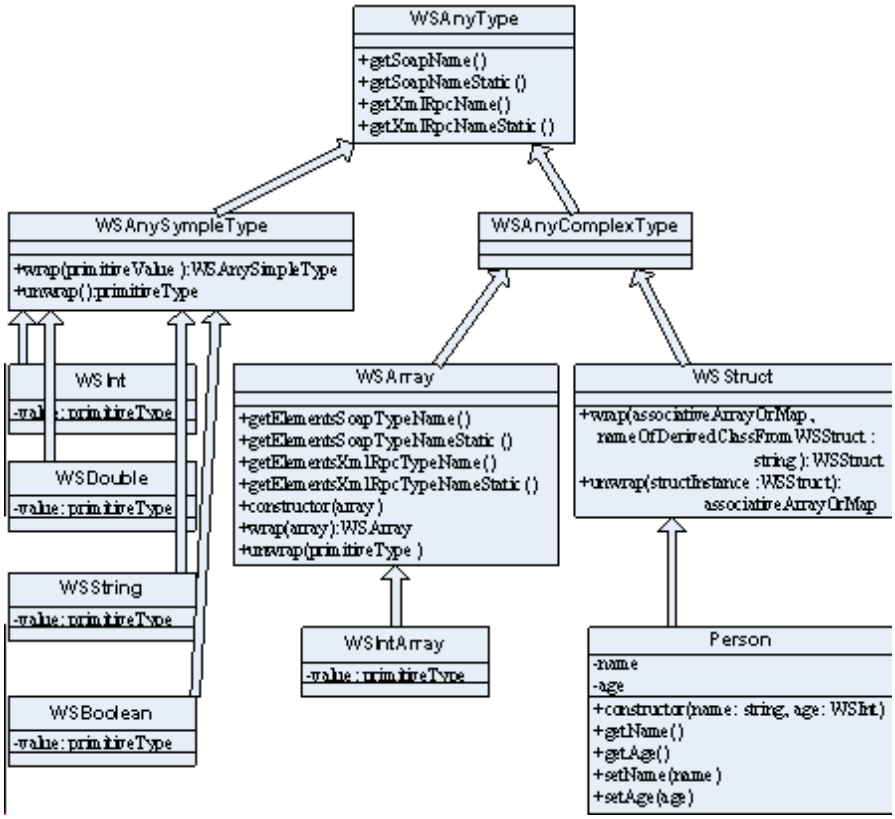


FIGURE 3. WSType classes' hierarchy

- ping()−−returns an identification string of the machine on which the service runs. This string contains the technology used for implementing the service, the name of the machine, the IP address of the machine and the current date and time of the machine.
- upcase(s)−−given as an argument the string s, this method return the same string, but transforming each letter in upper-case.
- add(a,b)−−given as argument two integer numbers a and b, the method returns the sum of a and b.

We present below brief specifications in WSWrapper of XML-RPC, SOAP, and REST services. For REST, some special specifications are necessary. The conventions we establish for implementing Exec follow the REST principles: having a fixed URL for the actions and creating mappings between HTTP methods and the exposed methods. After this service would be implemented, we can show five different methods of calling its exposed methods [2]:

URI	**HTTP method**	**HTTP body**	**Remarks**
http://address/ping	GET	-	No arguments for calling the method
http://address/upcase?s=black	GET	-	The string that has to be converted is passed as value of argument s.
http://address/upcase/black	POST		The string that has to be converted is passed as part of the URI.
http://address/add	PUT	a=66 & b=75	The two integers (66 and 75) are passed as the HTTP request body.
http://address/add/66/77	DELETE		The values of the integers to be added are part of the URL.

For example, we intended to use Java for services on the local host at the port 5678, and the class Implement has the static methods ping, upcase, and add. The services (for all three types) will be declared as follows:

```
WebService servX = new WebService("http://localhost:5678", "ExX",
                                  XMLRPC_SERVICE);
WebService servS = new WebService("http://localhost:5678", "ExS",
                                  SOAP_SERVICE);
WebService servR = new WebService("http://localhost:5678", "ExR",
                                  REST_SERVICE);
```

For mapping the implements of the methods, we specify:

```
servX.addOperation("ping", "Implement.ping");
servX.addOperation("upcase", "Implement.upcase");
servX.addOperation("add", "Implement.add");

servS.addOperation("ping", "Implement.ping");
servS.addOperation("upcase", "Implement.upcase");
servS.addOperation("add", "Implement.add");

servR.addOperation("ping", "Implement.ping", "GET");
servR.addOperation("upcase\?s=.+", "Implement.upcase", "GET");
servR.addOperation("upcase/.+", "Implement.upcase", "POST");
servR.addOperation("add", "Implement.add", "PUT");
servR.addOperation("add/-?[0-9]+/-?[0-9]+", "Implement.add",
```

```
                    "DELETE");
```

Remarks for REST: it is mandatory to specify at the name of the mapping, the prototype of the title of the URL. For this, it is necessary to use a regular expression [2,3].

After that, the service must be started, with servS.start(), servX.start(), servR.start().

Now, on the client part we intent to use PHP. First we must declare a proxy object to represent the service:

```
WebServiceClient $servX = new WebServiceClient
  ("http://TheServiceAddress:5678", "ExX", XMLRPC_CLIENT);
WebServiceClient $servS = new WebServiceClient
  ("http://TheServiceAddress:5678", "ExS", SOAP_CLIENT);
WebServiceClient $servR = new WebServiceClient
  ("http://TheServiceAddress:5678", "ExR", REST_CLIENT);
```

Finally, we can call the service methods as follows:

```
$servX->call("upcase", (array("s"=>"black");
$servS->call("ping", (array());
$servR->call("add", array("a"=>66,"b"=>75));
$servR->call("add/66/75", array();
```

5. Conclusions and Future work

Each of the technologies covered by the WS-Wrapper offer their own library for implementing web-services. We have presented above a solution that offers a unified approach to developing web services on all these platforms. To our knowledge, such unification has not yet been attempted in the existing art.

Currently, a team of PhD, MS, and BS students is working on implementing WS generators using the model from this paper. They will implement twelve projects, for three Web service technologies: XML-RPC, SOAP, REST and the programming languages: C #, Java, PHP, Python. After the first versions of these implementations, we will consider further extensions from which we mention:

(1) List of operations: the client requests the available operations (for XML-RPC and SOAP it is clear how to implement this feature, however for REST it is still a challenge)
(2) How should errors be handled?
(3) A long term proposal: specification and implementation of a code generator that is using the wrapper; code will be generated from other web services frameworks.

References

[1] Bean J., *SOA and Web Services Interface Design, Principles, Techniques and Stardards*. Elsevier, 2010.

[2] Boian F. M., *Servicii web; modele, platforme aplicatii*. Ed. Albastra, Cluj, 2010 (to appear).

[3] DaSilva S.D., *REST PHP Classes*. http://diogok.users.phpclasses.org/browse/author/529977.html

[4] Ferrara A., Mac.Donald M., *Programming .NET Web Services*. O'Reilly, 2002.

[5] Homorodean D., Boian F.M., *SOAP Web-Services in Python: Problems and Solutions*. Proceedings "Zilele Academice Clujene 2010 (ZAC2010)", Ed. Presa Universitară Clujeană, Cluj 2010, ISSN 2066-5768, pp. 105–111.

[6] Jancso B., *RESTful Web Services*. Proceedings "Zilele Academice Clujene 2010 (ZAC2010)", Ed. Presa Universitar Clujean, Cluj 2010, ISSN 2066-5768, pp. 158–163.

[7] Laurent S. St., Johnson J., Dumbill E., *Programming Web Services with XML-RPC*. O'Reilly, 2001.

[8] Ploscar A., *A Java implementation for REST-style client web service*. Proceedings "Zilele Academice Clujene 2010 (ZAC2010)", Ed. Presa Universitară Clujeană, Cluj 2010, ISSN 2066-5768, pp. 140–146

[9] Richardson L., Ruby S., *RESTful Web-Services*. O'Reilly, 2007.

[10] Tidwell D., SNall J., Kulchenko P., *Programming Web-Services with SOAP*. O'Reily, 2001.

[11] * * * *apache-cxf-2.2.7*. http://axis.apache.org

[12] * * * *apache-xmlrpc-3.1.3*. http://ws.apache.org/xmlrpc

[13] * * * *CherryPy-3.1.2.win32.exe*. http://www.cherrypy.org

[14] * * * *Developing a REST Web Service using C# - A walkthrough*. http://www.codeproject.com/KB/webservices/RestWebService.aspx

[15] * * * *Restful Python*. http://github.com/jkp

[16] * * * *Programming with NuSOAP*. http://www.scottnichol.com/nusoapprog.htm

[17] * * * *Python 2.65 Documentation*. http://www.python.org

[18] * * * *RESTEasy JAX-RS: RESTFul Web Services for Java*. http://jboss.org/resteasy

[19] * * * *xmlrpc-2.2.2*. http://ws.apache.org/xmlrpc

[20] * * * *XmlRpcCS-1.2*. http://xmlrpccs.sourceforge.net

Babes-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogalniceanu St., 400084-Cluj-Napoca, Romania

E-mail address: florin@cs.ubbcluj.ro, cdsd0192@scs.ubbcluj.ro, cdan@cs.ubbcluj.ro, hdsd0203@scs.ubbcluj.ro, bea@cs.ubbcluj.ro, adina@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

IMPLEMENTATION OF A RECOMMENDER SYSTEM USING COLLABORATIVE FILTERING

PRODAN ANDREI-CRISTIAN

Abstract. Nowadays, consumers have a lot of choices. Electronic retailers offer a great variety of products. Because of this, there is a need for Recommender Systems. These systems aim to solve the problem of matching consumers with the most appealing products for them. They do this by analyzing either the products information details (Content Based methods) or users social behavior (Collaborative Filtering). This paper describes the Collaborative Filtering technique in more detail. It then presents one of the best methods for CF: the Matrix Factorization technique. Next, it presents two algorithms used for matrix factorization. Then, the paper describes the implementation details of a framework created by us, called *Rho*, that uses Collaborative Filtering. In the end, we present some results obtained after experimenting with this framework.

1. Introduction

This paper is organized in two parts. The first part (sections 1 and 2) presents some theoretical aspects about *Recommender Systems*. It describes what they are, who uses them, a couple of examples and some mathematical background. In terms of building a user profile, the paper describes the Collaborative Filtering technique.

In the second part (section 3), the paper describes an implementation of a framework (named *Rho* and implemented by us) which can be used to model a Recommender System based on Collaborative Filtering techniques. Using the algorithms described in section 2.1, the presentation continues with the implementation details. Finally, in section 4, we comment on some results obtained after experimenting with this framework. In the end, section 5 we state our conclusions along with some ideas on how to further expand the capabilities of such systems.

Received by the editors: August 30, 2010.

2010 *Mathematics Subject Classification.* 68P20, 15A18, 15A23.

1998 *CR Categories and Descriptors.* H.3.3 [**Information Systems**]: Information storage and retrieval – *Information search and retrieval*; G.1.2 [**Mathematics of Computing**]: Numerical Analysis – *Approximation*.

Key words and phrases. collaborative filtering, Recommender Systems, svd.

We continue with this section by giving an overview of what Recommender Systems are, a general view on how do they work and some examples, used by several big companies.

Nowadays, consumers have a lot of choices. Electronic retailers offer a great variety of products. Because of this, there is a need for Recommender Systems. These systems aim to solve the problem of matching consumers with the most appropriate products.

Recommender Systems can be used on products such as books, movies, music, restaurants and TV shows. Many customers will view the same movie or purchase the same item. Every item has a couple of characteristics like genre or subject, and users can express their preferences (like of dislike) regarding them. Also, customers give feedback on products, indicating how much they liked it, so data about which product appeals to which customer, is available. Companies can analyze this data and recommend products to their customers.

Essentially, Recommender Systems compare the user's profile to some reference characteristics, and try to predict the "rating" that a user would give to an item they had not yet considered.

There are mainly two forms of data collection needed for building a users profile:

(1) **Explicit data**: ask a user to rate an item on a scale, present two items to a user and asking him/her to choose the best one, ask a user to create a list of items that he/she likes, ask a user to rank a collection of items from favorite to least favorite;
(2) **Implicit data**: observing the items that a user views in an online store, analyze item/user viewing times, keep a record of the items that a user purchases online, obtaining a list of items that a user has listened to or watched on his/her computer, analyzing the user's social network and discovering similar likes and dislikes.

The Recommender System compares the collected data to similar and non similar data collected from others and calculates a list of recommended items for the user.

The following issues represent the main challenges that must be considered when implementing a Recommender System.

- **The cold-start problem**: Recommender Systems must be capable of matching the characteristics of an item against relevant features in the user's profile. In order to do this, it must first construct a sufficiently-detailed model of the user's tastes and preferences. The cold start problem implies that the user has to dedicate an amount of effort to contribute to the construction of their user profile before the system can start providing any intelligent recommendations.

- **Sparsity**: In any Recommender System, the number of ratings already obtained is usually very small compared to the number of ratings that need to be predicted. Effective prediction of ratings from a small number of examples is important. Also, the success of the collaborative Recommender System depends on the availability of a critical mass of users.
- **Scalability**: Recommender Systems are usually designed to work on very large data sets. Therefore the scalability of the methods employed by them systems is crucial.

It is worth mentioning that big companies like Amazon, Google, Yahoo, Netflix, last.fm make use of Recommender Systems.

2. Designing a Recommender System using Collaborative Filtering

Collaborative filtering is a term coined by the developers of Tapestry, the first Recommender System ([1]). The underlying assumption of CF approach is that those who agreed in the past tend to agree again in the future. For example, a collaborative filtering or recommendation system for music tastes could make predictions about which music a user should like given a partial list of that user's tastes (likes or dislikes). These predictions are specific to the user, but use information gathered from many users.

The problem of collaborative filtering (CF) is defined in [2], as follows. The problem can be modeled by the random triplet (U, I, R), where:

- U taking values from $\{1, ..., N\}$ is the user identifier (N is the number of users),
- I taking values from $\{1, ..., M\}$ is the item identifier (M is the number of items), and
- R taking values from $X \subset \mathbb{R}$ is the rating value. Typical rating values can be binary ($X = \{0, 1\}$), integers from a given range (for example, $X = \{1, 2, 3, 4, 5\}$), or real numbers of a closed interval (for example, $X = [10, 10]$).

A realization of (U, I, R) denoted by (u, i, r) means that user u rated item i with value r. The goal is to estimate R from (U, I) such that the *root mean squared error* of the estimate,

$$RMSE = \sqrt{E\{(\hat{R} - R)^2\}} \tag{1}$$

is minimal, where $\hat{R}$ is the square estimate of R and E denotes the mean.

In practice, the distribution of (U, I, R) is not known: we are only given a finite sample, $\mathcal{T}' = \{(u_1, i_1, r_1), (u_2, i_2, r_2), ..., (u_t, i_t, r_t)\}$, generated by it.

The sample $\mathcal{T}'$ can be used for training predictors. We assume sampling without replacement in the sense that $(userID, itemID)$ pairs are unique in the sample, which means that users do not rate items more than once. Let us introduce the notation $\mathcal{T} = \{(u,i) : \exists r : (u,i,r) \in \mathcal{T}'\}$ for the set of $(userID, itemID)$ pairs. Note that $|\mathcal{T}'| = |T|$, and typically $|\mathcal{T}| \ll N \cdot M$, because most of the users rate only a small subset of the entire set of items. The sample can be represented as a partially specified matrix denoted by $\mathbf{R} \in \mathbb{R}^{N \times M}$, where the matrix elements are known in positions $(u,i) \in \mathcal{T}$, and unknown in positions $(u,i) \notin T$. The value of the matrix $\mathbf{R}$ at position $(u,i) \in \mathcal{T}$, denoted by r_{ui}, stores the rating of user u for item i. For clarity, we use the term (u,i)-th rating in general for $r_u i$, and (u,i)-th training example if $r_{ui} : (u,i) \in \mathcal{T}$.

The goal of this CF setup is to create such predictors that aim at minimizing the error (1). In practice, we cannot measure the error because the distribution of (U, I, R) is unknown, but we can estimate the error on a validation set. Let us denote the validation set by $\mathcal{V}' \subset [1, ..., N] \times [1, ..., M] \times X$, assuming sampling without replacement as defined above, and we further assume the uniqueness of $(userID, itemID)$ pairs across $\mathcal{T}'$ and $\mathcal{V}'$. We define $\mathcal{V} = \{(u,i) : \exists r : (u,i,r) \in \mathcal{V}'\}$. The assumptions ensure that $\mathcal{T} \cap \mathcal{V} = \emptyset$. If both the training set $\mathcal{T}$ and validation set $\mathcal{V}'$ are generated from the same distribution the estimate of RMSE can be calculated as:

$$\hat{RMSE} = \sqrt{\frac{1}{|\mathcal{V}|} \sum_{(u,i) \in \mathcal{V}} (\hat{r_{ui}} - r_{ui})^2} \tag{2}$$

2.1. **Matrix Factorization for Collaborative Filtering.** The idea behind Matrix Factorization (MF) is quite simple. We want to approximate matrix R (the ratings matrix) as the product of two matrices:

$$\mathbf{R} \approx \mathbf{PQ}, \tag{3}$$

where $\mathbf{P}$ is an $N \times K$ matrix and $\mathbf{Q}$ is a $K \times M$ matrix. We call P the user feature matrix and Q the item feature matrix. K is the number of features in the given factorization. $\mathbf{Q}$ and $\mathbf{P}$ typically contain real numbers, even when R contains only integers.

One way to do this is to use a techniques called *Singular Value Decomposition* (SVD). This technique is a matrix factorization technique commonly used for producing *low-rank* approximations of the initial matrix. Given an $m \times n$ matrix A, with rank r, the singular value decomposition, $SVD(A)$, is defined as:

$$SVD(A) = U \times S \times V^T, \tag{4}$$

where, where U is an $m \times m$ unitary matrix over $\mathbb{R}$, the matrix S is an $m \times n$ diagonal matrix with nonnegative real numbers on the diagonal, and V^T, an $n \times n$ unitary matrix over $\mathbb{R}$, denotes the conjugate transpose of V.

Getting back to collaborative filtering, the task is to factorize the R (rating matrix) according to SVD. Once the $m \times n$ ratings matrix R is decomposed and reduced into three SVD component matrices with k features U_k, S_k, and V_k, prediction can be generated from it by computing the cosine similarities (dot products) between m pseudo-customers $U_k \cdot \sqrt{S_k}^T$ and n pseudo-products $\sqrt{S_k} \cdot {V_k}^T$. In particular, the prediction score $P_{i,j}$ for the i-th customer on the j-th product by adding the row average r_i to the similarity. Formally,

$$P_{i,j} = r + U_k \cdot \sqrt{S_k}^T(i) \cdot \sqrt{S_k} \cdot {V_k}^T \tag{5}$$

Once the SVD decomposition is done, the prediction generation process involves only a dot product computation, which takes $O(1)$ time, since k is a constant.

However, this is unfeasible for very big and sparse matrices. An alternative to this is proposed by [2] and presented next.

2.2. **Background on the ISMF and RISMF algorithms.** In this section we give an overview of the theoretical aspects of two recommendation algorithms used in the framework implemented by us and described in section 3. These algorithms (denoted ISMF and RISMF [2]) use the following matrix factorization technique.

The notations are the same used in section 2. Let p_{uk} denote the elements of $\mathbf{P} \in \mathbb{R}^{N \times K}$, and q_{ki} the elements of $Q \in \mathbb{R}^{K \times M}$. Further, let $\mathbf{p_u}$, denote a row (vector) of $\mathbf{P}$, and $\mathbf{q_i}$, a column (vector) of $\mathbf{Q}$. Then:

$$\hat{r_{ui}} = \sum_{k=1}^{K} p_{uk} q_{ki} = \mathbf{p_u q_i}, \tag{6}$$

$$e_{ui} = r_{ui} - \hat{r_{ui}}, (u,i) = r_{ui} - \mathbf{p_u q_i} \in (T),$$

$${e_{ui}}' = \frac{1}{2} {e_{ui}}^2, \tag{7}$$

$$SSE = \sum_{(u,i)\in\mathcal{T}} {e_{ui}}^2 = \sum_{(u,i)\in\mathcal{T}} \left(r_{ui} - \sum_{k=1}^{K} p_{uk} q_{ki} \right)^2$$

$$SSE' = \frac{1}{2} SSE = \sum_{(u,i)\in\mathcal{T}} {e_{ui}}',$$

$$RMSE = \sqrt{\frac{SSE}{|\mathcal{T}|}},$$

$$(\mathbf{P}*, \mathbf{Q}*) = \arg\min_{(\mathbf{P}*,\mathbf{Q}*)} SSE' = \arg\min_{(\mathbf{P}*,\mathbf{Q}*)} SSE = \arg\min_{(\mathbf{P}*,\mathbf{Q}*)} RMSE \tag{8}$$

Here:

- $\hat{r_{ui}}$ denotes how the u-th user would rate the i-th item, according to the model;
- e_{ui} denotes the training error measured at the (u,i)-th rating;
- SSE denotes the sum of squared training errors.

Equation 8 states that the optimal $\mathbf{P}$ and $\mathbf{Q}$ minimize the sum of squared errors only on the known elements of $\mathbf{R}$.

In order to minimize $RMSE$, which is in this case equivalent to minimizing SSE', we apply a simple incremental gradient descent method to find a local minimum of SSE', where one gradient step intends to decrease the square of prediction error of only one rating, or equivalently, either ${e_{ui}}'$ or ${e_{ui}}^2$.

For the incremental gradient descent method, suppose we are at the (u,i)-th training example, r_{ui}, and its approximation $\hat{r_{ui}}$ is given.

We compute the gradient of ${e_{ui}}'$ and we obtain:

$$\nabla {e_{ui}}' = \left(\frac{\partial {e_{ui}}'}{\partial \mathbf{p_u}}, \frac{\partial {e_{ui}}'}{\partial \mathbf{q_i}} \right) \tag{9}$$

$$\frac{\partial {e_{ui}}'}{\partial \mathbf{p_{uk}}} = -e_{ui} \cdot \mathbf{q_{ki}} \tag{10}$$

$$\frac{\partial {e_{ui}}'}{\partial \mathbf{q_{ki}}} = -e_{ui} \cdot \mathbf{p_{uk}} \tag{11}$$

We update the weights in the direction opposite to the gradient:

$$p_{uk} \leftarrow p_{uk} + \gamma \cdot e_{ui} \cdot q_{ki} \tag{12}$$

$$q_{ki} \leftarrow q_{ki} + \gamma \cdot e_{ui} \cdot p_{uk} \tag{13}$$

That is, we change the weights in $\mathbf{P}$ and $\mathbf{Q}$ to decrease the square of actual error, thus better approximating r_{ui}. Here γ is the learning rate.

When the training has been finished, each value of $\mathbf{R}$ can be computed easily using Eq. 6 , even at unknown positions. In other words, the model ($\mathbf{P*}$, $\mathbf{Q*}$) provides a description of how an arbitrary user would rate any item.

This method is called **ISMF**, that is incremental simultaneous MF, according to [2] due to its distinctive incremental and simultaneous weight updating to other MF methods.

2.3. **Improving the ISMF algorithm.** The matrix factorization presented in the previous section can overfit for users with few (no more than K) ratings: assuming that the feature vectors of the items rated by the user are linearly independent and $\mathbf{Q}$ does not change, there exists a user feature vector with zero training error. Thus, there is a potential for overfitting, if γ and the extent of the change in $\mathbf{Q}$ are both small. A common way to avoid overfitting is to apply regularization by penalizing the square of the Euclidean norm of weights. Penalizing the weights results in a new optimization problem:

$$\begin{aligned} {e_{ui}}' &= \frac{{e_{ui}}^2 + \lambda \cdot \mathbf{p_u} \cdot {\mathbf{p_u}}^T + \lambda \cdot \mathbf{q_i}^T \cdot \mathbf{q_i}}{2}, \\ SSE' &= \sum_{(u,i)\in\mathcal{T}} {e_{ui}}', \\ (\mathbf{P}*, \mathbf{Q}*) &= \arg\min_{\mathbf{(P,Q)}} SSE'. \end{aligned} \tag{14}$$

Here $\lambda \geq 0$ is the regularization factor. Note that minimizing SSE'' is no longer equivalent to minimizing SSE, unless $\lambda = 0$, in which case we get back to the **ISMF**. The authors call this MF variant RISMF, that stands for regularized incremental simultaneous MF.

Similar to the ISMF approach, we compute the gradient of ${e_{ui}}'$:

$$\begin{aligned} \frac{\partial {e_{ui}}'}{\partial \mathbf{p_{uk}}} &= -e_{ui} \cdot \mathbf{q_{ki}} + \lambda * p_{uk}, \\ \frac{\partial {e_{ui}}'}{\partial \mathbf{q_{ki}}} &= -e_{ui} \cdot \mathbf{p_{uk}} + \lambda * p_{ki} \end{aligned} \tag{15}$$

We update the weights in the direction opposite to the gradient:

$$p_{uk} \leftarrow p_{uk} + \gamma \cdot (e_{ui} \cdot q_{ki} - \lambda * p_{uk}) \tag{16}$$

$$q_{ki} \leftarrow q_{ki} + \gamma \cdot (e_{ui} \cdot p_{uk} - \lambda * p_{ki}) \tag{17}$$

The training algorithm is for training the data can be found in [2]:

3. The Rho framework

This section presents a small framework implemented by us, that uses the algorithms mentioned in the previous chapter and which he called *Rho*. It can be used to train a model, analyze the results and provide recommendations for a user. Starting with the overall architecture, in which the main components of the software are presented, we then show the parameters supported by the framework on each of the four components and how to use each component.

The purpose of *Rho* is to provide a framework for Recommender Systems research, having a couple of tools for training, analyzing the results and making recommendations. It is formed of four components: **Trainer**, **BatchRunner**, **Analyzer**, **Recommender**. A diagram showing the interaction (inputs and outputs) between the components of the system is presented in Figure 1.

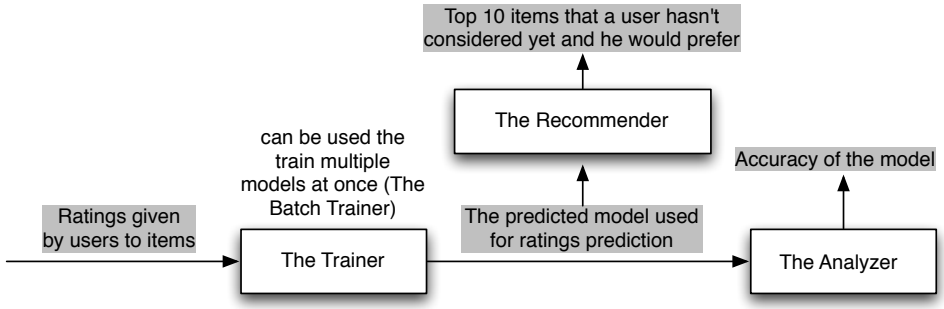


Figure 1. The components in the Rho framework

Next, we will analyze each component and explain its functionality.

3.1. **Training the model with the Trainer.** The main purpose of Rho is to allow making recommendations using matrix factorization techniques. Since it's unfeasible to factorize big sparse matrixes, the proposed algorithms uses some machine learning techniques. Discovering the model using machine learning assumes two phases: 1) train the model on a training dataset, 2) test the model on a test dataset, with the information gathered during training. As its name suggests, the Trainer component is used to train the model, on a

training dataset. The script located in `trainer.py` runs the effective training algorithm. The parameters can be changed in that file.

For now, Rho supports the 2 algorithms that were described earlier in sections 2.2 and 2.3 respectively. The parameters used to tune the algorithm are written in a Python hash format, which is easy to understand and follow. They are the following: *algorithm type* (ISMF or RISMF), *minimum improvement required to continue current feature*, *learning rate*, *regularization factor*, *number of features(factors)* to use, *initialization value for features*, *max epochs per feature*, *minimum number of epochs*, *number of items in entire training set*, *number of users in entire training set*, *number of ratings in entire training set*, *path to training dataset*, *path to test dataset*.

After running a training round, the results (the items features and user features vectors corresponding to $\mathbf{P}*$ and $\mathbf{Q}*$ respectively) are stored within the results filed in the following format:
`Features_[DAY]-[MONTH]-[YEAR]_[HOUR]-[MINUTE].txt` where,
`[DAY], [MONTH], [YEAR], [HOUR], [MINUTE]` refer to the current date of the system. The data is serialized using the `cPickle` python library, and contains `Numpy` vectors (see section 3.5 for details).

Further more, if the user choses the option to store additional results about that training round, things like RMSE, and the parameters with which the algorithm had ran, he can do that by enabling the `RECORD_RESULTS_TO_SQL` option. This stores the results in a sqlite3 database (it's format is the same as Table 4.2).

We can find the corresponding files for the user and item feature vectors by having a look at the date field.

3.2. **Analyzing the results with the `Analyzer`.** The Analyzer is a script which analyzes the results. We can use it in order to measure the efficiency of some of the tests we ran. This means, the script loads up the model, and tries to predict the ratings in the test file (learning the $\mathbb{P}, \mathbb{Q}$ models and predicting the ratings using Eq. 6). The level of acceptance, when analyzing weather a prediction was good or bad, can be adjusted using the `tolerance` parameter ($rating \in [predicted\_rating - tolerance, predicted\_rating + tolerance]$). For example, if the rating is 4.0, the predicted rating is 3.7, and the tolerance is 0.5, the prediction is considered to be a success. The parameters are also expressed in python hashes and can be configured within the script.

If no training/test files are specified, the script will load all the results existing in the database described earlier, and test them. The tolerance would be the same and can be modified within the script. This is useful when trying to analyze all the experiments carried so far.

3.3. **Carrying multiple experiments using the `BatchTrainer`.** We have found that running one experiment at a time can be a tedious and boring operation unless we really must do that. Most experiments usually involve changing some parameters and re-running the algorithm, whose speed can vary between a couple of seconds to tens of minutes.

The `BatchTrainer` overcomes this problem by allowing us to describe experiments, and ultimately creating workflows for running multiple algorithms sequentially.

We are able to state multiple parameters for different experiments in the `BatchTrainer.py` script (their meaning is the same as those described in section 3.1). The script will run the experiments one after the other, registering the results.

3.4. **A recommendation service using the `Recommender`.** `Recommender.py` provides a service for querying for user preferences. Given a user id and based on a model, the program returns a list of 10 items, that it "considers" the user would rate as high.

The parameters are the user feature file in the model learned and the user feature file in the model learned.

For example, when asking, "*What items should I recommend to user Alice ?*", the recommender would respond with a list of item ids and the corresponding predicted rating.

3.5. **Technologies used.** The implementation of *Rho* is done using Python version 2.6. The reason for using Python is that we wanted to model and test the different parameters quickly, rather then optimizing the algorithm for speed. For efficiently storing and working with the arrays and matrices, the `NumPy` library [3] was used. The code for this framework can be found in [4].

4. Results

This section presents the results obtained by us when running the implementation described in the previous section. It starts by presenting the dataset (the Movie Lens dataset [5]). Then it describes the attempt to get the correct γ and λ parameters, for minimizing the RMSE error on the models. We also analyzed how does the algorithms performs relative to the number of training epochs or features. Also we were interested in the time needed to run the experiments and the eventual correlation between it and the RMSE evolution. At the end of the chapter, we present a more comprehensive table with many values that have been obtained during the experiment.

$\lambda \setminus \gamma$	0.005	0.007	0.01	0.015	0.02
0.005	0.9333	0.8815	0.8462	0.7097	**0.7011**
0.007	0.9333	0.8815	0.8463	0.7168	0.7026
0.01	0.9333	0.8816	0.8465	0.7197	0.7027
0.015	0.9333	0.8819	0.8470	0.7263	0.7057

TABLE 1. Different RMSEs for λ - regularization factor and γ - learning rate

4.1. **Datasets.** The experiments presented in this article have been carried out using the MovieLens database [5]. MovieLens data sets were collected by the GroupLens Research Project at the University of Minnesota.

This data set consists of:

- 100,000 ratings (1-5) from 943 users on 1682 movies.
- Each user has rated at least 20 movies.

Users and items are numbered consecutively from 1. The data is randomly ordered. This is a tab separated list of user `id | item id | rating | timestamp`.

Regarding the tests dataset: `u1.base` and `u1.test` through `u5.base` and `u5.test` are 80%/20% splits of the u data into training and test data. Each of `u1, ..., u5` have disjoint test sets; this if for 5 fold cross validation (where you repeat your experiment with each training and test set and average the results).

4.2. **Experiments and results.** It has been observed that RSIMF (with regularization factor usually performs better than ISMF). As described throughout this paper, the idea is to minimize the RMSE error in order to obtain better results.

First we wanted to see what are the best learning rates and regularization factors. For that we have tried a couple of tests with different values for the two parameters, which we presented in Table 1. In order to obtain a reasonable training we have used **20 features** and **50 epochs**. We noticed that not every feature was trained 50 times. If the improvement between two epochs is not grater than 0.0001, we move on to training the next feature. The smaller RMSE obtain was 0.7011 which we have achieved for $\gamma = 0.02$ and $\lambda = 0.005$.

The running time for these experiments was about 70962 seconds, about 19.71 hours. The medium training is 0.98 hours. We show this on Figure 2. We notice that the time to train the models which yielded best RMSE is significantly longer than that used to train models with lower RMSE.

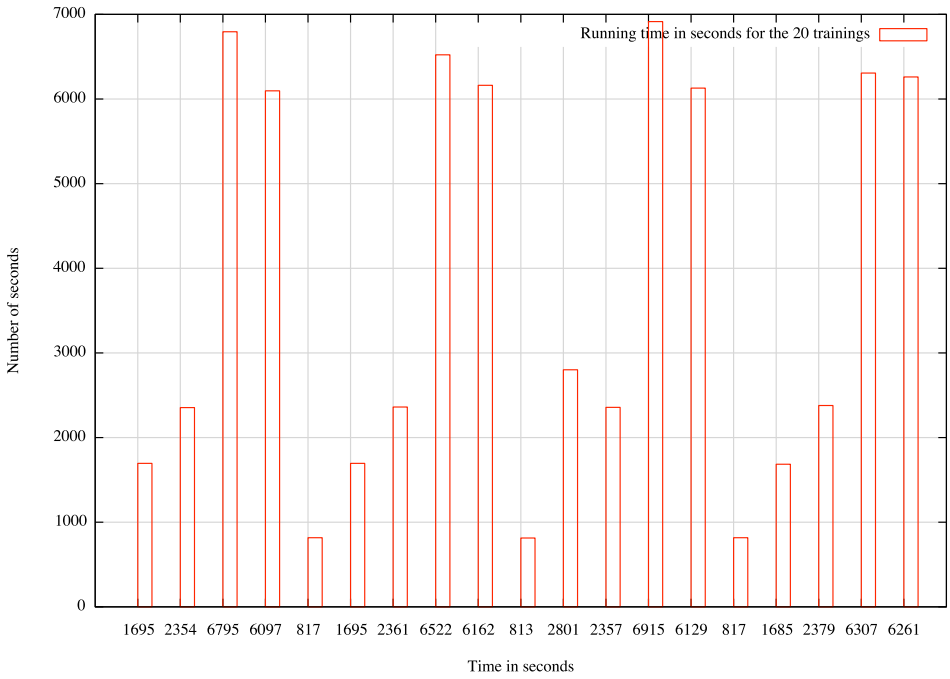


FIGURE 2. Running times for the 20 trainings. First bar corresponds to value (1,1) in Table 1, second bar - (1, 2) and so on.

We have also noticed in Figure 3 a certain periodicity on the RMSE. The values plotted here are from the same experiment presented in Table 1 and follows the same rule as Figure 2.

Table 2 offers a complete overview over the experiments which have been ran.

5. Conclusions and further work

In this thesis, we analyzed a couple of alternatives for building *Recommender Systems* with emphasis on Collaborative Filtering (CF), namely some Matrix Factorization (MF) techniques. Some of the biggest challenges imposed by such systems are scalability and sparsity. We find that often Recommender Systems have to deal with thousands of users and products and potentially hundreds of million of ratings, which result in big matrices with very few ratings - 98-99% sparse - in the case of CF and implicitly MF). It is unfeasible to factorize those matrices through classical linear algebra algorithms. In section 2.1 we describe two machine learning algorithms (named ISMF and RISMF

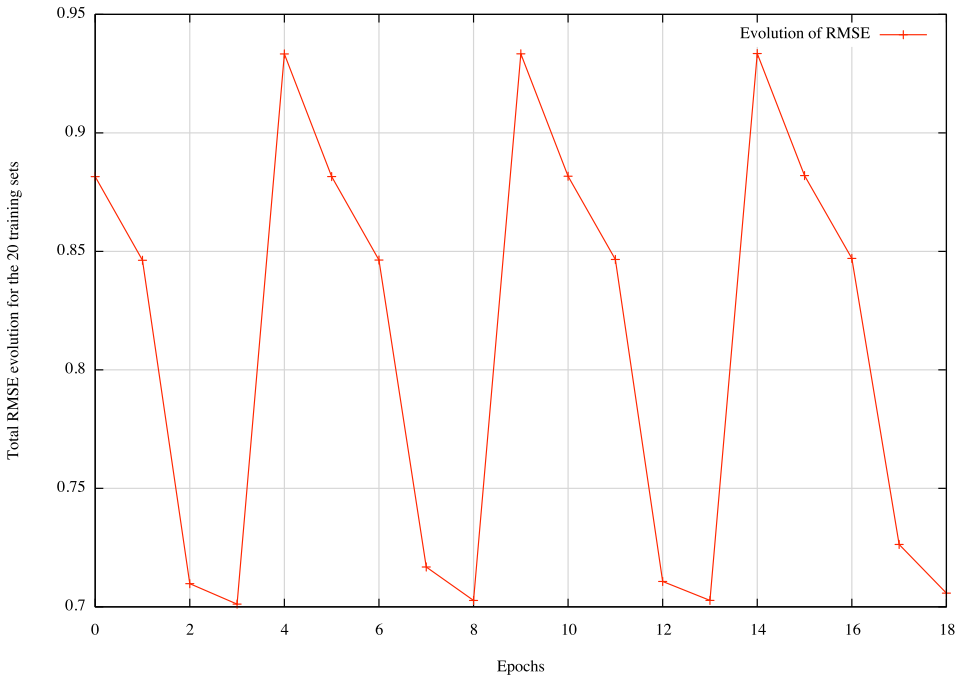


FIGURE 3. Total RMSE evolution on 20 trainings. First bar corresponds to value (1,1) in Table 1, second bar - (1, 2) and so on.

described in [2]) that do this. They overcome these problems by "guessing" the ratings (missing) values.

Further more, in order to do some experiments with these algorithms, we created a framework which allows training models based on the above mentioned algorithms, store and analyze the results in an easy to follow manner. The framework also can be used to query recommendations for best 10 items for a certain user.

Some of the results have been discussed on section 4. The best RMSE used on the MovieLends database (100.000 ratings), was 0.7011 with the RMSIF algorithm.

There are many aspects that can be improved when building Recommender Systems. From a prediction accuracy perspective, authors have tried many other techniques like SVD++ [6] or building a complex model which includes both a *k-Nearest Neighborhood* approach combined with a matrix factorization [7]. These techniques also take into account things like implicit feedback and even time ("temporal effects"). It would be interesting to enrich the existing

alg	epochs	features	time(sec)	RMSE	training DS	test DS
RISMF	50	20	826	0.9332	dataset/u1.base	dataset/u1.test
RISMF	50	20	1695	0.8815	dataset/u1.base	dataset/u1.test
RISMF	50	20	2354	0.8462	dataset/u1.base	dataset/u1.test
RISMF	50	20	6795	0.7097	dataset/u1.base	dataset/u1.test
RISMF	50	20	6097	0.7011	dataset/u1.base	dataset/u1.test
RISMF	50	20	817	0.9332	dataset/u1.base	dataset/u1.test
RISMF	50	20	1695	0.8815	dataset/u1.base	dataset/u1.test
RISMF	50	20	2361	0.8463	dataset/u1.base	dataset/u1.test
RISMF	50	20	6522	0.7168	dataset/u1.base	dataset/u1.test
RISMF	50	20	6162	0.7026	dataset/u1.base	dataset/u1.test
RISMF	50	20	813	0.9333	dataset/u1.base	dataset/u1.test
RISMF	50	20	2801	0.8816	dataset/u1.base	dataset/u1.test
RISMF	50	20	2357	0.8465	dataset/u1.base	dataset/u1.test
RISMF	50	20	6915	0.7107	dataset/u1.base	dataset/u1.test
RISMF	50	20	6129	0.7027	dataset/u1.base	dataset/u1.test
RISMF	50	20	817	0.9334	dataset/u1.base	dataset/u1.test
RISMF	50	20	1685	0.8819	dataset/u1.base	dataset/u1.test
RISMF	50	20	2379	0.8470	dataset/u1.base	dataset/u1.test
RISMF	50	20	6307	0.7263	dataset/u1.base	dataset/u1.test
RISMF	50	20	6261	0.7057	dataset/u1.base	dataset/u1.test

TABLE 2. Table format for storing training rounds results

framework with those algorithms or ideas from them. The users of the framework could either analyze their performances in terms of speed and accuracy.

Because of the large amounts of data (which keeps growing) it starts to be very hard to store it on single computing unit and perform complicated calculations. There may be useful to distribute the computations across multiple computers. Porting these algorithms on distributed frameworks like *hadoop* [**?**] (which uses the MapReduce programming model) would be beneficial. Another way would be to parallelize the factorization operations, on a single computer. In this case we could take advantage of the multicore processors.

Another thing worth mentioning is to make use of "ensemble methods" [**?**] to combine the predicted results of multiple recommender algorithms, using linear regression or other blending algorithms.

REFERENCES

[1] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35:61–70, 1992.

[2] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Scalable collaborative filtering approaches for large recommender systems. *J. Mach. Learn. Res.*, 10:623–656, 2009. ISSN 1533-7928. URL http://portal.acm.org/citation.cfm?id=1577069.1577091.

[3] Open Source Library. Numpy library. ***. URL http://numpy.scipy.org/.

[4] Cristian Prodan. The rho framework. 2010. URL http://github.com/christian/Rho.

[5] MOVIELENS-DATA. *MovieLens dataset.* URL http://www.grouplens.org/node/73.

[6] R. M. Bell and Y. Koren. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *ICDM '07: Proceedings of the 2007 Seventh IEEE International Conference on Data Mining*, pages 43–52, Washington, DC, USA, October 2007. IEEE Computer Society. ISBN 0-7695-3018-4. doi: 10.1109/ICDM.2007.90. URL http://dx.doi.org/10.1109/ICDM.2007.90.

[7] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proc. Int. Conf. on Knowledge Discovery and Data Mining*, pages 426–434, 2008.

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Cluj-Napoca, Romania

E-mail address: prodan.cristian@gmail.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

GLOBAL SEARCH AND LOCAL ASCENT FOR LARGE COURNOT GAMES

MARIA SĂRĂŞAN[(1)], TUDOR DAN MIHOC[(1)], RODICA IOANA LUNG[(2)], AND D. DUMITRESCU[(1)]

Abstract. Equilibria detection in large games is a fundamental problem in computational game theory. A memetic algorithm, Global Search and Local Ascent (GSLA), is proposed. GSLA's performance is evaluated by means of numerical experiments within the framework of a Cournot game involving up to 100 players and by comparison with an evolutionary multiobjective optimization algorithm adapted for Nash equilibria detection.

1. Introduction

Mathematical games share a lot of similarities with Multi-Objective Optimization Problems (MOOPs). A non-cooperative game consists of a set of players, a set of actions available for each player, and the corresponding payoff functions for each player respectively. Similar with a multi-objective optimization problem each player seeks to maximize her corresponding payoff function in order to increase her profits.

The most common solution concepts for the two types of problems are the Pareto optimal solution for MOOPs [5] and the Nash equilibria for games [10], respectively. While the Pareto dominance relation allowed an extensive study of MOOPs from an evolutionary perspective [4], for the Nash equilibrium only recently an appropriate fitness concept based on non-domination has been developed [6, 7]. This was attained with the use of a generative relation capable of guiding an evolutionary algorithm towards a game's Nash equilibrium. Thus existing evolutionary algorithms designed for multiobjective optimization can be adapted for searching Nash Equilibria.

Our aim is to compute equilibria for games involving large number of players. Preliminary experiments with [6, 3] show that common algorithms

Received by the editors: August 20, 2010.

2000 *Mathematics Subject Classification.* 91A80, 68T01.

1998 *CR Categories and Descriptors.* I.2.8 **Computing methodologies**: Artificial Intelligence *Problem Solving, Control Methods, and Search - Heuristic Methods*.

Key words and phrases. Nash equilibrium, generative relation for equilibria, large non-zero games.

designed for MOOPs are unsuccessfully answering that challenge. A memetic algorithm that aims to combine evolutionary techniques and local improvement procedures, named Global Search and Local Ascent (GSLA) algorithm, is introduced. Numerical experiments conducted for Cournot games with up to 100 players show the potential of GSLA.

The paper is organized as follows: in the next section some basic notions from game theory are presented as well as the Nash generative relation. In the third section the GSLA algorithm is described, and the algorithm's founding ideas and building blocks are illustrated. Next, numerical experiments conducted with the Cournot oligopoly offer an assessment of GSLA's performance, as well as a comparison to NSGA-II's. The paper ends with conclusions, acknoledgments and bibliographical references.

2. Prerequisites

Some fundamental concepts related to game theory are described in this section.

2.1. **Strategic game.** A finite strategic non-cooperative game [6, 10] is defined as a system $\Gamma = ((N, S_i, u_i), i = 1, n)$ where:

- $N = \{1, ..., n\}$ is a set of n players;
- for each player $i \in N$, S_i represents the set of actions (pure strategies) available to him, $S_i = \{s_{i_1}, s_{i_2}, ..., s_{i_{m_i}}\}$;
- $S = S_1 \times S_2 \times ... \times S_N$ is the set of all possible situations of the game;
- an element of S is a strategy profile (or strategy) of the game;
- for each player $i \in N$, $u_i : S \to R$ represents the payoff function.

Let s^* be a strategy profile. Denote by (s_{i_j}, s^*_{-i}) the strategy profile obtained from s^* by replacing the strategy of player i by s_{i_j} i.e.

$$(s_{i_j}, s^*_{-i}) = (s^*_i, s^*_2, ..., s^*_{i-1}, s_{i_j}, s^*_{i+1}, ..., s^*_n).$$

S_{-i} denotes a strategy profile of every player except i. It is important to notice that, for a given game outcome, each player's payoff is not determined solely by his own action, but rather by the combination of his chosen strategy and all the other players' actions.

2.2. **Nash Equilibrium.** In game theory, the prevalent solution concept of a non-cooperative game is Nash Equilibrium (NE) [9, 8]. Thus, a Nash equilibrium is a strategy profile reflecting a state of the game from which no single player can improve his payoff by unilaterally modifying her strategy.

Definition Profile strategy s^* is a Nash equilibrium if the inequality $u_i(s^*) \geq u_i(s_{i_j}, s^*_{-i})$ holds for every action s_{i_j} of every player i, $s_{i_j} \in S_i$.

2.3. **Nash ascendancy relation.** Given two strategy profiles s' and s'', the instituted operator $k(s', s'')$ assigns to the pair the cardinality of the set

$$k(s'', s') = card\{i \in \{1, ..., n\} | u_i(s'_i, s''_{-i}) \geq u_i(s''), s'_i \neq s''_i\}$$

i.e. $k(s', s'')$ denotes the number of individual strategies from s' which replaced in s'' give better payoff for the corresponding player.

$k(s'', s')$ measures the sensitivity of s'' with respect to perturbations supplied from s'. The lower sensitivity, the higher is the stability of s'' with respect to s'.

We may use

$$m(s'', s') = n - k(s'', s')$$

as a measure for the relative quality of s'' with respect to s'.

Let us consider a generative relation R_N on $S \times S$: $(s', s'') \in R_N$ if and only if s' is better than s'' with respect to m, i.e. $m(s', s'') > m(s'', s')$.

Therefore $(s', s'') \in R_N$ if and only if $k(s', s'') < k(s'', s')$.

A strategy profile s' ascends (dominates in Nash sense) another strategy profile s'' if there are less players capable of augmenting their profit by changing their strategy from s' to s'', than the reverse.

A strategy profile s is considered non-dominated in Nash sense if $\nexists s' \in S$: $(s', s) \in R_N$.

As stated in [6], all non-dominated strategy profiles with respect to R_N represent NE.

3. Proposed method

The above-mentioned Nash-based domination concept facilitates the comparison of two solutions, ascertaining that one is "closer" than the other to the equilibrium. Applying this domination concept in the framework of evolutionary computation, by using the generative relation within the EA comparison procedures, leads to algorithms for search and detection of a game's Nash equilibria, as the algorithm will converge to the Nash non-dominated solutions. An evolutionary model that incorporates a global search within the game solutions' space is proposed. This search is performed using a genetic algorithm that has been adapted so it detects a game's Nash equilibrium. Then, a local search algorithm is used, aimed to improve the quality (thus reducing the "distance" to the NE) of the new population's best candidate solution.

The proposed method, named Global Search and Local Ascent (GSLA), is constructed as a memetic algorithm, a GA's hybridization with a stochastic local optimization technique.

3.1. **Global Search.** Using a Genetic algorithm (GA) [1, 4] a population of individuals is evolved, by applying a set of particular rules, towards a state which maximizes the population's fitness. Because of the many advantages inherent to this technique, as well as its compatibility with the problem of detecting a game's NE, the model of a GA constitutes the basis of the proposed evolutionary method.

The algorithm uses a real-coding of chromosomes: each chromosome represents a strategy profile and each individual gene indicates the strategy that a player chooses to play.

In its run, the algorithm first triggers the creation of a random initial population of candidate solutions. Each generation's individuals are assigned – via a Nash-ascendancy fitness evaluation – an factor that indicates the quality of the candidate's solution within the population. Less dominated individuals (with a smaller ascendancy factor) represent better-quality solutions.

Selection of individuals for crossover is performed tournament-style offering the benefit of a stricter selection pressure while still not completely discounting weaker individuals. Selected individuals undergo a convex crossover, their offspring possibly also undergo uniform mutation to preserve genetic variety within the population.

The offspring population, with the same size with the parent population, replaces the old population, discarded in favour of the young one. The best individual of the old population is also kept in the new population – a partial elitist approach which avoids the risk of premature convergence associated with full elitism, but still ensures the further exploration of the current best solution and its neighbours within the next generation. The algorithm stops after a preset number of generations or when the number of fitness function evaluations is reached.

3.2. **Local search.** GAs are usually competent at detecting adequate global solutions, but are often lacking precision. On the other hand, local search methods, such as hill climbing, are quite successful at detecting the optimum in a bounded section. An approach wherein the GA technique and hill climbing are alternated should increase the search' efficiency while surmounting the deficiencies inherent in GA as well as in hill climbing.

Within the proposed model, after reaching a specified number of iterations, before resuming with the initiation of a new generation cycle, the current generation's fittest individual is in fact set apart and submitted to undergo a local refinement process (before being added to the new population).

This process is constructed as follows:

Let $s = (s_1, ..., s_n)$ denote the strategy profile represented by the fittest individual. For a random value $i \in \{1, ..., n\}$, the gene s_i is modify by summing

a value $\pm e$ ($\pm$ is randomly decided upon). For the potential offspring s', we would have $s'_i = s_i \pm e$.

If the new strategy profile Nash-ascends its parent, it takes its place and the refinement process is applied again.

If the new strategy profile does not dominate its parent a new position i is considered for mutation. The following positions are obtained by either increases or decreases (a randomly made decision as well) the precedent i value by 1, depicting a circular movement along the chromosomal genes.

If, however, within the current strategy profile s no i was found so that player s' could increase her payoff by altering her strategy with e, the search ends.

4. Numerical experiments

The current section offers a brief view into the performance analysis of the GSLA algorithm, as well as a performance comparison with multi-objective optimization evolutionary algorithm NSGA-II [3], adapted to search for a game's Nash equilibria instead of the Pareto optimal solutions.

The Cournot oligopoly model has been considered for the numerical experiments [2]. Results show the two algorithms – GSLA and NSGA-II – individual performances in detecting the game's NE. A graphical visualization of the obtained values also illustrates how the two algorithms behave.

4.1. **Cournot oligopoly.** Consider n competing companies, all producing a single product, and the product quantity produced by each firm is denoted by q_i respectively.

A game strategy profile is

$$s = (q_1, ..., q_n).$$

Let $Q = \Sigma q_i$ denote the total quantity for that product available on the market. The market clearing price is

$$P(Q) = \begin{cases} a - Q, & \text{for } Q < a, \\ 0, & \text{for } Q \geq a. \end{cases}$$

where a denotes the maximum number of the product that are possible to be sold on the market.

We assume that any firm i's full expenditure for producing the quantity q_i is $C_i\ (q_i) = cq_i$, $c < a$. Working with the supposition that all firms choose their quantities at the same time, the payoff for each firm i is its profit:

$$\begin{aligned} \pi_i(q_1, ..., q_n) &= q_i P(Q) - C_i(q_i) \\ &= q_i \left[a - (q_1 + ... + q_n) - c\right], i = 1, ..., n. \end{aligned}$$

	GSLA		NSGA II	
No. of players	Average distance	Standard deviation	Average distance	Standard deviation
2	0.000074	0.000050	0.000757	0.001712
5	0.000548	0.001124	1.321731	1.758642
10	0.002871	0.004088	7.467808	2.818299
20	0.265021	0.230464	18.976990	4.420855
50	0.379441	0.256289	26.378490	3.942610
100	0.447361	0.029569	89.040740	16.546090

TABLE 1. Average distance to Nash equilibrium in 30 runs for GSLA and NSGA II

The Cournot oligopoly in the form presented here has one Nash equilibrium $q^* = (q_i{}^*)_{(i=\overline{1,n})}$, and

$$q_i{}^* = \frac{a-c}{n+1}, \forall i \in \{1, ..., n\}$$

This model is used for numerical simulation with different numbers of players in order to asses GSLA's and NSGA-II's performances in detecting the game's Nash equilibrium.

4.2. **Parameter setting.** Both algorithms GSLA and NSGA-II are tested for 2, 5, 10, 20, 50 and 100 players.

Population size was set to 75, the maximum number of generations to 250, the initialization domain to [0, 100]. The tournament size is 30, crossover probability 0.5 and probability of mutation 0.2.

Cournot parameters a and c were set to 205 and 3 respectively.

30 runs were completed for each algorithm with different random seeds, and the average and standard deviation of the best obtained solutions' distance to the NE was calculated.

4.3. **Numerical results.** The numerical results obtained for games with 2, 5, 10, 20, 50 and 100 players with the two methods – GSLA and NSGA II are depicted in Table 1. The GSLA performance is superior, the average distance to Nash equilibria being 0.44 even for the game having 100 players while NSGA-II has an average distance 89.04. A graphical illustration of the differences between the two methods is given in Figure 1.

5. Conclusion and Further work

A new hybrid method, called Global Search and Local Ascent algorithm is proposed. GSLA combines a generational evolutionary algorithm with a hill

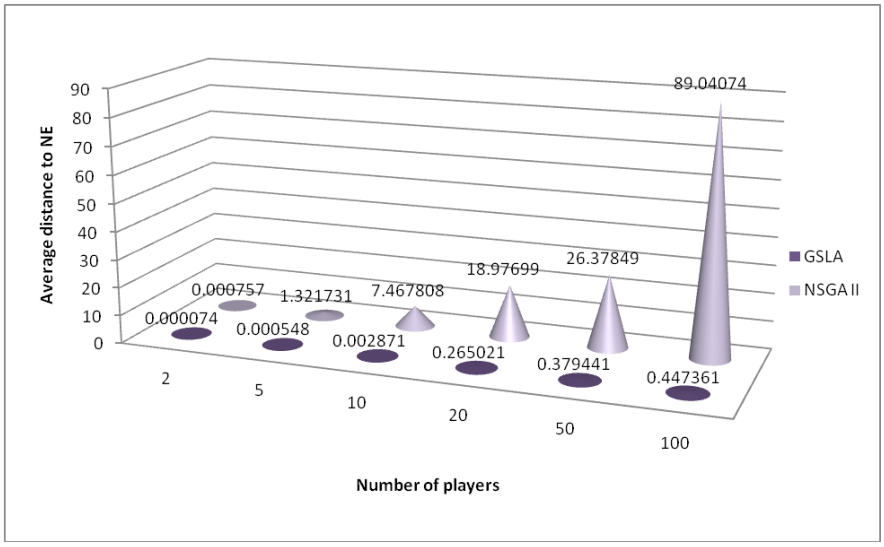


FIGURE 1. Comparison of average distance to NE for GSLA and NSGA II

climbing procedure in order to compute the Nash equilibria for a large non-cooperative game. The search is guided using a generative relation allowing the comparison of two strategy profiles within a game.

The efficiency of the method is evaluated using a Cournot oligopoly taking into account up to 100 firms.

Results are compared with a modify version of the NSGA-II algorithm. For the given setting, GSLA significantly outperforms NSGA-II, suggesting a very good search potential.

Further work will consist in exploring this potential by using GSLA for equilibria detection in games characterized by the existence of multiple Nash equilibria.

6. Acknowledgements

This research is supported partially from the Sectorial Operational Programme Human Resources Development, Contract POSDRU 6/1.5/S/3 "Doctoral studies: through science towards society", Babeş - Bolyai University, Cluj - Napoca, România, and by CNCSIS UEFISCSU, project number PNII IDEI 2366/2008, and by the CNCSIS Grant ID508 *"New Computational paradigms for dynamic complex problems"* funded by the MEC.

References

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*, Oxford Univ. Press, 1996.

[2] A. F. Daughety, *Cournot oligopoly: characterization and applications*, edited by Andrew. F. Daughety, Cambridge University Press, Cambridge; New York, 1988.

[3] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan, *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II* , Springer, 2000, pp. 849-858.

[4] D. E. Goldberg,*Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley, 1989.

[5] H. Ishibuchi, N. Tsukamoto, Y. Nojima, *Evolutionary many-objective optimization: A short review*, in Proc. IEEE Congr. Evol. Comput., Hong Kong, Jun. 2008, pp. 2424-2431.

[6] R. I. Lung, D. Dumitrescu, *Computing Nash equilibria by means of evolutionary computation*, in Int. J. of Computers, Communications & Control, vol. III, no. suppl. issue, 2008, pp. 364-368.

[7] R. I. Lung, T. D. Mihoc, D. Dumitrescu, *Nash Equilibria Detection for Multi-Player Games*, WCCI 2010 IEEE World Congress on Computational Intelligence, July, Barcelona, Spain, 2010, pp.3447-3451.

[8] R. D. McKelvey, A. McLennan, *Computation of Equilibria in Finite Games*, in Handbook of Computational Economics, H. M. Amman, D. A. Kendrick, J. Rust, Eds. Elsevier, Amsterdam, 1996, vol. 1, ch. 2, pp. 87-142.

[9] J. F. Nash, *Non-cooperative games*, in The Annals of Mathematics, Second Series, vol. 54, issue 2, 1951, pp. 286-295.

[10] V. V. Vazirani, N. Noam, T. Roughgarden,É. Tardos,*Algorithmic Game Theory*, Cambridge University Press, Cambridge, 2007.

(1) Babeş-Bolyai University, Faculty of Mathematics and Computer Science, 1 Mihail Kogălniceanu St., RO-400084 Cluj-Napoca, Romania

(2) Babeş-Bolyai University, Faculty of Economics and Business Administration, 58-60 Teodor Mihaly St., RO-400591 Cluj-Napoca, Romania

E-mail address: smsd0362@scs.ubbcluj.ro

E-mail address: mihoct@cs.ubbcluj.ro

E-mail address: rodica.lung@econ.ubbcluj.ro

E-mail address: ddumitr@cs.ubbcluj.ro

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

METADATA FOR CONTENT-BASED IMAGE RETRIEVAL

ADRIAN STERCA AND DANIELA MIRON

Abstract. This paper presents an image retrieval technique that combines content based image retrieval with pre-computed metadata-based image retrieval. The resulting system will have the advantages of both approaches: the speed/efficiency of metadata-based image retrieval and the accuracy/power of content-based image retrieval.

1. Introduction

Traditionally, there are two approaches for image searching and retrieval. The first one uses pre-computed metadata and involves a textual search in this metadata and the other one uses computer vision for extracting various features from images and comparing them and so, it is a content-based search.

The first approach, also known as *metadata-based image retrieval* (i.e. MBIR), implies that images are annotated with keywords that are then stored in traditional databases for later access to the image. While metadata-based searching is the most prominent search technique used in the Intrenet (e.g. for example it is used by the Google search engine) due to its low computational cost, its many limitations led to the development of alternative methods of image searching and retrieval that would solve these limitations. The efficiency of metadata-based image retrieval equals the search efficiency in the underlying metadata database.

Therefore, the second method for searching and retrieving images is built starting from the disadvantages of metadata-based search techniques: as databases become increasingly large, a few words describing the image are not sufficient to capture the entire contents of the image . Moreover, performing annotations involves (partially or totally) the human work, which is subjective in terms of image descriptions, but also time consuming. This technique is

Received by the editors: September 10, 2010.

2010 *Mathematics Subject Classification.* 94A08, 68P20.

1998 *CR Categories and Descriptors.* I.4.0 [**Image Processing and Computer Vision**]: General – *Image processing software*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval – *Search process*.

Key words and phrases. image retrieval, content-based image retrieval, metadata-based image retrieval, image feature extraction.

known as *content-based image retrieval* (i.e. CBIR). This search technique applies computer vision methods for extracting features from digital images and using these features in searching and retrieval of images. In this context, the term *content* refers to any of the characteristics of images: color, shape, texture, edge or other information that is encoded in the image itself and not in metadata.

In content-based search, the response time is large (i.e. low efficiency), but the power of the search (i.e. expressiveness of search) is high. In contrast, for metadata-based search the efficiency is high (i.e. lower response time), but the search power is rather small, since we can not fully capture the contents of an image in just a few keywords.

This paper tries to combine the two aforementioned techniques, namely content-based image retrieval and metadata-based image retrieval. More precisely, we are building metadata to help the process of image searching and retrieval, so in essence the retrieval process is metadata-based, but this metadata is extracted/built automatically and is non-textual as it contains simplified content-data from the image like color, shape, edge etc., so this makes the retrieval process a content-based one.

The paper is structured as follows. Section 2 describes the idea of our image search technique, i.e. combining metadata-based with content-based image retrieval in order to obtain an image searching technique more powerful than metadata-based image retrieval, but at the same time, less computationally expensive than content-based image retrieval. Then Section 3 presents our search algorithm, followed by Section 4 which presents the methods used for image feature extractions. Section 5 presents metrics used for comparing image features and the paper ends with evaluation tests in Section 6 and conclusions.

2. Combining Metadata-based and Content-based Image Retrieval

As we already mentioned, we developed in this paper an image searching technique which combines content-based image retrieval and metadata-based image retrieval. The main idea of our image searching technique is to use a metadata database in the search process to speed up the process, so it is primarily a metadata-based search, but this metadata is non-textual, it refers to the color content of the image (e.g. color histogram, shape, edge), so in this sense the searching technique has the power of a content-based searching technique. Also, in order to reduce the computing time of the search process, this metadata is automatically extracted when an image is added to the search

set and stored in a database, so it is done separately from the searching process itself and does not add to the search time.

By combining these two approaches to image searching and retrieval, the search process will gain efficiency (i.e. lower response time) because it uses metadata for searching and does not compare images pixel by pixel, and will also gain power (due to the fact that metadata is not textual, but in the form of features extracted from the image content itself).

3. The Search Algorithm

Our image searching technique uses a relational database for storing the binary metadata of each image from the search set. The schema of this database (in fact a table from a database) should contain for each image the following type of information:

- *Average colors* - average red, average blue and average green across the image matrix pixels;
- *Color Histogram* - the vector of the color histogram of the image;
- *Edge points* - the vector of coordinates of edge points from the image; We have used the following first-order edge operators: Roberts, Prewitt, Canny.

We have considered these three types of image content features because we want the search process to be flexible. If we want the search to be efficient (i.e. low computing time), then comparing average colors of images would be the best choice. But the search would not be so accurate. In order to increase the accuracy of the search color histogram comparisons can be used. Note that this would also increase the computing time. In order to be even more accurate comparing the edge profile of images is the best choice. In section 5 several metrics for comparing image features are presented.

The search algorithm is formed by two functions: the search itself and the function for building metadata of a newly added image.

The search algorithm is no different than a common image search algorithm:

Input: $queryIMG$ - the image (or a part of an image) which is searched for in the image search set
$searchSet$ - the set of available images where $queryIMG$ is searched for

BEGIN
$meta = metadata(queryIMG)$;
for img in $searchSet$ do

```
        meta1 = select_metadata(img);
        if compare(meta, meta1) < threshold
                echo "image found!", img;
        endif;
endfor;
END
```

For the search operation, first the metadata is extracted from the query image (i.e. using the *metadata*() function and then this metadata is compared against the metadata (retrieved from the database using function *select_metadata*()) of each image from the search set. The metadata retrieved from the database was pre-computed when the respective image was added to the search set. The *compare*() function compares various features from the metadata (average colors or histograms or edge profile). When a new image is added to the search set, the following is executed:

```
BEGIN
meta = metadata(newIMG);
add_metadata(meta);
searchSet = searchSet + newIMG;
END
```

More precisely, the metadata of the new image is automatically extracted from the image (using function *metadata*()) and saved into the metadata database (using function *add_metadata*()) and the image is then added to the search set.

4. Feature Extraction Operators

We have considered three types of image features: average colors, color histograms and edges.

The average red, green and blue are computed across the image's matrix pixel.

Then, the color histogram is computed as follows. First the luminance of each pixel is computed from the red (R), green (G) and blue (B) components of the pixel using the following formula:

$$Y = 0.299 * R + 0.587 * G + 0.114 * B$$

and then for each level of luminosity (from 0 -black to 255 -white) the frequency of occurrence of that luminosity is computed across the picture. The vector

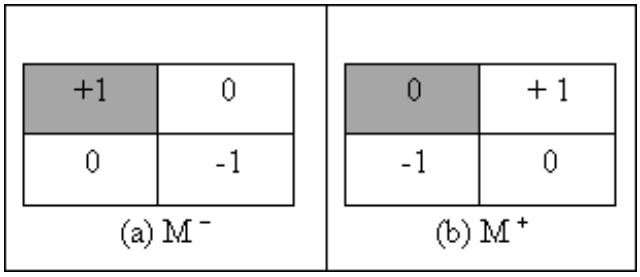


FIGURE 1. The Roberts operator mask

of 256 levels of luminance which contains the frequency of occurrence of each luminance level in the picture is the histogram vector for the picture.

For computing edges, we considered the following first-order edge detection operators: Roberts, Prewitt, Canny. The Roberts edge operator convolves the mask from Fig.1 with the image's pixels. The value of the edge in the pixel with coordinates (x, y) is computed as follows [3]:

$$E_{x,y} = max\{|M^+ * P_{x,y}|, |M^- * P_{x,y}|\},$$

where $x, y \in 1..n$ and $P_{x,y}$ is the luminance value of the pixel with coordinates (x, y).

Prewitt operator uses the mask from Fig. 2 in order to detect edge points. After this mask is convolved with the image's pixels, the Prewitt operator computes the edge magnitude, M, and the edge direction, θ, using the following formula [2]:

$$M(x,y) = \sqrt{Mx(x,y)^2 + My(x,y)^2}$$
$$\theta(x,y) = tan^{-1}\left(\frac{My(x,y)}{Mx(x,y)}\right)$$

The Canny edge detection operator is an improved multi-stage algorithm which has the following steps [1, 6]:

- *noise reduction*: the image is convolved with the first derivative of a Gaussian; results a blurred version of the original image;
- *compute the edge gradient and direction using the Sobel operator* [4]; the direction angle is rounded to 0, 45, 90 or 135;
- *non-maximum suppression*: determine if the gradient magnitude assumes a local maximum in the gradient direction; "thin edges" are obtained;
- *threshold with hysteresis to connect edge points*;

We can see in Fig. 3 that the Canny edge operator is more accurate than Prewitt and Roberts and produces less edge points.

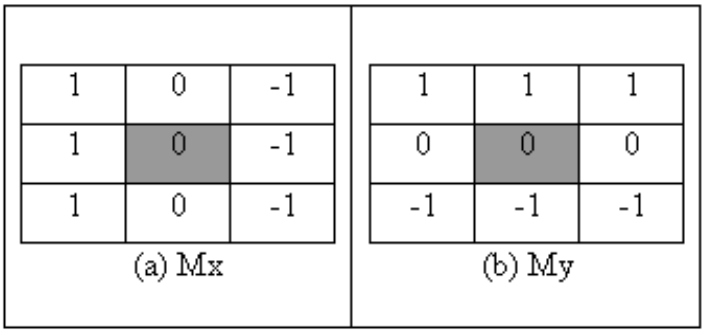


FIGURE 2. The Prewitt operator mask

5. Metrics for Comparing Image Features

We introduce in this section the metrics used to compare average colors, histogram and edge profiles (i.e. the *compare*() function from the search algorithm. When comparing two average colors (red, green or blue) their absolute difference is used as metric. When comparing two luminance histogram, we use the following metric, where $H1$ is the first histogram vector and $H2$ is the second histogram vector:

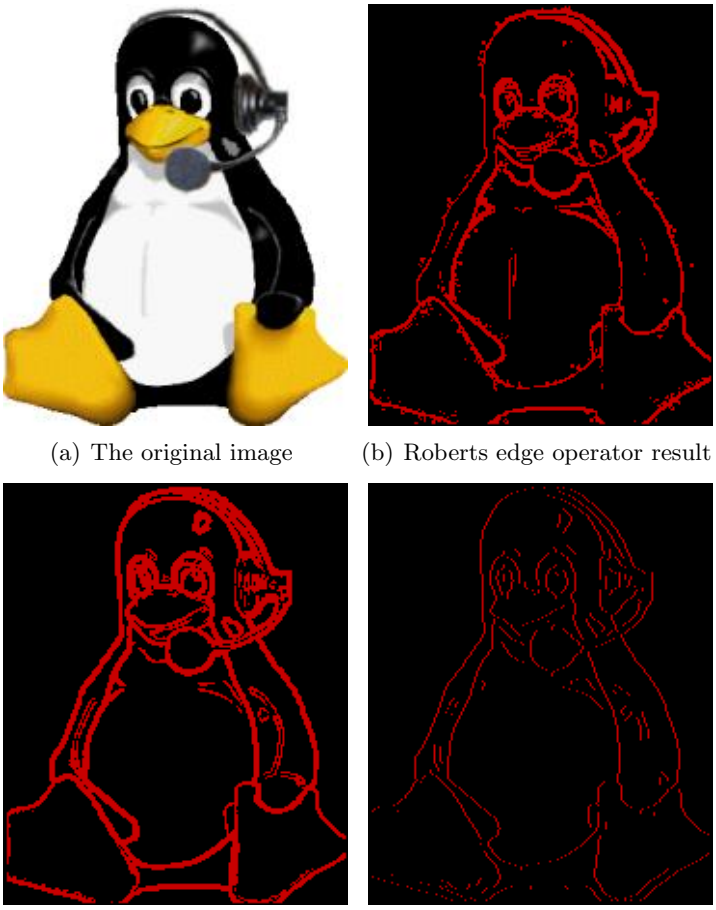
$$distance = \frac{\sum_{i=0}^{255} min(H1[i], H2[i])}{\sum_{i=0}^{255} H2[i]}$$

When we compare two edge profile (where an edge profile of an image is a vector of components (x, y, θ) where x and y are the coordinates of an edge point and θ is the direction of the edge) we use the following two metrics:

$$distance1 = \frac{1}{n}\sum_{i=1}^{n}(X_1[i] + Y_1[i]) - \frac{1}{m}\sum_{j=1}^{m}(X_2[j] + Y_2[j])$$

$$distance2 = \frac{1}{n}\sum_{i=1}^{n}\theta_1[i] - \frac{1}{m}\sum_{j=1}^{m}\theta_2[j]$$

The first metric compares positions of edge points ($(X_1[i], Y_1[i])$ is the position of an edge point in the first edge profile and $(X_1[j], Y_1[j])$ is the position of an edge point in the second edge profile) and the second one compares directions of the edge ($\theta_1[i]$ is the direction of the i-th edge point in the first edge profile). n and m are the number of edge points of the first edge profile and the second edge profile respectively. For the second metric each $\theta[i]$ is scaled



(a) The original image (b) Roberts edge operator result

(c) Prewitt edge operator result (d) Canny edge operator result

FIGURE 3. Edge detection operators

(i.e. divided) to the maximum value of direction across the image, in order to be in the interval $[0, 1]$.

6. EVALUATION

We have implemented our searching technique as a PHP application [5] and evaluated the performance of the search on a search set of 10 images each having a resolution ranging from 179x218 to 376x394 pixels. Depending on the threshold we set, we were able to find in the search set the image we were looking for or similar images (when the threshold was higher). However, we

must say that setting the threshold to the right value was critical for our experiments and the setting we have used might not work on other image search sets. The images from the search set contain computer-generated graphical data (like the one from Fig. 3) and real-world data (e.g. animal pictures). We have performed several search operations and the average results are displayed in the following table.

Feature used	Percent of correct results	Percent of incorrect results	No. of average comparisons per image	No. of average pixel-by-pixel comparisons	Threshold used
Average Red, Green, Blue	10%	90%	3	63240	0.04
Histogram	14%	86%	256	63240	0.4
Roberts edge profile	10%	90%	9106	63240	0.04
Prewitt edge profile	66%	33%	9851	63240	0.04
Canny edge profile	66%	33%	1836	63240	0.04

Column 1 presents the image feature used for searching (i.e. for comparing two images). Column 2 presents the percent of the result images which are correct results (i.e. similar image to the query image). Column 3 presents the percent of the result images which are incorrect results (i.e. images that are not similar to the query image). Column 4 presents the average number of metadata component comparisons (i.e. the number of components from the metadata of an image; for example, when the Average Red, Green and Blue feature is used, the number of components from the metadata of an image is 3) between two images during the search operation, while column 5 presents the same data if pixel-by-pixel comparisons would have been used for comparing two images (i.e. average width*height of all images). The last column presents the threshold value used for detecting similarity between 2 images. During the edge profile search the second metric for comparing edge profiles was used.

Regarding the efficiency of the search operation, for the image displayed in Fig. 3 if pixel-by-pixel comparison is done, 179*218 = 39022 comparisons are performed. But if the Canny edge profile of this picture is used in comparisons, only 1452 comparisons are used. This is of course an increase in efficiency.

From the values of columns 4 and 5 from the table we can see that the most efficient search is done using the Average color feature, followed by Histogram, then Canny edge profile, Roberts edge profile and Prewitt edge profile. We can see from column 4 and 5 that all these feature comparisons based search are more efficient (regarding the number of individual comparisons) that if content-based searching was used and each pixel from an image is compared with each pixel from the other image. We can see from column 2 and 3 that Canny edge profile and Prewitt edge profile comparisons produce the best results.

7. Conclusions

We have presented an image searching technique which combines metadata-based with content-based image searching and retrieval. This technique has the advantage of both approaches, namely the speed/efficiency of metadata-based search and the power of content-based search. The main idea of our technique is to construct a metadata database for an efficient search and retrieval process, but this metadata is not textual, but extracted from the content of images. This technique is more efficient than content-based searching of images with pixel-by-pixel comparisons. We have seen from the test performed that our technique produces reasonable good results during the image search operation. As future work, new metrics for comparing edge profiles can be considered and also second-order edge detection operators can be considered, for better search results.

Acknowledgment

This work was supported by the national grant CNCSIS No. PD-444, 2010.

References

[1] Canny, J., *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8(6):679698, 1986.

[2] Prewitt, J. M. S. and Mendelsohn, M. L., *The Analysis of Cell Images*, Ann. N. Y. Acad. Sci., 128, pp. 10351053, 1966

[3] Roberts, L. G., *Machine Perception of Three-Dimensional Solids*, In: Optical and Electro-Optical Information Processing, MIT Press, Cambridge, MA, pp. 159197, 1965

[4] Sobel, I. E., *Camera Models and Machine Perception*, PhD Thesis, Stanford University, 1970

[5] Miron D., *Metadata pentru Cautarea Bazata pe Continut a Imaginilor*, Master thesis, Babes-Bolyai University, 2010.

[6] M. Nixon, A. Aguado: *Feature Extraction and Image Processing*, Second Edition, Academic Press, 2008.

Babes-Bolyai University, Faculty of Mathematics and Computer Science, 1 M. Kogalniceanu St., 400084 Cluj-Napoca, Romania
E-mail address: forest@cs.ubbcluj.ro

Fortech, 19 Meteor St., Cluj-Napoca, Romania
E-mail address: daniela2003_ro@yahoo.com

STUDIA UNIV. BABEŞ–BOLYAI, INFORMATICA, Volume **LV**, Number 4, 2010

ONTOLOGY ASSISTED FORMAL SPECIFICATION EXTRACTION FROM TEXT

ANDREEA-DIANA MIHIŞ

Abstract. In the field of knowledge processing, the ontologies are the most important mean. They make possible for the computer to understand better the natural language and to make judgments. In this paper, a method which use ontologies in the semi-automatic extraction of formal specifications from a natural language text is proposed.

Between ontologies and natural language exists an old connection. The first ontologies created in the computer science were written in natural language. But in the natural language form they were not very useful. So, a formal way of representing ontologies was preferred. Some early formal forms were Hyper Text Markup Language (HTML) and Unified Modeling Language (UML). Now, the ontologies are represented in Resource Description Framework (RDF) or Web Ontology Language (OWL)[16], and in these forms they have many applications in the Semantic Web. Also, some of the existing ontologies were obtained in an automatic or an semi-automatic way from natural language texts.

In the Semantic Web, the ontologies are used for the semantic information coding behind a Web page or in a Web page, and in this way they support information retrieval. But the majority of the existing ontologies are designed for a specific domain, and they are used in that specific domain. And, as they have proved their usefulness in those domains, whay not for the specification extraction from natural language texts?

When the clients and the developers meet, the client's requirements are written in natural language, since the natural language is commonly understood. But, unfortunately, the natural language is ambiguous. In the requirements phase, the developers and the clients understand the requirements in the same way. But this does not guaranty that they will recall the same things

Received by the editors: December 3, 2010.

2010 *Mathematics Subject Classification.* 68T30, 68T50.

1998 *CR Categories and Descriptors.* I.2.7 [**Computing Methodologies**]: Artificial Intelligence – *Natural Language Processing* D.2.1 [**Software**]: Software Engineering – *Requirements/Specifications*;

Key words and phrases. Requirements, Ontology, Fformal Specification.

later. Maybe they are not the only persons which will develop the software. Usually, especially for large scale or average scale projects, the development team is split in smaller teams, specialized in a particular phase from the development process: specification, coding, testing, and so on. Sometimes the team responsible with the requirement elicitation is so specialized, that it can even belong to a different firm. And the requirements elicitation techniques gathers a lot of natural language requirements, which must be analyzed and transformed in specifications, from which the most useful are formal specifications. The purpose of the requirement analysis phase is the development of the Specification Document, which is in fact the contract between client and developers, and in order to be unambiguous, formal specifications must be used[17].

A method of semi-automatic formal specification extraction from requirements written in natural language is proposed in this paper.

1. Ontology

An ontology represents a rigorous and exhaustive organization of some knowledge domain that is usually hierarchical and contains all the relevant entities and their relations [28]. It's an old field of study in philosophy, with greek origins. "Ontology (from the Greek $\ddot{o}\nu$, genitive $\ddot{o}\nu\tau o\varsigma$: "of being" (neuter participle of $\epsilon\tilde{\imath}\nu\alpha\iota$: "to be") and $\lambda o\gamma\acute{\imath}\alpha$, -logia: science, study, theory) is the philosophical study of the nature of being, existence or reality in general, as well as the basic categories of being and their relations. Traditionally listed as a part of the major branch of philosophy known as metaphysics, ontology deals with questions concerning what entities exist or can be said to exist, and how such entities can be grouped, related within a hierarchy, and subdivided according to similarities and differences." [29].

In the field of computer science, the ontology is a relatively new field of study, but promising. As in philosophy, it intends to organize the knowledge from the web in a semantic form. And because the web is used as a support to publish information from different domains, also, the ontologies differ, some being domain specific, some being upper level ontologies [15]. The first ontologies used in the computer science domain were defined using natural language. These kinds of ontologies were called informal ontologies. But the most formal ones have the most applicability [5, 2]. Today, the backbone of the Semantic Web is consisted by the OWL (Web Ontology Language) and RDF (Resource Description Framework) [16].

A lot of ontologies were developed, by humans, as the work of different specialists or volunteers [20], to semi-automatic [13] and automatic ways [4]. Ontologies also differ in respect to the scope and purpose of their content.

The most prominent distinction is between the domain ontologies describing specific fields of endeavour, like economics or biology, and upper level ontologies describing the basic concepts and relationships invoked when information about any domain is expressed in natural language [22]. Usually the domain ontology is subordinated to the upper level ontology [15] (see Figure 1).

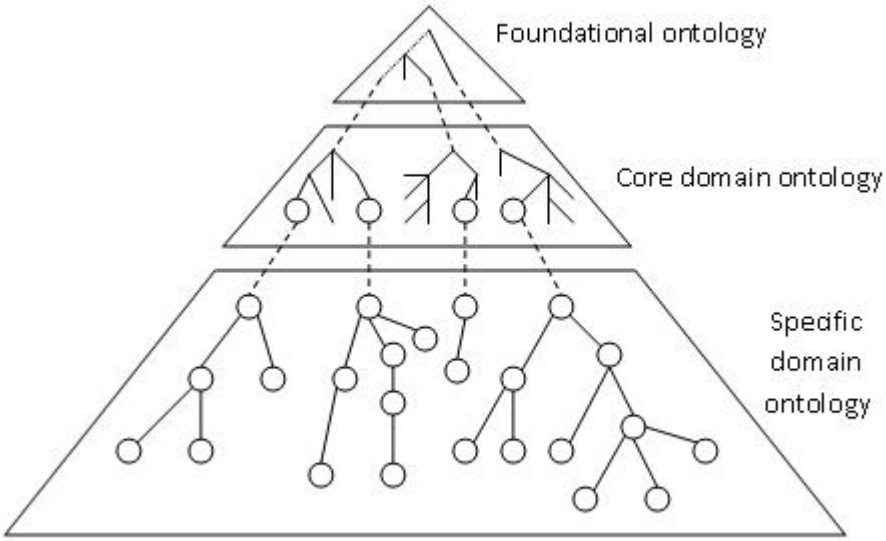


FIGURE 1. The tree levels of generality in a domain ontology [15]

From the representation point of view, one type of ontology is the taxonomy. A taxonomy is a hierarchical classification of terms. The relation between two terms, a parent and a child, is usually an "is a" relation. The terms can also be characterized by a set of properties.

But the most general way to define an ontology is by using triplets [3]. A triplet is composed from a subject or the concept, the predicate, which is a directional relation and the object, which is a characteristic. The relation and the characteristic can also be concepts. Any type of ontology can be represented in the triplet form, even if the ontology is a taxonomy or it has an integrate graph type structure.

2. Related work

Since the ontologies make the computer capable to better understand the natural language, they have a great applicability in the field of information

extraction. Since 2003, their applicability for information extraction was emphasized in [14], where the authors propose a step by step method for enriching an existing ontology in order to make it more appropriate for information extraction from a new text source.

In a newer paper [21], an ontology unsupervised method for information extraction is presented, without using any other knowledge sources. However, the precision of the results will depend on the quality of the input ontology. The method also identifies unused elements from the ontology, and in this way the quality of the ontology can be improved, and also the results of the extraction.

3. The semi-automatic formal specification extraction

In a previous paper [10], was proposed an application which assist the Stepwise Refinement process. In that application, the Stepwise Refinement process started with an abstract program [6], and the goal of the refinement was to obtain code. In another paper [11], another application which assist the Z schema usage, and also transform a Z schema into an abstract program was presented. The goal of this paper is to extract an abstract program from a natural language text which represent a program requirement. In other words, the goal is to obtain the specifications, i.e. the precondition, the postcondition and the variable list from natural language requirements. In natural language, the sentences which correspond to the precondition respective postcondition can be selected by the tense of their predicates. And the variables involved are usually the subjects of those sentences.

There are two types of requirements: requirements which are expressed by many sentences, and many requirements expressed in a single sentence. In the first case, the identification of the preconditions and postconditions seems to be much easier to be resolved, since it can be reduced to a sentence selection problem. In the second case, a single sentence must be split into one or more preconditions and one or more postconditions. In the second case, the sentence itself must be analyzed.

In the field of natural language processing, the accuracy of some natural language processing methods has improved. Some still have an accuracy less then 70%, such as the text entailment relationship [1]. But there are a lot of tools capable of identifying the correct part of speech of words from sentences, with an accuracy more than 95% [18]. In the field of grammatical analysis of a sentence also a lot of work was done, and there are available free tools capable to analyse from the syntactic point of view a sentence, such as the online tool developed by the Stanford Natural Language Processing Group [26], with an

accuracy of more than 87% [8], tool which is updated continuously [27]. The result of the grammatical analysis of a sentence is usually represented as a tree. Or, between the words of a sentence, dependence relations can be identified [9], relations which are astonishing similar to the RDF triplets, the simplest way to represent an ontology.

3.1. **Sentence Analysis.** In a previous paper [12], the output of the Stanford Parser was used to extract ontology triples, in order to identify the best ontology which matches a natural language text. In this paper, Stanford Parser is used again, and the dependence relations will be used as ontology triples concept-predicate-object, where the concept is the first word from the dependency, the predicate is the dependency, and the object is the second word from the dependency.

So, for instance for the following requirement: *"Generate the first prime number larger than a given natural number n."*[23], the syntactic analysis tree can be seen in the figure 2, the dependencies list in Figure 3 and the associated graph for the dependencies in Figure 4.

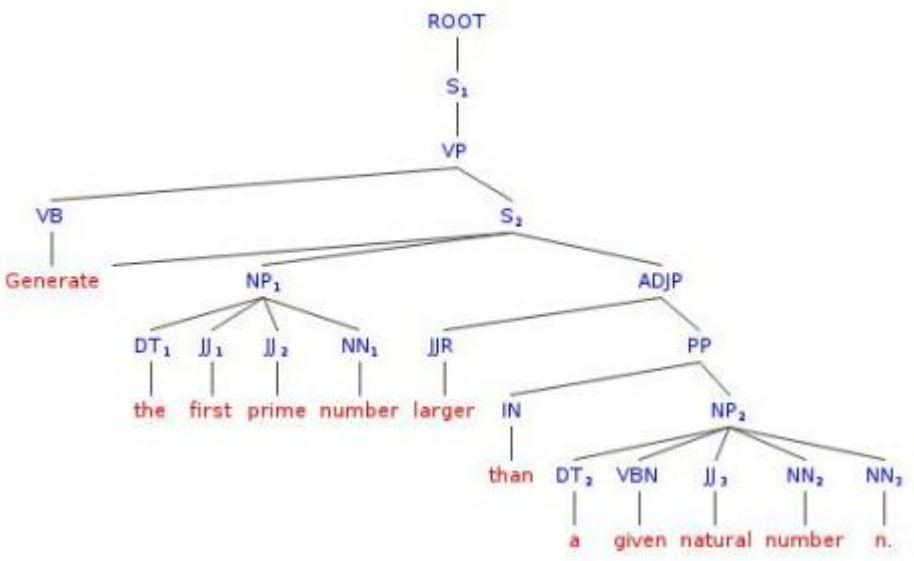


FIGURE 2. The syntactic analysis tree

From the dependencies graph, it can be observed that the sentence has two main parts: Generate + larger + number (the subject) + the + first + prime, and larger + n. + a + given + number + natural. In the first list, Generate is a VB (verb, base form: imperative, infinitive or subjunctive [25]),

det(number-5, the-2)
amod(number-5, first-3)
amod(number-5, prime-4)
nsubj(larger-6, number-5)
xcomp(Generate-1, larger-6)
det(n.-12, a-8)
amod(n.-12, given-9)
amod(n.-12, natural-10)
nn(n.-12, number-11)
prep_than(larger-6, n.-12)

FIGURE 3. The dependencies list

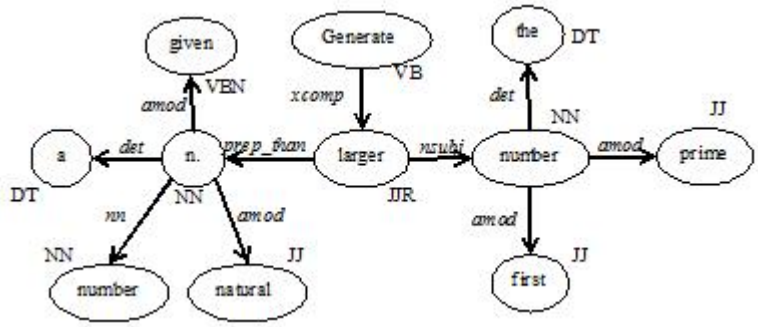


FIGURE 4. The dependencies graph

and is an imperative verb, so this list will represent the postcondition. In the second list, given is a VBN (verb, past participle [25]) and is in the past tense, so the second list will be the precondition. The two nouns, "n" and "number" will represent the variables.

The same conclusion can be reached by using the syntactic analysis tree. "the first prime number" is a NP (noun phrase [24]), and "a given number n" is another NP, subordinate to the first one. Because the verb is an imperative one, the first NP will represent the requirement, and with the first NP, will compose the postcondition.

In the case in which more sentences represent the requirements, such as in the case of: "The sequence a1, ..., an with distinct integer numbers is given. Determine all subsets of elements with sum divisible to n."[23], is easy to select the sentences by their tense. In the first sentence "given" is a VBN, and in the second "Determine" is a VB. SO, the first sentence will represent the

precondition, and the last the postcondition. Between the two sentences is a connection, but it is not an evident one. The subject of the first sentence, "the sequence" appears in the second, but not directly. There is a connection between the object from the second sentence, "subsets" and "the sequence", but the connection is a logical one, and in order to be recognized by the computer, new knowledge must be available. The most profitable is the existence of an ontology, where sequence is similar with set, and subset is a subordinate of the set, or one in which subset and sequence are related directly. If no such information is available, then the similarity of the two words will provide a hint for their relationship. The similarity may be computed between the two sentences or text to which the two words belong, or, between their glosses from a dictionary, if the words were disambiguated. If this kind of connections can be made, then the sentences graphs can be merged, and considered as a big graph as input for the following subalgorithm:

```
Subalgorithm PrePostSelection(g, pre_list, post_list)
    DATA: g - a dependencies directional graph
    @Identify the VB words list: VB_list
    @Initialize a list of lists of words, Word_ll and place in
     every list on the first position the VB word
    @Initialize an empty list of distinct words, Var_l
    For @every list L from Word_ll do
        @Identify the graph path which ends with a dependency
         "sub" for a VB or a VBD, and if none exist, the path
         which ends with a dependency "obj", respective the path
         which ends with an "amod" dependency for a VBN
        @Add all the words from the path to L
        @The NN word which is in relation "sub", "obj" or "amod"
         must be added to the variable list Var_l
        @Add all to L all the words connected to the last word
         in a recursive way (they and all the words connected to
         them)
    endfor
    RESULTS: pre_list - the lists L from Word_ll which starts
                        with a VB
             post_list - the other lists from Word_ll
             Var_l - the list of variables
endSub
```

In the case in which the requirements text contains many sentences, the proposed algorithm can be applied after a ontology alignment operation. From every sentence, the dependency graph can be constructed, and this graph is a

mini-ontology (it contains entities and relations). This is the way an ontology alignment process can be used to merge all the sentences graph into a larger graph.

For an even larger requirement text, the text must be segmented first, then for every segment, the sentences dependency graph merged, and then the proposed algorithm can be applied.

3.2. **Ontology assisted.** For the further refinement from natural language preconditions/postconditions into abstract programs, or Z schemas, is necessary the use of an ontology, which, for instance for the case of Z schemas, has as base-nodes the base types and the base operations for these types. For instance, from the first example, *"Generate the first prime number larger than a given natural number n."*[23], it can be noticed that, the basic type natural number will be used, with prime and larger than as operations, and that only first is not a simple predicate involving a natural number.

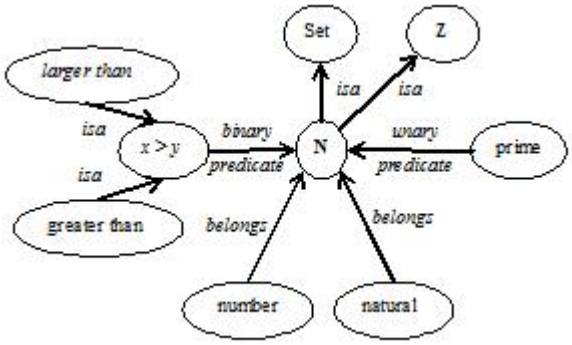


FIGURE 5. A part of Natural numbers ontology

In this case, first words which represent the variables will be searched in the ontology (see Figure 5), and replaced with the closest base type (after the node "natural" is identified, the closest base type is **N**). Then, from the natural language preconditions and postconditions all the words are searched between the sub-ontology with the base root the base type identified (If the variable list is formed from many variables, all with different types, a union of sub-ontology will be used). If they are found, then the base predicate directly connected to the base type will replace the current word.

In the discussed example, *"Generate the first prime number larger than a given natural number n."*[23], the identified sentence which express the post-condition is Generate + larger + number (the subject) + the + first + prime.

Number can be replaced by $x \in \mathbf{N}$, and also larger $x > n$. The computer can provide these base types/predicates, and in this way to assist the translation process from natural language into formal specifications.

4. Experiments

The proposed algorithm was implemented in C++ and tested on the laboratory requirements from Fundamentals of Programming [23]. It worked well with natural language requirements. Because these formulas are not natural language sentences, the Stanford parser cannot identify any connection between the elements of the formula. So, these elements will be disconnected from the other words from the dependencies graph. As the Stanford parser output is the current algorithm input, they were ignored further, and the quality of the results suffered. Unfortunately a more appropriate set of test was not found. A solution to overcome this problem is to replace these formulas with different symbols, and only then to apply the algorithm.

5. Conclusions and Future work

In this paper a method for semi-automatic formal specification extraction from a natural language text was presented. It uses the Stanford Parser to obtain the dependency graph. The dependency graph is seen as a mini-ontology of the sentence. Then ontology alignment process is used to merge the mini-ontologies. In the end, using semantic principles, natural language sentences which represents the preconditions and postconditions are extracted.

In the future, I wish to construct manually an ontology for the Z language base types, and using it, to further refine the identified natural language preconditions and post-conditions into formal specifications.

6. Acknowledgements

This work was supported by ANCS-CNMP, project number PNII - 91037 /2007.

References

[1] Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B. and Szpektor, I., *The Second PASCAL Recognising Textual Entailment Challenge*, Venice, 2006.

[2] Benett, B. and Fellbaum, C., *Formal Ontology in Information Systems*, Formal ontology in information systems: proceedings of the fourth international conference, IOS Press, Amsterdam, 2006.

[3] Brachman, R. and Schmolze, J., *An Overview of the KL-ONE Knowledge Representation System*, Cognitive Science, vol. 9, no. 2, 1985, pp. 171-216, http://nlp.shef.ac.uk/kr/papers/klone.ps

[4] Dahab, M., Hassan, H. and Rafea, A., *TextOntoEx: Automatic Ontology Construction from Natural English Text*, International Conference on Artificial Intelligence and Machine Learning (AIML-06), Sharm El Sheikh, Egypt, June 13-15 2006, http://www.icgst.com/con06/aiml06/Final_Articles/P1120615104.pdf

[5] Davies, J., Fensel, D. and van Harmelen, F., *Towards the Semantic Web: Ontology-driven Knowledge Management*, John Wiley & Sons Ltd, 2002, New York.

[6] Dijkstra, E.W., *Guarded commands, nondeterminacy and formal derivation of programs*, Comm. ACM, Vol. 18(1975), Nu. 8, pp. 453-457.

[7] Euzenat, J. and Shvaiko, P., *Ontology Matching*, Springer, 2007.

[8] Klein, D. and Manning, C. D., *Fast Exact Inference with a Factored Model for Natural Language Parsing*, in Advances in Neural Information Processing Systems 15 (NIPS 2002), Cambridge, MA: MIT Press, pp. 3-10.

[9] de Marneffe, M.-C. and Manning, C. D., *The Stanford typed dependencies representation*, in COLING Workshop on Cross-framework and Cross-domain Parser Evaluation, Manchester, United Kingdom, 2008, http://nlp.stanford.edu/pubs/dependencies-coling08.pdf.

[10] Mihiş, A.D., *An Application That Assist StepWise Refinement*, Proceedings of Symposium "Zilele Academice Clujene", 2006, pp. 143-147.

[11] Mihiş, A.D., *A Tool for Refinenent of Z Schemas*, Proceedings of Symposium "Zilele Academice Clujene", 2010, pp. 64-67.

[12] Mihiş, A.D., *Ontology Learning from Text Based on the Syntactic Analysis Tree of a Sentence*, The 5th International Conference on Virtual Learning (ICVL - 2010), Târgu Mureş, Edituara Universităţii Bucureşti, Editor: Vlada Marian, Albeanu Grigore, Popovici Dorin Mircea, ISSN: 1844-8933, 1842-4708, http://c3.icvl.eu/2010, 2010, pp. 128-134, Intel®Special Award winning article.

[13] Morita, T., Fukuta, N., Izumi, N. and Yamaguchi, T., *DODDLE-OWL: A Domain Ontology Construction Tool with OWL*, Proceedings of the 1st Asian Semantic Web Conference, Lecture Notes in Computer Science, vol. 4185, Beijing, China, 2006, pp. 537-551, http://iws.seu.edu.cn/resource/Proceedings/ASWC/2006/papers/4185/41850537.pdf

[14] Maedche, A., Neumann1, G. and Staab, S., *Bootstrapping an Ontology-based information extraction system*, in Intelligent exploration of the web, Editors: Piotr S. Szczepaniak, Javier Segovia, Janusz Kacprzyk, Lotfi A. Zadeh, Physica-Verlag GmbH Heidelberg, Germany, 2003, pp. 345 - 359.

[15] Navigli, R., Velardi, P. and Gangemi, A., *Ontology Learning and Its Application to Automated Terminology Translation*, IEEE Intelligent Systems, vol. 18, no. 1, January 2003, pp. 22-31, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.79.1758&rep=rep1&type=pdf

[16] Pollock, J. T. *Semantic Web for Dummies*, Wiley Publishing, 2009, Indianapolis, Indiana.

[17] Pressman, R. S., *Software Engineering - A Practitioners Approach*, McGraw Hill, third edition, 1992.

[18] Toutanova, K., Klein, D., Manning, C. D. and Singer, Y., *Feature-Rich Part-of-Speech Tagging with a Cyclic Dependency Network*, in Proceedings of HLT-NAACL 2003, Edmonton, Canada, pp. 252-259.

[19] Walther, M., Hhne, L., Schuster, D. and Schill, A., *Locating and Extracting Product Specifications from Producer Websites*, 12th International Conference on Enterprise Information Systems (ICEIS), Funchal, Madeira, Portugal, 2010.

[20] Wilkinson, M. and Good, B., *Construction and Evaluation of OWL-DL Ontologies*, Microsoft Research Faculty Summit, Redmond, USA, July 17-18 2006, http://research.microsoft.com/en-us/um/redmond/events/fs2006/agenda_mon.aspx
[21] Yildiz, B. and Miksch, S., *ontoX - a method for Ontology-driven information extraction*, Proceedings of the 2007 international conference on Computational science and its applications - Volume Part III, Workshop on CAD/CAM and web based collaboration (CADCAM 07), Springer-Verlag Berlin, Heidelberg 2007, pp. 660-673.
[22] Zervas D., *WS-TALK - Automatic Ontology Construction*, http://thames.cs.rhul.ac.uk/ wstalk/papers.html, 2005.
[23] http://www.cs.ubbcluj.ro/ adriana/Teaching.html
[24] http://grammar.about.com/od/mo/g/nounphraseterm.htm
[25] http://bioie.ldc.upenn.edu/wiki/index.php/POS_tags
[26] The Stanford Natural Language Processing Group, *Stanford Parser*, http://nlp.stanford.edu:8080/parser/
[27] The Stanford Natural Language Processing Group, *The Stanford Parser: A statistical parser*, http://nlp.stanford.edu/software/lex-parser.shtml
[28] http://wordnetweb.princeton.edu/perl/webwn?s=ontology
[29] http://en.wikipedia.org/wiki/Ontology
[30] http://www.wordnet-online.com/business.shtml

Babeş-Bolyai University, Faculty of Mathematics and Computer Science, Department of Computer Science, 1 M. Kogălniceanu St., 400084, Cluj-Napoca, Romania

E-mail address: mihis@cs.ubbcluj.ro