



I T H E A



International Journal

**INFORMATION THEORIES
&
APPLICATIONS**



2009 Volume 16 Number 4



International Journal
INFORMATION THEORIES & APPLICATIONS
Volume 16 / 2009, Number 4

Editor in chief: **Krassimir Markov** (Bulgaria)

International Editorial Staff

Chairman: **Victor Gladun** (Ukraine)

Adil Timofeev	(Russia)	Ilia Mitov	(Bulgaria)
Aleksey Voloshin	(Ukraine)	Juan Castellanos	(Spain)
Alexander Eremeev	(Russia)	Koen Vanhoof	(Belgium)
Alexander Kleshchev	(Russia)	Levon Aslanyan	(Armenia)
Alexander Palagin	(Ukraine)	Luis F. de Mingo	(Spain)
Alfredo Milani	(Italy)	Nikolay Zagoruiko	(Russia)
Anatoliy Krissilov	(Ukraine)	Peter Stanchev	(Bulgaria)
Anatoliy Shevchenko	(Ukraine)	Rumyana Kirkova	(Bulgaria)
Arkadij Zakrevskij	(Belarus)	Stefan Dodunekov	(Bulgaria)
Avram Eskenazi	(Bulgaria)	Tatyana Gavrilova	(Russia)
Boris Fedunov	(Russia)	Vasil Sgurev	(Bulgaria)
Constantine Gaiandric	(Moldavia)	Vitaliy Lozovskiy	(Ukraine)
Eugenia Velikova-Bandova	(Bulgaria)	Vitaliy Velichko	(Ukraine)
Galina Rybina	(Russia)	Vladimir Donchenko	(Ukraine)
Gennady Lbov	(Russia)	Vladimir Jotsov	(Bulgaria)
Georgi Gluhchev	(Bulgaria)	Vladimir Lovitskii	(GB)

IJ ITA is official publisher of the scientific papers of the members of
the ITHEA® International Scientific Society

IJ ITA welcomes scientific papers connected with any information theory or its application.

IJ ITA rules for preparing the manuscripts are compulsory.

The rules for the papers for IJ ITA as well as the subscription fees are given on www.ithea.org.

The camera-ready copy of the paper should be received by <http://ij.ithea.org>.

Responsibility for papers published in IJ ITA belongs to authors.

General Sponsor of IJ ITA is the Consortium FOI Bulgaria (www.foibg.com).

International Journal "INFORMATION THEORIES & APPLICATIONS" Vol.16, Number 4, 2009

Printed in Bulgaria

Edited by the Institute of Information Theories and Applications FOI ITHEA®, Bulgaria,
in collaboration with the V.M.Glushkov Institute of Cybernetics of NAS, Ukraine,
and the Institute of Mathematics and Informatics, BAS, Bulgaria.

Publisher: ITHEA®

Sofia, 1000, P.O.B. 775, Bulgaria. www.ithea.org, e-mail: info@foibg.com

Copyright © 1993-2009 All rights reserved for the publisher and all authors.

© 1993-2009 "Information Theories and Applications" is a trademark of Krassimir Markov

ISSN 1310-0513 (printed)

ISSN 1313-0463 (online)

ISSN 1313-0498 (CD/DVD)

CLASSIFICATION OF HEURISTIC METHODS IN COMBINATORIAL OPTIMIZATION

Sergii Sirenko

Abstract: *An important for the scientific as well as the industrial world is the field of combinatorial optimization. These problems arise in many areas of computer science and other disciplines in which computational methods are applied, such as artificial intelligence, operation research, bioinformatics and electronic commerce. Many of combinatorial optimization problems are NP-hard and in this field heuristics often are the only way to solve the problem efficiently, despite the fact that the heuristics represent a class of methods for which in general there is no formal theoretical justification of their performance. A lot of heuristic methods possessing different qualities and characteristics for combinatorial optimization problems were introduced. One of the approaches to the description and analysis of these methods is classification. In the paper a number of different characteristics for which it is possible to classify the heuristics for solving combinatorial optimization problems are proposed. The suggested classification is an extension of the previous work in the area. This work generalizes existing approaches to the heuristics' classification and provides formal definitions for the algorithms' characteristics on which the classes are based. The classification describes heuristic methods from different viewpoints. Among main considered aspects is decision making approach, structure complexity, solution spaces utilized, memory presence, trajectory-continuity, search landscape modification, and adaptation presence.*

Keywords: *combinatorial optimization, classification of methods, heuristics, metaheuristics.*

ACM Classification Keywords: *G.1.6 [Numerical Analysis] Optimization, I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search – Heuristic methods, General Terms: Algorithms.*

Introduction

Development of the combinatorial optimization (CO) field has reached the level of generalizing the accumulated primary knowledge. This includes devoting more attention to the study of existing approaches, to the analysis of their similarities, differences, and conceptual features enabling performance increase. The similarities can be captured through formulation of generalized search procedures, specifying components of which one can outline distinct classes of algorithms. The distinguishing of key differences can be done through a classification.

Heuristics represent a class of methods for which in general there is no formal proof of their performance/completeness. We will under "heuristic" only approximate methods, though a number of exact or approximation methods are also based on some "heuristic rules". But many of the practically relevant CO problems are NP-hard [Garey and Johnson, 1979; Papadimitriou and Steiglitz, 1982; Korte and Vygen, 2006] and the best exact algorithms known so far for these problems have exponential time complexity. In this situation approximate algorithms and, in particular, heuristics may be the only way to get solutions of a "good" quality in a reasonable amount of time.

Work concerning classification of heuristics in CO and their subclasses include [Stützle, 1998; Vaessens et al., 1998; Birattari et al., 2001; Blum and Roli, 2003; Sergienko and Shilo, 2003; Leont'ev, 2007; Talbi, 2009]. In this paper known approaches to the heuristics classification are generalized and extended. The classification is based

on defining characteristics by which the heuristics can be classified. These characteristics can be divided into three categories: structure characteristics, search process characteristics and performance characteristics.

Algorithms for static CO are considered here, leaving dynamic, stochastic, and multiobjective problems out of the scope of the work. Parallel implementation issues are also beyond consideration.

The remainder of this paper is structured as follows. Next section states definitions and notations used. Then a proposed classification divided into categories is presented and discussed in detail. Last section concludes this paper and sets out some issues for future research.

Definitions and Notations

Objects (solutions) that are considered in the CO problems typically are integer numbers, assignments, permutations, orderings, or graphs. All of them are generalized by the following definition [Hulianyskyi and Sergienko, 2007].

Definition 1. Consider a set $Y = \{1, \dots, m\}$, an at most countable set Z called *base space*, and a homomorphism $\varphi: Y \rightarrow Z$ that satisfies constraints defined by some predicate Ω . A triple $\kappa = (\varphi, Z, \Omega)$ is called a *combinatorial object*. A size of combinatorial object is a power of the set Y .

For increased readability, the combinatorial object will be denoted simply by φ .

Using as a basis [Papadimitriou and Steiglitz, 1982] we will give the following definition of CO problem (without loss of generality we will consider minimization problems).

Definition 2. A *combinatorial optimization problem* is a problem of finding $x_* \in D \subseteq X$ such that

$$\forall x \in D \subseteq X \quad f(x_*) \leq f(x) \quad (1)$$

where X is a finite (or possible countably infinite) set of combinatorial objects, $D \subseteq X$ is a subspace of the feasible solutions, and $f: X \rightarrow \mathbb{R}^1$ is an *objective (cost) function* of the problem. x_* is called a *globally optimal solution* or simply an *optimal solution*.

In general, X may consist of the combinatorial objects of different size.

Many of the heuristics solve problems in a special representation that may differ from initial model. For example, in generic algorithms integer solutions can be coded as binary strings. An *evaluation function* (sometimes called a *fitness function*) that differs from the objective function is often introduced for guiding the search process. These choices are "algorithmic-oriented", because the representation is defined to tune the problem definition with the algorithm that will be used to solve the problem [Roli and Milano, 2001]. Further we will consider that the problem (1) is solved the representation (S, g) , where S is a set called a *search space*, which represents solution space, and $g: S \rightarrow \mathbb{R}^1$ is an evaluation function. Elements of S are called *candidate solutions*. The representation can be equal to the initial problem itself: $S \equiv X, g \equiv f$.

Another important notions are a neighbourhood structure and a local minimum [Papadimitriou and Steiglitz, 1982].

Definition 3. A *neighbourhood structure* is a mapping $N: S \rightarrow 2^S$ that assigns to every candidate solution $s \in S$ a set of neighbours $N(s) \subseteq S$. $N(s)$ is called a *neighbourhood* of s .

Definition 4. A candidate solution s is called a *locally minimal solution with respect to a neighbourhood structure N* (or simply a *local minimum*) if

$$\forall s' \in N(s) \ f(s) \leq f(s').$$

The set S and the neighbourhood structure N induces a *neighbourhood graph* $G_N = (S, V_N)$, $V_N = \{(s, s') \mid s' \in N(s)\}$.

We will also use a notion of a search landscape [Hoos and Stützle, 2005].

Definition 5. Given a problem representation (S, g) and a neighbourhood structure N , the *search landscape* of the combinatorial optimization problem is defined as (S, g, N) .

Classification of Heuristics in Combinatorial Optimization

In the last decades tens of different heuristic approaches possessing certain qualities and characteristics were introduced to solve a CO problems. Basically all computational approaches for solving hard CO problems can be characterised as search algorithms [Hoos and Stützle, 2005]. For marking out different features of these CO methods this classification is suggested. The fundamental idea behind the search approach is to iteratively generate and evaluate candidate solutions. In the context of the algorithms operating at each iteration with only one candidate solution generation and acceptance of neighbour candidate solution is called a *move*.

In [Stützle, 1998] metaheuristics were suggested to classify by whether they are trajectory or discontinuous, by the number of operated solutions, by memory usage, by the number of neighbourhood structures, by changes to the objective functions, and by source of inspiration (Table1). More formal classification of local search algorithms based on an abstract algorithmic skeleton was suggested in [Vaessens et al., 1998]: the algorithms are divided by the number of solution operated at a time and by the number of levels. Levels corresponds to neighbourhoods used. Paper [Birattari et al., 2001] repeats the ideas presented in [Stützle, 1998]. Work [Blum and Roli, 2003] suggests a similar approach. In [Talbi, 2009] another classification for metaheuristics was presented (Table 2).

Table 1. Classification [Stützle, 1998]

Characteristic
Trajectory methods vs. discontinuous methods
Population-based vs. single-point search
Memory usage vs. memoryless methods
One vs. various neighborhood structures
Dynamic vs. static objective function
Nature-inspired vs. non-nature inspiration

Table 2. Classification [Talbi, 2009]

Characteristic
Nature-inspired vs. non-nature inspiration
Memory usage vs. memoryless methods
Deterministic vs. stochastic
Population-based vs. single-point search
Iterative vs. greedy

Table 3 lists characteristics by which we propose to classify the heuristics. Structure characteristics reflect choices made during designing of the algorithm and they determine characteristics of the rest two categories. Search process characteristics describe possible implementation results of previous category characteristics. Next category represents results of the theoretical study of heuristics' performance: a priori and a posteriori

performance guarantees and convergence-related properties. On obtaining new theoretical results some algorithms may change class membership in this category.

Heuristics can also be classified by their original source of inspiration. Many of them are actually inspired by phenomena occurring in nature. But this characteristic does not reflect essential features of the algorithms and was not included in the presented classification.

Table 3. Characteristics by which the heuristics can be classified

Category	Characteristic
Structure characteristics	Decision making approach
	Structure complexity
	Solution spaces utilized
	Memory presence
Search process characteristics	Trajectory type
	Search landscape modification
	Adaptation/learning presence
	Problem model presence
Performance characteristics	Performance guaranties presence
	Convergence-related properties

It is hardly possible to state their place in the classification for all of the suggested algorithms so in the paper we discuss only the most prominent and exemplifying methods. Besides that many of the state-of-art CO algorithms are hybrid in some sense, so "standard" implementations of algorithms are mainly considered. Where possible both references to the early papers and to the recent surveys of methods/algorithms are provided. It also should be noted that this classification emphasizes differences between heuristics, but don't revoke their similarities.

Structure Characteristics

Decision making approach. First of all we will distinguish different heuristics by whether they are *stochastic* or *deterministic* (Table 4), that is whether or not decision making (e.g., generating or selecting of the candidate solution from a neighbourhood) is randomized in the algorithm. In deterministic algorithms, using the same initial solution will lead to the same final solution, whereas in stochastic metaheuristics, different final solutions may be obtained from the same initial solution. Simple Greedy Algorithms [Khuller et al., 2007] and standard Local Search [Sergienko, 1964; Papadimitriou and Steiglitz, 1982] are examples of deterministic methods. Greedy Algorithms sequentially add by some rule components to partial solution until a complete solution is constructed. Local Search starts from some initial complete solution and iteratively replaces it by better solution (in the sense of the objective/evaluation function) choosing it from the neighbourhood of a current solution. Simulated Annealing [Kirkpatrick et al., 1983; Aarts et al., 2007] is a well known stochastic algorithm, which extends Local Search by enabling moves to worse solutions. Candidate solutions are chosen according to the parametric probabilistic rule that depends on a solution quality.

Many of the state of the art heuristics are stochastic [Handbook of Applied Optimization, 2002; Hoos and Stützle, 2005; Handbook of Approximation Algorithms and Metaheuristics, 2007; Talbi, 2009].

Table 4. Classification by decision making approach

Class	Examples
Deterministic algorithms	<i>Local Search</i> [Sergienko, 1964; Papadimitriou and Steiglitz, 1982] ...
Stochastic algorithms	<i>Simulated annealing</i> [Kirkpatrick et al., 1983; Aarts et al., 2007], ...

Structure complexity. Classifying heuristics by complexity of a structure, we distinguish simple algorithms (heuristics), hybrid algorithms (hybrid heuristics), metaheuristics, hybrid metaheuristics, and hyperheuristics (Table 5).

A *metaheuristic* shortly can be defined as generic technique or approach that is used to guide or control the underlying problem-specific heuristic method in order to improve its performance or robustness [Hoos and Stützle, 2005]. The term metaheuristic was first introduced in [Glover, 1986]. The metaheuristics are general optimization methods that can be adopted to many problems or problem classes through defining problem-specific components. This usually requires less work than developing a specialized heuristic from scratch.

A *hybrid heuristic* is such combination of two or more simple algorithms (heuristics) that do not induce a metaheuristic. For example, it can be sequential or parallel execution of the algorithms [Zhuravlev, 1977; Bertsekas et al., 1997]. *Hybrid metaheuristics* are methods that combine components from two or more metaheuristics. Metaheuristics can also be hybridized with exact methods, specialized techniques or other operation research procedures. Also we will regard as hybrid metaheuristics a cooperative multiagent procedures, where agents are individual methods (metaheuristics, exact or specialized algorithms etc.). Classification of hybrid metaheuristics is presented and discussed in detail in [Raidl, 2006].

Hyperheuristics is a new developing class of methods aimed to solve wider classes of various problems without been special tuned for each problem or problem subclass, as, for example, is in the case of metaheuristics. Despite the significant progress in building search methodologies for a wide variety of application areas so far, they still require specialists to integrate their expertise in a given problem domain. For detailed description of hyperheuristic approaches see [Burke et al., 2003; Ozcan et al., 2008].

Table 5. Classification by structure complexity

Class	Examples
Simple algorithms (heuristics)	<i>Greedy Algorithms</i> [Khuller et al., 2007], ...
Hybrid algorithms (hybrid heuristics)	Sequential execution of heuristics [Zhuravlev, 1977; Bertsekas et al., 1997], ...
Metaheuristics	<i>Ant Colony Optimization</i> [Dorigo et al., 1991; Dorigo and Stützle, 2004], <i>Memetic Algorithms</i> [Moscato, 1989; Moscato and Cotta, 2007], ...
Hybrid metaheuristics	[Raidl, 2006]
Hyperheuristics	[Burke et al., 2003; Ozcan et al., 2008]

Though classification by the complexity of the structure is important, it is hardly possible to distinguish algorithms strictly by this characteristic due to the absence of generally accepted and formal definition of terms "metaheuristic" and "hyperheuristic". For example, Simulated Annealing some authors consider as metaheuristic and some as a simple algorithm, as its difference from Local Search is minor even if compared to such a "simple" metaheuristic as Iterated Local Search [Lourenço et al., 2002]. In Iterated Local Search method the subordinate Local Search routine is iteratively restarted from perturbed local optima obtained at previous iterations. This enables to overcome one of the major drawbacks of standard Local Search: stopping in a local optimum, which is not always a global optimum.

Solution spaces utilized. We will outline by the type of solution spaces utilized a class of *sequential (constructive, greedy)* algorithms. These methods generate (construct) candidate solution from scratch by sequentially adding solution components. Solution construction can be seen as a search process in the problem representation that is extended by partial solutions. The search stops when a complete (feasible) solution is reached. If we assume that all candidate solutions are of the same size m than this extended search space can be represented using notations of Definitions 1 and 2 as following

$$\begin{aligned} S &= \bigcup_{i=1, \dots, m} X_i, \\ X_i &= \{\varphi^i : Y_i \rightarrow Z, Y_i \subset Y, |Y_i| = i\}, i = 1, \dots, m-1, \\ X_m &= X, \end{aligned} \quad (2)$$

where φ^i is a combinatorial object that satisfies constraints defined by a predicate Ω^i and the following conditions hold: i) $\Omega^{i-1} \rightarrow \Omega^i$, $i = 2, \dots, m-1$, ii) $\Omega^{m-1} \rightarrow \Omega$.

For example, the Nearest Neighbour Heuristic for the Traveling Salesman Problem [Hoos and Stützle, 2005] sequentially adds to a current partial tour the shortest arc that don't make the tour infeasible.

On the contrary, *iterative (perturbative)* algorithms (Table 6) perform search in the space of complete candidate solutions. They are naturally divided in two subclasses by the number of candidate solutions operated at a time: *single-point* and *population-based* algorithms [Stützle, 1998]. In the single-point algorithms only one candidate solution is generated and evaluated at each iteration. These are such methods as Simulated Annealing, G-algorithm [Hulianyskyi, 1991], Tabu Search [Glover, 1986; Glover and Laguna, 1997], Variable Neighbourhood Search [Hansen et al., 2008], Variable Depth Search [Hoos and Stützle, 2005], Dynamic Local Search [Hoos and Stützle, 2005], Iterated Local Search, GRASP [Feo and Resende, 1989; Pitsoulis and Resende 2002], and others. These algorithms are often called *trajectory* because search process of most of them represents continuous trajectory on the neighbourhood graph. The concept of trajectory-continuity is discussed below.

Population-based algorithms manipulates at every iteration a set of candidate solutions. There are two major paradigms to which most of the population-based heuristics belong. Evolutionary Computation [De Jong, 2006; Fogel, 2006; Leguizamón et al., 2007] is one of the most prominent heuristic classes that has been inspired by concepts from biological evolution. Evolutionary methods used for CO problems include Genetic Algorithms [Holand, 1975; Fogel, 2006], Memetic algorithms [Moscato, 1989; Moscato and Cotta, 2007], Estimation of Distribution Algorithms [Larrañaga and Lozano, 2002]. There are also several evolutionary related approaches: Scatter Search [Glover et al., 2004], Artificial Immune Systems [Cutello and Nicosia, 2002; de Castro and Timmis, 2002]. In the CO context Swarm Intelligence [Bonabeau et al., 1999; Kennedy et al., 2001; Stigmergic

Optimization, 2006] represents the systems composed of many individuals that coordinate using decentralized control and self-organization.

Table 6. Classification by solution spaces utilized

Class		Examples
Sequential (constructive) algorithms		<i>Nearest Neighbour Heuristic</i> [Hoos and Stützle, 2005] ...
Iterative (perturbative) algorithms	Single-point algorithms	<i>Simulated Annealing</i> [Kirkpatrick et al., 1983; Aarts et al., 2007], <i>G-algorithm</i> [Hulianyskyi, 1991] ...
	Population-based algorithms	<i>Genetic Algorithm</i> [Holland, 1975; Fogel, 2006], <i>Discrete Particle Swarm Optimization</i> [Clerc, 2006], <i>H-method</i> [Hulianyskyi and Sergienko, 2007] ...

These methods include Ant Colony Optimization [Dorigo et. al., 1991; Dorigo and Stützle, 2004], Particle Swarm Optimization [Eberhart and Kennedy, 1995; Kennedy et al., 2001; Clerc, 2006], Stochastic Diffusion Search [De Meyer et al., 2006], methods inspired by bee colonies behaviour [Teodorovic et al., 2006; Bitam et al., 2008; Talbi, 2009], and others. Among methods that are not closely related to these two paradigms are Frontal Optimization Algorithms [Sergienko and Hulianyskyi, 1981], Cross-Entropy method [Rubinstein, 1999; Rubinstein and Kroese, 2004], and *H-method* [Hulianyskyi and Sergienko, 2007].

Sequential heuristics typically are the fastest solution methods for CO, but they often return solutions of inferior quality comparing to the iterative methods. In practice they are used for very large size problems or for problem with costly solution evaluation. Sequential algorithms are built-in in iterative methods for generating initial candidate solution(s) (in fact, random generation of initial solution also can be seen as construction process). They also can be incorporated in iterative methods in a more sophisticated way. For example, in GRASP and Ant Colony Optimization subordinate stochastic constructive procedures plays a key role. Iterated Greedy heuristic (see, for example, [Jacobs and Brusco, 1995; Ruiz and Stützle, 2007]) iterates greedy construction procedure, alternating between phases of destructive and constructive search similarly to it is performed with local search and perturbation phases in Iterated Local Search. Coined in [Roli and Milano, 2001; Gendreau and Potvin, 2005] frameworks for the (meta-) heuristics describe from different points of view the contribution of construction and improvement phases to the search process.

Memory presence. A very important feature to classify heuristics is the presence of a memory (Table 7). Memory is a set of special variables in which information about the performed search process is stored. There are several different ways of making use of memory (Table 8).

Memory can be used simply to store some information, which will be returned after by the algorithm (e.g. best found so far candidate solutions). This mechanism is used in almost every implementation of modern heuristics.

Search experience also can be used for guiding the search process. This type of memory can be short term or long term [Blum and Roli, 2003]. The former usually keeps track of recently performed moves, visited solutions or, in general, decisions taken. The latter is usually an accumulation of synthetic parameters about the search.

Table 7. Classification by memory presence

Class	Examples
Algorithms without memory	<i>G-algorithm</i> [Hulianytskyi, 1991] ...
Algorithms with memory	<i>Tabu Search</i> [Glover, 1986; Glover and Laguna, 1997] ...

Table 8. Memory utilization / functions

Type		Examples
Storing		Keeping best-so-far solution
Guiding	Short term	<i>Tabu list in Tabu Search</i> [Glover, 1986; Glover and Laguna, 1997] ...
	Long term	<i>Penalties in Guided Local Search</i> [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003] ...
Adaptive		<i>Pheromone values in Ant Colony Optimization</i> [Dorigo and Stützle, 2004]

Tabu list in Tabu Search is one of examples of short term memory. In Guided Local Search [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003], which is an example of specific Dynamic Local Search algorithm, in order to escape from local minimum penalties are introduced for the components of encountered local optimums. A set of penalty values correspond to long term memory. Some algorithms use both short and long term guiding memory (for example, specific Tabu Search variants [Glover and Laguna, 1997]). Guiding function of the memory is closely connected with a search landscape modification, which is discussed below. Majority of algorithms possessing this memory type make changes to the search landscape during their execution.

The last and the most important function of the memory is adaptation. Usage of search history for adaptation is discussed below.

In fact, current candidate (or partial) solution(s) can be seen as the most simple implementation of the memory. This memory is inherent in almost any heuristic (except of a random search) and is used for solution generation and other activity, depending on the complexity of the algorithm. While a population of candidate solutions such as in Genetic Algorithms is considered as a kind of memory [Stützle, 1998; Taillard et al., 2001; Gendreau and Potvin, 2005], current solution in a single-point algorithm is not accepted as a memory structure. Such algorithms as Simulated Annealing or GRASP in their standard implementation are generally referred to memoryless. This is due to eventually absent exploitation of the search history in these algorithms. To provide a formal basis for distinguishing memory using and memoryless algorithms we introduce the following definitions.

Definition 6. Given a deterministic search algorithm that generates at every iteration $I \in \mathbb{N}$ a set of candidate solutions $P_I \subseteq S$. The algorithm is called an *algorithm with memory* if does not exist a function $STEP: 2^S \rightarrow 2^S$ such that

$$P_{I+1} = STEP(P_I), I \geq 1.$$

Definition 7. Given a stochastic search algorithm that generates a set of candidate solutions $P_I \subseteq S$ at every iteration $I \in \mathbb{N}$ and is represented as a stochastic process with states P_I . The algorithm is called an *algorithm with memory* if

$$\exists S_I \subseteq S, I \in \mathbb{N}, \exists n \in \mathbb{N} : \Pr(P_{n+1} = S_{n+1} | P_n = S_n, \dots, P_1 = S_1) \neq \Pr(P_{n+1} = S_{n+1} | P_n = S_n).$$

In other words, the stochastic algorithm is memory using if the Markov property does not hold for the respective stochastic process with states P_I . However, it does not mean that the algorithm behavior cannot be modeled as a Markov process: stochastic process with states (P_I, M_I) , where M_I is a memory state at iteration I , will be a Markov chain.

Some memory structures can perform more than one function. For example, in Iterated Local Search variants the best solution found so far can be used as a starting point for the perturbation step. In this case the best solution performs both storing and guiding functions.

The use of memory is recognized as one of the fundamental elements of a powerful (meta-)heuristic. The Adaptive Memory Programming framework [Taillard et al., 2001] has been suggested to refer to algorithms that use some kind of memory and to identify common features among them.

Search Process Characteristics

Trajectory type. In [Stützle, 1998] there was suggested to characterize algorithms by whether they follow one single search trajectory corresponding to a closed walk on the neighbourhood graph or whether larger "jumps" in the neighbourhood graph are allowed. We introduce the following formal definition of this characteristic.

Definition 8. Given a set of neighbourhood structures $\{N_1, \dots, N_L\}$ and an algorithm that operates at every iteration $I \in \mathbb{N}$ a set of candidate solutions P_I , the algorithm is called *trajectory-continuous* if for any $I \geq 2$

$$P_I \subseteq \bigcup_{j=1}^L \bigcup_{x_{I-1} \in P_{I-1}} N_j(x_{I-1}) \quad (3)$$

If the set P_I consist of one element x_I only, that is the algorithm is a single-point, condition (3) is equivalent to the following: $\exists N^I \in \{N_1, \dots, N_L\} : x_I \in N^I(x_{I-1})$.

Definition 9. Any algorithm that do not meet condition (1) is called *trajectory-discontinuous* with respect to the set of neighbourhood structures $\{N_1, \dots, N_L\}$.

The trajectory-continuity of a heuristic with respect to the set of neighbourhood structures $\{N_1, \dots, N_L\}$ means that for every generated by the algorithm candidate solution $x \in X_I$ there exists a path in an aggregated neighbourhood graph $G_{N_1, \dots, N_L} = (S, \bigcup_{j \in \{1, \dots, L\}} V_{N_j})$ connecting one of the initial solutions and x . As it was

already mentioned, the search process of a single-point trajectory-continuous algorithm can be represented as a single trajectory (path) in the respective aggregated neighbourhood graph.

Table 9 lists examples of trajectory-continuous and trajectory-discontinuous algorithms in the context of a single neighbourhood structure. Local Search, Tabu Search and Simulated Annealing are typical examples of trajectory-continuous algorithms with respect to the neighbourhood used in them. Also local search algorithms that perform complex moves which are composed of simpler moves may be interpreted as trajectory-continuous [Stützle, 1998]. Such algorithms are, for example, variable depth methods [Hoos and Stützle, 2005], in particular, the Lin-Kernighan Algorithm for [Lin and Kernighan, 1973] and algorithms based on ejection chains [Glover, 1996].

Most of the population-based algorithms are trajectory-discontinuous with respect to any single neighbourhood structure. Among exceptions are Frontal Optimization Algorithms [Sergienko and Hulanyskyi, 1981]. A modification of this method, which is based on the Local Search, performs a number of parallel local steps with respect to predefined neighbourhood structure at every iteration.

The notion of trajectory-continuity is closely related to the number of neighbourhood structures that are used in the algorithm, which will be discussed below in the context of a search landscape modification: algorithms that use more than one neighbourhood structure are trajectory-discontinuous with respect to any single neighbourhood structure.

Influence on a search landscape. Another characteristic by which the heuristics can be classified is a search landscape modification during the search process (Table 10). There are three possibly types of such modifications (Table 11): the search space modification, evaluation function modification, and change of a neighbourhood structure.

Table 9. Classification by trajectory type (with respect to a single neighbourhood)

Class	Examples
Trajectory-continuous algorithms	<i>Tabu Search</i> [Glover, 1986; Glover and Laguna, 1997], <i>Frontal Optimization Algorithms</i> [Sergienko and Hulanyskyi, 1981], ...
Trajectory-discontinuous algorithms	GRASP [Feo and Resende, 1989; Pitsoulis and Resende 2002] ...

These changes are performed in order to escape from local minimum and to increase the efficiency of the algorithm.

Search space modification. The search space modification is either inclusion or exclusion some candidate solutions into it. In Tabu search, the returning to recently visited candidate solutions is forbidden, that is they are temporarily excluded from the search space. Other heuristics do not make changes to the search space.

Evaluation function change. Some heuristics modify the evaluation of the single candidate solutions during the run of the algorithm. In Breakout Method [Morris, 1993] penalties for the inclusion of certain solution components were introduced. This idea was generalized in Dynamic Local Search. In Noisy Methods (see, for example, [Charon and Hudry, 1993]) and Smoothing Methods (see, for example, [Gu and Huang, 1994]) evaluation function is also changed during the search process. For the review of last two approaches see [Talbi, 2009].

Solution construction procedures that are built-in Ant Colony Optimization algorithms may be interpreted as using a dynamic evaluation function: in the low level construction procedure partial solutions are evaluated using

pheromone values, which are updated at every iteration. At the same time at a high level the pheromone values represent a special memory structure, so the algorithm in general uses static evaluation function.

Tabu Search also can be interpreted as using a dynamic evaluation function: forbidding of some points in a search space corresponds to setting the infinitely high evaluation function values. But this interpretation does not reflect basic algorithmic idea of this approach.

Table 10. Classification by influence on a search landscape

Class	Examples
Algorithms that do not modify search landscape	<i>G-algorithm</i> [Hulianytskyi, 1991] ...
Algorithms that modify search landscape	<i>Tabu Search</i> [Glover, 1986; Glover and Laguna, 1997], <i>Guided Local Search</i> [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003], ...

Table 11. Types of search landscape modification

Type	Examples
Search space modification	<i>Tabu Search</i> [Glover, 1986; Glover and Laguna, 1997]
Evaluation function change	<i>Guided Local Search</i> [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003], ...
Neighborhood change	<i>Search in Pulsating Neighborhoods</i> [Hulianytskyi and Khodzinskyi, 1979], <i>Variable Neighborhood Search</i> [Mladenovic and Hansen, 1997], ...

Neighborhood change. Many heuristics use single neighbourhood structure, which defines allowed moves on a neighbourhood graph. Iterated Local Search typically use at least two different neighbourhood structures N and N' : the Local Search starts with neighbourhood structure N until a local optimum is reached and then a perturbation of obtained solution is performed, which can be interpreted as a move in a secondary neighbourhood structure N' . This idea was extended in independently developed Search in Pulsating Neighborhoods [Hulianytskyi and Khodzinskyi, 1979] and Variable Neighborhood Search [Mladenovic and Hansen, 1997], where neighbourhood structures are systematically changed among a predefined list. Tabu Search also can be interpreted as using the dynamic neighbourhoods [Hertz et al., 1995].

The solution construction process in Ant Colony Optimization or in GRASP can be interpreted as a search on a neighbourhood graph with the set of vertexes defined by (2) and the neighbourhood structure defined as

$$\forall \varphi^{i-1} \in X_{i-1}, \varphi^{i-1} : Y_{i-1} \rightarrow Z, N(\varphi^{i-1}) = \left\{ \varphi^i : Y_i \rightarrow Z : \left(\varphi^i \in X_i, Y_{i-1} \subset Y_i, \varphi^i|_{Y_{i-1}} \equiv \varphi^{i-1} \right) \right\}, i = 2, \dots, m.$$

Thus, GRASP and Ant Colony Optimization algorithms with Local Search uses at least two neighbourhood structures, switching them between solution construction and improvement phases. Many other hybrid algorithms also perform neighbourhood switching during execution.

Adaptation/learning presence. We will distinct algorithms that adapt (learn) during the search process (Table 12). Adaptation (learning) is an ability of the heuristic to adjust its behaviour during (before) the search process. This includes dynamically tuning parameter values, automatically selecting subordinate routines, and presence of a problem model (Table 13). The field of adaptive/learning approaches for the CO problems is still developing and is rather a collection of independent techniques.

Reactive Search [Battiti and Brunato, 2007] provides a mechanism to include the parameter tuning within the algorithm: parameters are adjusted by an automated feedback loop that acts according to the quality of the solutions found, the past search history and other criteria. In Reactive Tabu Search [Battiti and Tecchiolli, 1994], one of the first reactive methods, a period of prohibition is automatically adjusted during the search.

Automated selecting of subroutines is performed in algorithms that belong to the mentioned above class of hyperheuristics.

Specific adaptation techniques are also developing within individual methods. Examples vary from adaptive cooling schedule (dynamical parameter adjust) in Simulated Annealing [Aarts et al., 2007] to dynamical meme (subordinate local search routine) selection in Adaptive Memetic Algorithms [Ong et. al., 2006].

Adaptation can also be achieved through the usage of a special memory structure that represents a model of the problem, which is discussed below.

Mentioned above approaches for adaptation/learning in heuristics do not cover all of the proposed approaches and techniques.

Table 12. Classification by adaptation/learning

Class	Examples
Algorithms without adaptation/learning	<i>Iterated Local Search</i> [Lourenço et al., 2002] ...
Algorithms with adaptation/learning	<i>Reactive Tabu Search</i> [Battiti and Tecchiolli, 1994], Adaptive Memetic Algorithms [Ong et. al., 2006], ...

Table 13. Types of adaptation

Type	Examples
Parameters tuning	<i>Adjusting tabu tenure in Reactive Tabu Search</i> [Battiti and Tecchiolli, 1994], ...
Subroutines selecting	Hyperheuristics [Burke et al., 2003; Ozcan et al., 2008]
Problem model presence	Model-based search [Zlochin et al., 2004]

Problem model presence. According to the presence of the problem model heuristic algorithms can be classified as being either *instance-based* or *model-based* [Zlochin et al., 2004] (Table 14). Most of the classical search methods may be considered instance-based, since they generate new candidate solutions using solely the current candidate solution or the current "population" of candidate solutions. Typical representatives of this class are Genetic Algorithms, Simulated Annealing and Iterated Local Search. In model-based search algorithms, candidate solutions are generated using a parameterized probabilistic model that is updated using the previously

seen candidate solutions in such a way that the search will concentrate in the regions containing high quality solutions.

In model-based methods the tackled optimization problem is replaced by the following continuous maximization problem [Zlochin et al., 2004]

$$\tau^* = \arg \max_{\tau} W_F(\tau),$$

where τ is a parameter values of the respective model and $W_F(\cdot)$ denotes the expected quality of a generated solutions depending on the values of the parameters.

Well-known model-based method is the Ant Colony Optimization. The distinctive feature of Ant Colony Optimization is a particular type of probabilistic model, in which a structure called *construction graph* is coupled with a set of stochastic procedures called *artificial ants*.

Table 14. Classification by problem model presence

Class	Examples
Instance-based algorithms	<i>Simulated annealing</i> [Kirkpatrick et al., 1983; Aarts et al., 2007], <i>Genetic Algorithm</i> [Holland, 1975; Fogel, 2006] ...
Model-based algorithms	<i>Ant Colony Optimization</i> [Dorigo and Stützle, 2004], <i>Cross Entropy Method</i> [Rubinstein and Kroese, 2004], <i>Estimation of Distribution Algorithms</i> [Larrañaga and Lozano, 2002] ...

Introduced in the field of evolutionary computations the Estimation of Distribution Algorithms [Mühlenbein and Paaß, 1996; Larrañaga and Lozano, 2002] may be considered a particular realization of model-based search with an auxiliary memory that stores high-quality solutions encountered during the search.

The Cross-Entropy Method [Rubinstein and Kroese, 2004], originated from the field of rare event simulation, where very small probabilities need to be accurately estimated, is another example of model-based search algorithm used for CO problems.

Performance Characteristics

Performance guarantees. Although in general heuristics have no performance guaranties [Handbook of Applied Optimization, 2002; Hoos and Stützle, 2005; Handbook of Approximation Algorithms and Metaheuristics, 2007; Talbi, 2009], in some special cases or for particular classes of problems some theoretical results can be derived. A lot of different performance guaranties definitions were proposed, but all of them can be divided in two classes a priori and a posteriori. A *priori* performance guarantee is a guarantee that follows directly from the algorithm. A *posteriori* performance guarantee is a guarantee that is calculated during the algorithms execution and with respect to the obtained solution(s). In [Sergienko et al., 1989] for a special problem subclass a posteriori performance guarantee for the Local Search was derived. Some examples for these two classes are shown in Tables 15.

Convergence-related properties (for stochastic algorithms). A convergence to the globally optimal solution is an important property of the stochastic optimization algorithm. But theoretical study of the heuristics is rather difficult and rarely provides practically applicable results. A number of convergence and related notions definitions were introduced to describe algorithm's behavior. In the context of CO particularly a convergence in value is important.

Definition 9. Let $p^*(I)$ be a probability of a stochastic algorithm for optimization to generate an optimal solution at least once at first I iterations. The algorithm is called *convergent in value* if

$$\lim_{I \rightarrow \infty} p^*(I) = 1.$$

This property guarantees the algorithm not to be trapped in a suboptimal area of the search space. Convergence in value is equivalent to the probabilistic approximate completeness property and has been proven for Simulated Annealing, specific Ant Colony Optimization algorithms, specific Tabu Search algorithms, Evolutionary Algorithms, and others [Hoos and Stützle, 2005].

Table 16 lists few results on convergence the heuristic methods. As in the case of performance guaranties, for many of the well known and widely used stochastic heuristics the practically relevant results on performance are still not obtained.

Table 15. Classification by a priori performance guarantee presence

Class	Examples
Algorithms with performance guarantee	<i>ε-approximate algorithms, Local Search with known lower bound</i> [Papadimitriou and Steiglitz, 1982; Sergienko et al., 1989; Handbook of Approximation Algorithms and Metaheuristics, 2007; Kochetov, 2008] ...
Algorithms without performance guarantee	[Handbook of Applied Optimization, 2002; Hoos and Stützle, 2005; Talbi, 2009] ...

Table 16. Classification by convergence

Class	Examples
Algorithms convergent to the global solution	<i>Simulated Annealing</i> [Hajek, 1988], <i>Genetic Algorithm with elitism</i> [Rudolph, 1994] <i>ACO_{τ_{\min}}</i> [Dorigo and Blum, 2005], ...
Do not convergent algorithms	<i>Standard Genetic Algorithm</i> [Holland, 1975], ...
Algorithms without results on convergence	[Handbook of Applied Optimization, 2002; Hoos and Stützle, 2005; Talbi, 2009] ...

Figure 1 summarizes the presented classification and shows relationships between classes.

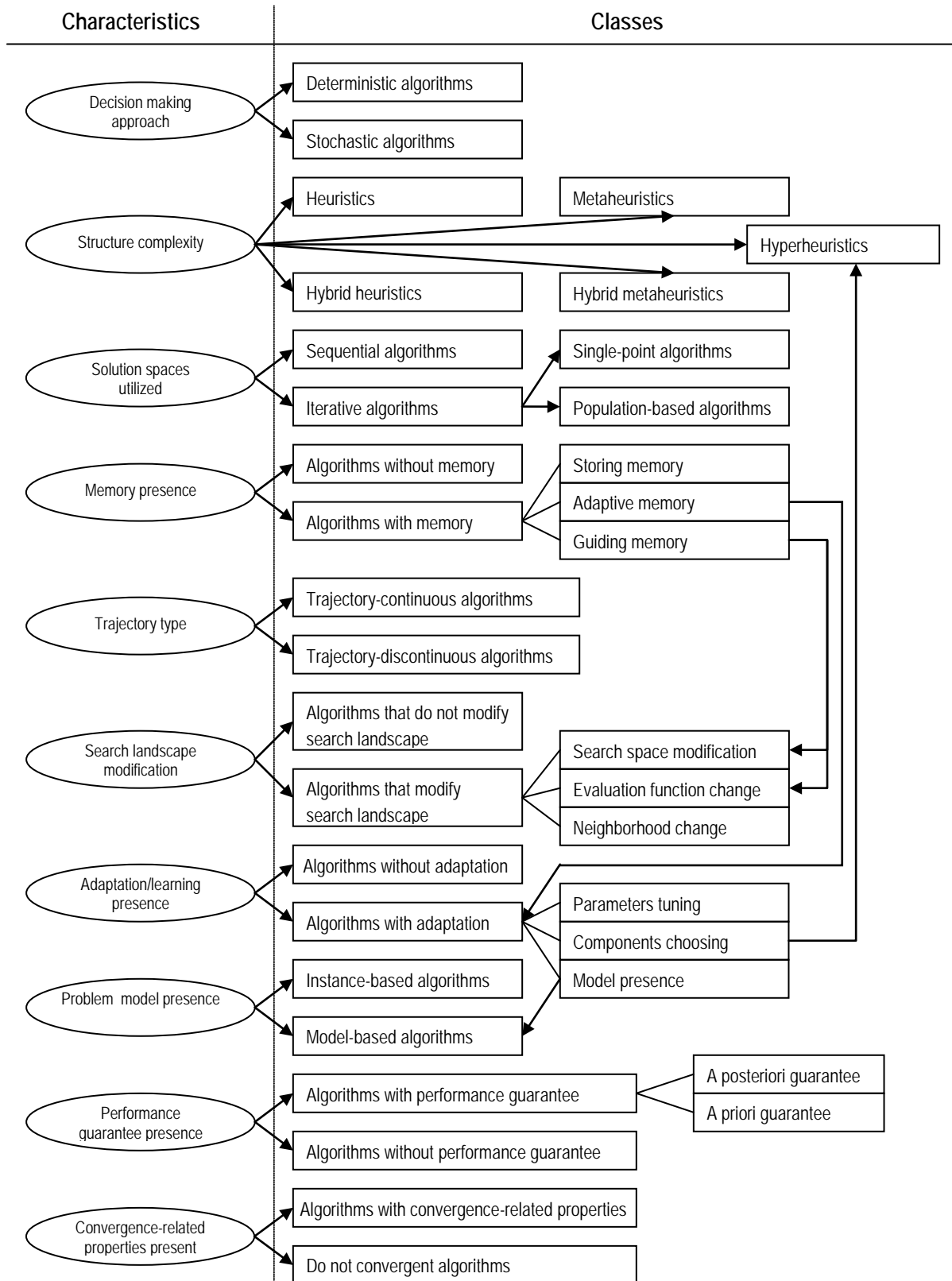


Figure 1. Classification of heuristic methods for combinatorial optimization

Conclusion

Heuristics represent a practically relevant class of incomplete methods for combinatorial optimization problems. In the paper the classification of the heuristic algorithms is suggested. It extends and generalizes other approaches for the classification of heuristics and their special subclasses in combinatorial optimization. It is based on defining characteristics by which the heuristics can be classified. These characteristics can be divided into three categories: structure characteristics, search process characteristics and performance characteristics. Structure characteristics are decision making approach, structure complexity, solution spaces utilized, memory presence. Search process characteristics include trajectory-continuity, search landscape modification, adaptation presence, and problem model presence. Performance characteristics consist of a priori and a posteriori approximation guarantees and convergence-related properties. Considered aspects of different heuristic approaches are general, but for most of them the formal definition are introduced. The exception is the structure complexity: terms "heuristics", "metaheuristic", and "hyperheuristics", being widely used, have no formal and at a time generally accepted definition.

Supplementary to the classification is formulation of a generalized (but enough detailed at a time) framework for heuristics search in CO. Presented characteristics and classes provides the basis for designing such a framework, which can address two problems. First, it can provide a more formal basis for classifying algorithms with complex structure. There are a number of algorithms that have shown their practical applicability, but being "hybrid" cannot be classified strictly within presented approach. Second, it can serve as skeleton for designing new efficient algorithms. Developing the framework for heuristic methods is an important issue for future research.

Acknowledgements

The author would like to thank L.F. Hulianytskyi for his helpful comments on drafts of this paper.

Bibliography

- [Aarts et al., 2007] E. Aarts, J. Korst, W. Michiels. Simulated annealing. In: Handbook of approximation algorithms and metaheuristics (Ed. T.F. Gonzalez). Boca Raton: CRC press, 2007. pp. 387–397.
- [Battiti and Brunato, 2007] R. Battiti, M. Brunato. Reactive search: Machine learning for memory-based heuristics. In: Handbook of approximation algorithms and metaheuristics (Ed. T.F. Gonzalez). Boca Raton: CRC press, 2007. pp. 329–345. <http://rtm.science.unitn.it/~battiti/archive/chap21.pdf>
- [Battiti and Tecchioli, 1994] R. Battiti, G. Tecchioli. The Reactive Tabu Search. In: ORSA Journal on Computing. 1994, Vol. 6, No. 2. pp. 126–140. <http://rtm.science.unitn.it/~battiti/.../reactive-tabu-search.ps.gz>
- [Bertsekas et al., 1997] D.P. Bertsekas, J.N. Tsitsiklis, C. Wu. Rollout Algorithms For Combinatorial Optimization. In: Journal of Heuristics, 1997, No. 3. pp. 245–262. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.5095&rep=rep1&type=pdf>
- [Birattari et al., 2001] M. Birattari, L. Paquete, T. Stützle, K. Varrentrapp. Classification of metaheuristics and design of experiments for the analysis of components. Technical Report AIDA-01-05. Technische Universität Darmstadt, 2001 12 p. <http://www.intellektik.informatik.tu-darmstadt.de/TR/2001/01-05.pdf>
- [Bitam et al., 2008] S. Bitam, M. Batouche, E.-G. Talbi. A taxonomy of artificial honeybee colony optimization. In: META'08 International Conference on Metaheuristics and Nature Inspired Computing (Eds. E.-G. Talbi, K. Mellouli), Hammamet, Tunisia, 2008.

-
-
- [Blum and Roli, 2003] C. Blum, A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. In: ACM Computing Surveys, 2003, Vol. 35, No. 3. pp. 268–308. <http://iridia.ulb.ac.be/~meta/newsite/downloads/ACSUR-blum-roli.pdf>
- [Bonabeau et al., 1999] E. Bonabeau, M. Dorigo, G. Theraulaz. Swarm Intelligence: From Natural to Artificial System. Oxford University Press, New York, 1999. 320 p.
- [Burke et al., 2003] E. Burke, E. Hart, G. Kendall, J. Newall, P. Ross and S. Schulenburg. Hyperheuristics: An Emerging Direction in Modern Search Technology. In: Handbook of Metaheuristics (eds. Glover F., Kochenberger G.). Kluwer Academic Publishers, 2003. pp. 457–474. <http://www.cs.nott.ac.uk/~gxx/papers/hhchap.pdf>
- [de Castro and Timmis, 2002] L.N. de Castro, J. Timmis. Artificial Immune Systems: A New Computational Intelligence Approach. Springer, 2002. 380 p.
- [Charon and Hudry, 1993] I. Charon, O. Hudry. The noising method: A new method for combinatorial optimization. In: Operations Research Letters, 1993, No. 14. pp. 133–137.
- [Clerc, 2006] M. Clerc. Particle Swarm Optimization. ISTE, London, 2006. 244 p.
- [Cutello and Nicosia, 2002] V. Cutello, G. Nicosia. An immunological approach to combinatorial optimization problems. In: Lect. Notes Computer Sci., 2002, No. 2527. pp. 361–370.
- [De Jong, 2006] K.A. De Jong. Evolutionary Computation: A Unified Approach. MIT Press, 2006. 272 p.
- [De Meyer et al., 2006] K. De Meyer, S.J. Nasuto, J.M. Bishop. Stochastic diffusion search: partial function evaluation in swarm intelligence dynamic optimisation. In: Stigmergic optimization (Eds. A. Abraham, C. Grosam, V. Ramos). Springer Verlag, 2006. pp. 185–208. <http://www.doc.gold.ac.uk/~mas02mb/Selected%20Papers/2006%20Swarm%20Intelligence.pdf>
- [Dorigo et. al., 1991] M. Dorigo, V. Maniezzo, A. Coloni. Positive feedback as a search strategy. Tech. Rep. 91-016. Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica, 1991. 22 p. <http://iridia.ulb.ac.be/pub/mdorigo/tec.reps/TR.01-ANTS-91-016.ps.gz>
- [Dorigo and Stützle, 2004] M. Dorigo, T. Stützle. Ant Colony Optimization. Cambridge: MIT Press, 2004. 348 p.
- [Dorigo and Blum, 2005] M. Dorigo, C. Blum. Ant colony optimization theory: A survey. In: Theoretical computer science, 2005, Vol. 344, No. 2–3. pp. 243–278. <http://code.ulb.ac.be/dbfiles/DorBlu2005tcs.pdf>
- [Eberhart and Kennedy, 1995] R.C. Eberhart, J. Kennedy. Particle swarm optimization. In: Proc. IEEE International Conference on Neural Networks, Piscataway, NJ, 1995. pp. 1942–1948. <http://www.engr.iupui.edu/~shi/Coference/psopap4.html>
- [Feo and Resende, 1989] T.A. Feo, M.G.C. Resende. A Probabilistic Heuristic For A Computationally Difficult Set Covering Problem. In: Operations Research Letters, 1989, Vol. 8, No. 2. pp. 67–71.
- [Fogel, 2006] D.B. Fogel. Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. 3rd Edition. Wiley-IEEE Press, 2006. 296 p.
- [Garey and Johnson, 1979] M.R. Garey, D.S. Johnson. Computers and intractability: a guide to the theory of NP-completeness. WH Freeman and Company, 1979. 338 p.
- [Gendreau and Potvin, 2005] M. Gendreau, J.-Y. Potvin. Metaheuristics in Combinatorial Optimization. In: Annals of Operations Research, 2005, Vol. 140, No. 1. pp. 189–213.
- [Glover, 1986] F. Glover. Future paths for integer programming and links to artificial intelligence. In: Computers & Operations Research, 1986, Vol. 5. pp. 533–549.
- [Glover, 1996] F. Glover. Ejection Chains, Reference Structures and Alternating Path Methods for Traveling Salesman Problems. In: Discrete Applied Mathematics, 1996, Vol. 65, No. 1–3. pp. 223–253.
- [Glover and Laguna, 1997] F. Glover, M. Laguna. Tabu Search. Kluwer Academic Publishers, 1997. 408 p.
- [Glover et al., 2004] F. Glover, M. Laguna, R. Martí. Scatter search and path relinking: foundations and advanced designs. In: New optimization techniques in engineering (Eds. G. Onwubolu, B.V. Babu). Springer-Verlag, 2004. pp. 87–100. <http://www.uv.es/~rmarti/paper/docs/ss7.pdf>

-
-
- [Gu and Huang, 1994] J. Gu, X. Huang. Search Space Smoothing: A Case Study of the Traveling Salesman Problem. IEEE Trans. Systems, Man, and Cybernetics, 1994, Vol. 24, No. 5. pp. 728–735.
- [Handbook of Applied Optimization, 2002] Handbook of Applied Optimization (eds. P. Pardalos, M.G.C. Resende). Oxford University Press, 2002. 2026 p.
- [Handbook of Approximation Algorithms and Metaheuristics, 2007] Handbook of Approximation Algorithms and Metaheuristics (ed. Gonzalez T.F.). Chapman & Hall/CRC, 2007. 1432 p.
- [Hajek, 1988] B. Hajek. Cooling schedules for optimal annealing. In: Mathematics of Operations Research, 1988, Vol. 13, No. 2. pp. 311–329. <http://web.mit.edu/6.435/www/Hajek88.pdf>
- [Hansen et al., 2008] P. Hansen, N. Mladenovic, J.A. Moreno-Pérez. Variable neighbourhood search: methods and applications. In: Quarterly Journ. Oper. Res, 2008, Vol. 6, No. 4. pp. 319–360.
- [Hertz et al., 1995] A. Hertz, E. Taillard, D.A. Werra. A Tutorial on Tabu Search. In: Proc. of Giornate di Lavoro AIRO'95, Enterprise Systems: Management of Technological and Organizational Changes, 1995. pp. 13–24. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.28.521&rep=rep1&type=pdf>
- [Holland, 1975]. J.H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975. 183 p.
- [Hoos and Stützle, 2005] H.H. Hoos, T. Stützle. Stochastic Local Search: Foundations and Applications. San Francisco: Morgan Kaufmann Publ, 2005. 658 p.
- [Hulianytskyi, 1991] L.F. Hulianytskyi. Modified stochastic modeling algorithms in combinatorial optimization. In: Technology and methods of solving applied mathematics' problems (in Russian). Kyiv: Institute of Cybernetics NAS Ukraine, 1991. pp. 10–14.
- [Hulianytskyi and Khodzinskyi, 1979] L.F. Hulianytskyi, A.N. Khodzinskyi. Implementation features of algorithms branch and bound method and decrease vector method in VECTOR–1V package. In: Computational aspects of application packages (in Russian). Kyiv: Institute of Cybernetics AS Ukrainian SSR, 1979. pp. 25–30.
- [Hulianytskyi and Sergienko, 2007] L.F. Hulianytskyi, I.V. Sergienko. Metaheuristic downhill simplex method in combinatorial optimization. In: Cybern. Syst. Anal., 2007, Vol. 43, No. 6. pp. 822–829.
- [Jacobs and Brusco, 1995] L.W. Jacobs, M. J. Brusco. A local search heuristic for large set-covering problems. In: Naval Research Logistics Quarterly, 1995, Vol. 42, No. 7. pp. 1129–1140.
- [Kennedy et al., 2001] J. Kennedy, R.C. Eberhart, Y. Shi. Swarm Intelligence. Morgan Kaufmann, San Francisco, CA, 2001. 512 p.
- [Kernighan and Lin, 1970] B.W. Kernighan, S. Lin. An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal., 1970, Vol. 49. pp. 291–307. <http://www.cs.princeton.edu/~bwk/btl.mirror/new/partitioning.pdf>
- [Khuller et al., 2007] S. Khuller, B. Raghavachari, N.E. Young. Greedy methods. In: Handbook of approximation algorithms and metaheuristics (Ed. T.F. Gonzalez). Boca Raton: CRC press, 2007. pp. 67–80.
- [Kirkpatrick et al., 1983] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi. Optimization by simulated annealing. In: Science, 1983, Vol. 220, No. 4598. pp. 671–680. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.18.4175&rep=rep1&type=pdf>
- [Kochetov, 2008] Yu.A. Kochetov. Computational capabilities of local search in combinatorial optimization. In: Computational Mathematics and Mathematical Physics (in Russian), 2008, Vol. 48, No. 5. pp. 788–807.
- [Korte and Vygen, 2006] B.H. Korte, J. Vygen. Combinatorial Optimization: Theory and Algorithms. Springer-Verlag, 2006. 595 p.
- [Larrañaga and Lozano, 2002] P. Larrañaga, J.A. Lozano. Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation. Boston, MA: Kluwer Academic, 2002. 382 p.
- [Leguizamón et al., 2007] G. Leguizamón, C. Blum, E. Alba. Evolutionary Computation. In: Handbook of Approximation Algorithms and Metaheuristics (ed. Gonzalez T.F.). Boca Raton: CRC press, 2007. pp. 372–386.
- [Leont'ev, 2007] V. Leont'ev. Discrete optimization. In: Computational Mathematics and Mathematical Physics (in Russian), 2007, Vol. 47, No. 2. pp. 328–340.

-
-
- [Lourenço et al., 2002] H.R. Lourenço, O. Martin, T. Stützle. Iterated local search. In: Handbook of Metaheuristics: International Series in Operations Research & Management Science, vol. 57 (Eds. F. Glover and G. Kochenberger). Norwell: Kluwer Academic Publishers, 2002. pp. 321–353.
- [Mladenović and Hansen, 1997] N. Mladenović, P. Hansen. Variable Neighbourhood Search. In: Computers & Operations Research, 1997, Vol. 24. pp. 1097–1100.
- [Morris, 1993] P. Morris. The Breakout Method for Escaping from Local Minima. In: Proceeding of the 11th Conference on Artificial Intelligence, MIT Press, 1993. pp. 40–45.
- [Moscato, 1989] P. Moscato. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Techn. Rep. C3P 826. Pasadena, California Institute of Technology, 1989. 68 p.
<http://www.densis.fee.unicamp.br/~moscato/papers/bigone.ps>
- [Moscato and Cotta, 2007] P. Moscato, C. Cotta. Memetic algorithms. In: Handbook of approximation algorithms and metaheuristics (Ed. T.F. Gonzalez). Boca Raton: CRC press, 2007. pp. 412–423.
- [Mühlenbein and Paaß, 1996] H. Mühlenbein, G. Paaß. From recombination of genes to the estimation of distributions. I. Binary parameters. In: Lect. Notes Computer Science: Parallel Problem Solving from Nature, 1996, No. 4. pp. 178–187.
www.iais.fhg.de/fileadmin/images/pics/Abteilungen/INA/muehlenbein/PDFs/technical/from-recombination-of-genes1.pdf
- [Ong et al., 2006] Y.-S. Ong, M.-H. Lim, N. Zhu, K.-W. Wong. Classification of adaptive memetic algorithms: a comparative study. In: IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics. 2006, Vol. 36, No. 1. pp. 141–152. http://www3.ntu.edu.sg/SCE/labs/erlab/tech_report/05-004.pdf
- [Ozcan et al., 2008] E. Ozcan, B. Bilgin, E. E. Korkmaz. A Comprehensive Analysis of Hyper-heuristics. In: Intelligent Data Analysis, 2008, Vol.12, No. 1. pp. 3–23. <http://cse.yeditepe.edu.tr/~eoacan/research/papers/IDA08.pdf>
- [Papadimitriou and Steiglitz, 1982] C. Papadimitriou, K. Steiglitz. Combinatorial optimization: algorithms and complexity. Prentice Hall, Englewood Cliffs, NJ, 1982. 512 p.
- [Pitsoulis and Resende 2002] L. Pitsoulis, M.G.C. Resende. Greedy randomized adaptive search procedures. In: Handbook of applied optimization (Eds. P.M. Pardalos, M.G.C. Resende). Oxford Univ. Press, 2002. pp. 168–181.
- [Raidl, 2006] G.R. Raidl. A Unified View on Hybrid Metaheuristics. In: Lecture Notes in Computer Science. Springer-Verlag, 2006, Vol. 4030. pp. 1–12. <http://www.ads.tuwien.ac.at/publications/bib/pdf/raidl-06.pdf>
- [Roli and Milano, 2001] A. Roli, M. Milano. Metaheuristics: A Multiagent Perspective. DEIS Technical Report no. DEIS-LIA-01-006. University of Bologna, 2001. 23 p. <http://www-lia.deis.unibo.it/Research/TechReport/lia01006.ps.gz>
- [Rubinstein, 1999] R.Y. Rubinstein. The Cross-Entropy Method for Combinatorial and Continuous Optimization. In: Methodology and Computing in Applied Probability, 1999, Vol. 2, No. 1. pp. 127–190.
<http://iew3.technion.ac.il/~ierr01/PAPERS/entropy.ps>
- [Rubinstein and Kroese, 2004] R.Y. Rubinstein, D.P. Kroese. The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning. Springer, 2004. 300 p.
- [Rudolph, 1994] G. Rudolph. Convergence Analysis of Canonical Genetic Algorithms. In: IEEE Transactions on Neural Networks, 1994, Vol. 5. pp. 96–101. <http://ls11-www.cs.uni-dortmund.de/people/rudolph/publications/papers/TNN5.1.pdf>
- [Ruiz and Stützle, 2007] R. Ruiz, T. Stützle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. In: Eur. J. Oper. Res., Vol. 177, No. 3. pp. 2033–2049.
http://iridia.ulb.ac.be/~stuetzle/publications/IG_FSP.ps.gz
- [Sergienko, 1964] I.V. Sergienko. A method to solve the problem of finding extreme values. In: Automatics (in Ukrainian), 1964, No 5. pp. 15–21.
- [Sergienko and Huliantskyi, 1981] I.V. Sergienko, L.F. Huliantskyi. Frontal optimization algorithms for multiprocessor computers. In: Cybernetics, 1981, Vol. 17, No. 6. pp. 719–721.
- [Sergienko et al., 1989] I.V. Sergienko, L.F. Huliantskyi, S.A. Malyshko. On solving layout problems of one class. In: Econ. and Math. Methods (in Russian), 1989, Vol. XXV, No. 3. P. 560–564.

- [Sergienko and Shilo, 2003] I.V. Sergienko V.P. Shilo. Discrete Optimization Problems. Challenges, Solution Techniques, and Study (in Russian). Kyiv: Naukova Dumka, 2003. 261 p.
- [Stigmergic Optimization, 2006] Stigmergic Optimization (Eds. A. Abraham, C. Grosan, V. Ramos). Springer-Verlag, 2006. 310 p.
- [Stützle, 1998] T. Stützle. Local Search Algorithms for Combinatorial Problems – Analysis, Improvements and New Applications. PhD. Thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1998. 214 p.
<http://iridia.ulb.ac.be/~stuetzle/publications/Thesis.ThomasStuetzle.pdf>
- [Taillard et al., 2001] E.D. Taillard, L.M. Gambardella, M. Gendreau, J.-Y. Potvin. Adaptive Memory Programming: A Unified View of Metaheuristics. In: Eur. J. Oper. Res., 2001, No. 135. pp. 1–6.
http://ina2.eivd.ch/Collaborateurs/etd/articles.dir/ejor135_1_1_16.pdf
- [Talbi, 2009] Talbi E.-G. Metaheuristics: From Design To Implementation. Hoboken, New Jersey: John Wiley & Sons, 2009, 618 p.
- [Teodorovic et al., 2006] D. Teodorovic, P. Lucic, G. Markovic, M. D'Orco. Bee colony optimization: principles and applications. In: 8th Seminar on Neural Network Applications in Electrical Engineering, Belgrade, Serbia, 2006. pp. 151–156.
- [Vaessens et al., 1998] R. J. M. Vaessens, E. H. L. Aarts, J.K. Lenstra. A local search template. In: Computers and Operations Research, 1998, Vol. 25, No. 11. pp. 969–979.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.4457&rep=rep1&type=pdf>
- [Voudoris and Tsang, 1995] C. Voudoris, E. Tsang. Guided Local Search. Technical Report CSM-247, Department of Computer Science, University of Essex, England. 1995.18
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.49.9608&rep=rep1&type=pdf>
- [Voudouris and Tsang, 2003] C. Voudouris, E.P.K. Tsang. Guided local search. In: Handbook of metaheuristics (Ed. F. Glover). Norwell: Kluwer Acad. Publ, 2003. pp. 185–218.
- [Zhuravlev, 1977] Yu. I. Zhuravlev. Correct Algebras over Sets of Incorrect (Heuristic) Algorithms: Part I," Kibernetika, 1977, No. 4. pp. 5–17.
- [Zlochin et al., 2004] M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo. Model-Based Search for Combinatorial Optimization: A Critical Survey. In: Annals of Operations Research, 2004, Vol. 131. pp. 373–395.
<http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.66.672&rep=rep1&type=pdf>

Authors' Information



Sirenko Sergii – Postgraduate student, department 135, V.M. Glushkov Institute of Cybernetics National Academy of Sciences of Ukraine, 40 Prospekt Akademika Glushkova, 03680 Kyiv, Ukraine;

e-mail: mail@sirenko.com.ua

Major Fields of Scientific Research: Combinatorial Optimization, Metaheuristics, Swarm Intelligence

PARALLELIZATION METHODS OF LOGICAL INFERENCE FOR CONFLUENT RULE-BASED SYSTEM¹

Irene Artemieva, Michael Tyutyunnik

Abstract: *The article describes the research aimed at working out a program system for multiprocessor computers. The system is based on the confluent declarative production system. The article defines some schemes of parallel logical inference and conditions affecting scheme choice. The conditions include properties of a program information graph, relations between data objects, data structures and input data as well.*

Keywords: *logical Inference, parallel rule-based systems*

ACM Classification Keywords: *D 3.2 – Constraint and logic languages, I 2.5 Expert system tools and techniques.*

Introduction

Development of computer architecture and network technologies, new theoretical and applied problems requiring a lot of computations, slow running speed of sequential computer systems and theoretically limited growth of their efficiency resulted in the necessity of using multiprocessor and multicore computer systems thus making parallel computations take centre stage in modern programming and computing technologies. This, in return, gave impetus to development of programming languages and language processors used to create applications for such systems.

At present, there are several directions for development of languages and language processors. First, high-level programming languages developed for sequential machines are extended via special tools for parallel computations and their parallel versions are designed (e.g., parallel versions of Fortran, C/C++, Modula-3 [3-6]). Developers of parallel programs write problem-solving algorithms using these means.

Second, there are special communication libraries and interfaces for organizing interprocessor interaction (e.g., PVM, MPI, MPL, OpenMP, ShMem [7-11]). They can be used in programs written in high-level languages. Developers of parallel programs organize parallel processes in a program aimed at solving problems on a multiprocessor computer system by means of these libraries and interfaces.

Third, language processor designed specifically for multiprocessor computer system implements automatic and semi-automatic parallelization of computations. In this case, methods of organization of parallel computations are hidden from developers of programs solving applied tasks (e.g., BERT 77, PIPS, VAST/Parallel [12-14]). Automatic parallelization of computations found practical use in packages of applied programs (e.g., ATLAS, DOUG, GALOPPS, NAMD, ScaLAPACK [15]) aimed at solving problems in specific areas.

¹ This paper was made according to the program № 2 of fundamental scientific research of the Presidium of the Russian Academy of Sciences, the project 09-I-П2-04

Fourth, there are programming languages or systems specialized for certain architecture or problems (e.g., KL1 and languages for programming on vector or matrix computers). These languages use parallelization of nested data, enable description of pipeline parallelism and parallelism at the level of problems.

Using high-level languages extended by means of organization tools for parallel computations or special communication libraries and interfaces urges developers of parallel programs to have special knowledge enabling creation of efficient parallel problem-solving algorithm. During parallelization of programs describing sequential problem-solving algorithm, compiler usually finds only a few fragments of a code for parallelization. Specialized languages are either inaccessible to a broad spectrum of specialists or applicable only to specialized architectures.

Problem-solving method may be represented either as an algorithm or as a set of rules (productions). There are many classes of applications using rules. Such representation of method is often used to create knowledge-based systems requiring thorough search for solutions and uses a lot of data stored in system knowledge bases. Representation of problem as a set of rules is a more natural way compared to an algorithm and does not urge program system developers to have special knowledge about organization of computing which is controlled by a language processor of rule-based language.

By now, representation of solution method as a set of rules is supported by various rule-based systems and the Prolog language. There are also parallel language processors for the Prolog language [16] and other rule-based languages. However, results of problem solutions, where the Prolog language (which is claimed to be rule-independent) is used, depend on the writing order of these rules, i.e. the language does not possess inherent parallelism. Results of problem solutions in rule-based systems which are not confluent also depend on the implementation order of these rules during the logical inference. Thus, when creating knowledge-based systems for multiprocessor computer systems with the usage of the Prolog language or rule-based non-confluent system, developers face the same problems as when they use languages of other classes.

In rule-based confluent systems, the result of the logical inference does not depend on the implementation order of these rules, i.e. such systems possess inherent parallelism. A language processor – a production language compiler allocates computations according to a process. When generating an object code, a language processor analyzes characteristics of the source program, a set of constraints imposed by the computing environment, characteristics of input data defined by the user and selects the most applicable scheme of the parallel logical inference.

The aim of this article is to describe schemes of the parallel logical inference implemented by the language processor of the confluent production system and conditions affecting the scheme choice.

Production System Language Characteristics

The research on ontologies [2] and design knowledge-based systems [1] on their basis makes it possible to formulate requirements for the language of the confluent production system.

1. The language must allow to represent a problem-solving method as a set of solving methods for subtasks described by the modules. Each module must have its interface, i.e. data description, that can be used by another module or that are required for its operation/performance. There must be an explicit condition of a module call, i.e. among the rules there can be rules the right part of which is a module call.
2. The language must allow to use operations on numeric data and sets. The language must allow to use limited logical and mathematical quantifiers that are analogs of loops in the rules.

3. The language must admit rules that are dependent on parameters (rule scheme). The scheme assigns a set of rules, i.e. it can be considered as an analog of a subprogram in the algorithmic language.

The language admits rules of two kinds:

(1) (prefix) $P(X) \rightarrow S_1(X_1) \& \dots \& S_k(X_k)$, where $P(X)$ is a formula, $S_1(X_1), \dots, S_k(X_k)$ are simple formulas, X, X_1, \dots, X_k are vectors of terms];

(2) (prefix) $P(X) \rightarrow \text{NameMod}(S_1, \dots, S_k)$ is a rule for module call, where $P(X)$ is a formula, $\text{NameMod}(S_1, \dots, S_k)$ is a module name, X is vector of terms, and S_1, \dots, S_k are arguments of the module.

The rule before the symbol \Leftarrow is a production antecedent, the rule after the symbol \Rightarrow is its consequent. The antecedent is a logical expression made up of relations, functional terms, atomic formulas, generalized formulas according the following rules.

Each rule must meet the main antedecent for variables: $V(\{S_1(X_1), \dots, S_n(X_n)\}) \cup V(P(X)) \subseteq V(\text{prefix})$, where $V(O)$ is a set of variables included into O (O can be any construction).

Prefix is a sequence of descriptions of variables $(v_1:t_1)(v_2:t_2)\dots(v_m:t_m)$ where $(v_i:t_i)$ is a description of a variable, v_i is a variable, t_i is a term for all $i=1,\dots,m$. Term t_1 does not contain free variables. For $i=2,\dots, m$ only variables v_1, v_2, \dots, v_{i-1} can be free variables of term t_i . The sequence of descriptions can be empty. All variables v_1, v_2, \dots, v_m are dually different.

The following construction can be called a scheme:

(3) $\text{NameSch}([\text{.CONST}]w_1, [\text{.CONST}]w_2, \dots, [\text{.CONST}]w_n)$: rule where

NameSch is a scheme name, w_1, w_2, \dots, w_n are variables, rule is of the kind (1). Variables w_1, w_2, \dots, w_n are formal parameters of the scheme. The scheme body is a rule and must contain formal parameters. $\llbracket \text{.CONST} \rrbracket$ means that $\llbracket \text{.CONST} \rrbracket$ can be absent.

The following construction can be called a scheme concretization:

(4) $\text{NameSch}(zw_1, zw_2, \dots, zw_n)$ where NameSch is a scheme name, zw_1, zw_2, \dots, zw_n are terms that are actual parameters of the scheme.

The scheme is an analog of a procedure in programming languages, the scheme concretization is an analog of a procedure call.

A domain symbol (a term of the domain ontology) – name n ; variable v ; sets I, R, S ; empty set \emptyset ; set $\{t_1, t_2, \dots, t_k\}$ where t_1, t_2, \dots, t_k are terms; intervals $I[t_1, t_2]$, $R[t_1, t_2]$ where t_1 and t_2 are terms; expression $t_1 \bowtie t_2$ where t_1 and t_2 are terms, sign $\bowtie \in \{+, -, *, /\}$, $t_1 \bowtie t_2$ where t_1 and t_2 are terms-sets, sign $\bowtie \in \{\cup, \cap, \setminus\}$ is a sign of operation on sets; $\mu(t)$ is a power of set where t is a term-set, $(Z_n(v : t_1) t_2)$ is a quantifier term where $Z_n \in \{+, *, \cup, \cap\}$, v is a variable (index of a quantifier term), t_1 is a term-set that assigns a range of v , t_2 is a term (the body of a quantifier term) that contains operation index v ; $f(t_1, \dots, t_k)$ is a functional term where f is a functional symbol (a term of the domain ontology), t_1, \dots, t_k are terms (arguments of functional term) are terms;

$p(t_1, \dots, t_k)$ or $\neg p(t_1, \dots, t_k)$ where $p(t_1, \dots, t_k)$ is an atomic formula, p is a predicate symbol (a term of the domain ontology), t_1, \dots, t_k are terms (arguments of atomic formula); $t_1 @ t_2$ where t_1 and t_2 are terms, sign $@$ is a sign of mathematical relation or relation on sets are simple formulas.

Logical expression $f_1 @ f_2$ where $@ \in \{\&, \vee\}$; quantifier formula $(Z_n(v : t) f)$ where $Z_n \in \{\&, \vee\}$, v is a variable (index of a quantifier formula), t is a term-set that assigns a range of v , f is a formula (the body of a quantifier formula) that contains index v are formulas.

Information Graph Definition

A language processor is a compiler that translates a text in the production language into the object code in the algorithmic high-level language. The object code implements the logical inference assigned by the production rules and contains calls of modules of the run period support environment. Before the code generation the language processor makes the program information graph and analyzes its characteristics.

An aligned cyclic graph the vertices of which are rules and arcs of which indicate information relations between rules, i.e. arcs connect those rules that exchange data, is called the information graph. The arc between two vertices exists if the following antecedent is met: $IF(\pi_j) \cap THEN(\pi_i) \neq \emptyset$ where $IF(\pi_j) = \{o_1', \dots, o_a'\}$ – a set of terms of a domain included in the antecedent of the rule π_j , $THEN(\pi_i) = \{o_1'', \dots, o_b''\}$ – a set of terms of a domain – arguments of the module called in the consequent of the rule π_i , or a set of terms of a domain included in the consequent of the rule π_i , $i \neq j$. The rule π_j will be called dependent on π_i . The information graph is created for each module. So, the information graph of a program is an array of information graphs of its modules.

Let us consider the structures used for representing the information graph of each module and different properties of a module that can be defined on basis of its graph. The element (i,j) of the incidence matrix IncMatrix with dimensions $\mu(\Pi_m) \times \mu(\Pi_m)$ where $\mu(\Pi_m)$ means the number of module rules is equal to 1 if the j -th rule is a direct child of the i -th rule, and is equal to 0 otherwise. LoopMatrix stores the information about the graph vertices that are included in a loop: elements correspondent to the rules not included in a loop are equal to 0, and included – 1. The element of an array iLoopAr with the number i is equal to -1, if the rule i is not included in any of the loops, is equal to 0 – the rule is included in one of loops but is not a loop entry; if the rule is a loop entry, the element is equal to the number of direct children of this rule. The number of direct children of the rule i is a value of each element with the number i of an array iParentsAr.

Parallelization of Rules Using Set of Active Rules

Since the language considered in the paper is confluent, the parallelization scheme where all the rules are sent one by one to slave processes and are executed in parallel for so long as they change the condition of the run-time environment is the most natural one. It is evident that when computations are organized in such a way, a set of empty executions of rules will be done because there are no input data for them at the initial moment of execution. One can use a set of active rules (SAR) to eliminate empty calls of rules. SAR is a set of rules which have input data at the moment of computation. Let us describe the parallelization scheme inside the module using SAR.

Let us introduce the designations. R_m – a set of all the rules of the program module, R' – a set of rules that are being executed by dependent processes; R'' – a set of rules that have been executed by dependent processes; FormSAR – a set of operations to create a set of rules that are being used at the current moment of execution; ComputeRule(π) – a set of operations to start dependent process to compute rule π ; GetResult(π, M) – a set of operations to receive computed data M of rule π from the executed process; Synchronize(M) – a set of operations to synchronize data received from the executed process with data stored in the main process; Compute(π, M) – a set of operations to compute rule π , i.e. to search for new values M objects included in the consequent of rule.

Let us describe the processes of parallelized logical inference by means of the designations.

Begin of main process;

FormSAR; $\Pi' = \emptyset$; $\Pi'' = \emptyset$;

While $SAR \neq \emptyset$ do:

select π from SAR;

$SAR = SAR \setminus \{\pi\}$;

ComputeRule(π);

$\Pi' = \Pi' \cup \{\pi\}$;

End of while;

While $\neg(MA\Pi = \emptyset \text{ и } \Pi' = \emptyset)$ do:

If $\Pi'' \neq \emptyset$ then

GetResult(π'' , M);

Synchronize(M);

FormSAR;

$\Pi'' = \Pi'' \setminus \{\pi''\}$;

end of if;

While $SAR \neq \emptyset$ do:

select π from $MA\Pi$;

ComputeRule(π);

$SAR = SAR \setminus \{\pi\}$;

$\Pi' = \Pi' \cup \{\pi\}$;

End of while;

End of while;

End of main process.

The dependent process begins to execute the rule during StartProcess(π) command of the main process.

Begin of dependent process;

Compute(π , M);

$\Pi'' = \Pi'' \cup \{\pi\}$;

$\Pi' = \Pi' \setminus \{\pi\}$;

End of dependent process.

Let us comment on the algorithms. In the beginning, a set of active rules is formed by the main process. The set contains the rules for all the objects of which included in the rule condition in the input data are specified. Then in the loop each rule from SAR is sent for computation to the dependent process. The main process waits for the execution result of at least one of the dependent process. After that, the results of the rule execution are synchronized with the current values of data and SAR is updated: this set is completed with the rules or the objects of which new values appear. If SAR is not empty, rules from SAR are sent to free slave process for computation. The program ends when there are no dependent process executing rules and SAR is empty.

As opposed to the common scheme or parallelization, this scheme solves the problem of execution of empty rules. However, it has a drawback: the system incurs additional expenses due to forming of SAR during computations.

Using the Information Graph at the Parallelization of Logical Inference

This paper suggests using the "client-server" architecture when a separate process is a dispatcher (main process), other processes are handling processes (dependent processes) for constructing a parallel production system. The main process inputs and outputs data synchronizes them and exchanges data with dependent processes; it prioritizes rules and provides each process with a subprogram to process a rule. Each dependent process executes a subprogram that implements the logical inference for the rule, i.e. it searches for all substitutions at which the condition of the rule applicability is true and for each substitution it performs actions defined by the consequent of the rule and passes the received data to other processes. Below there are schemes of the workflow of the main process and dependent process.

Before the main process starts to work, $iCurParentsAr$ – a copy of $iParentsAr$ – is created, at the same time if the information graph contains loops, we must change values of elements of the array $iCurParentsAr$ in the following way: if $iLoopAr[i] > 1$, then $iCurParentsAr[i] = iLoopAr[i]$, i.e. substitute rules-loop entries for the number of parents that do not belong to loops. Elements of the array $iCurParentsAr$ change their values during the calculations. The array element i gets equal to -1 if the rule i is being processed, -2 – if the rule i has been processed.

Scheme 1a (main process):

Calculations Begin: $\mu(P_i) = \mu(P)$; $P_w = \emptyset$. Block 1.

LOOP: While exist $iCurParentsAr[i] = 0$ or $P_w \neq \emptyset$, do: Block 2; Block 3. Loop End.

Block 4. Calculations End.

Block 1 (Start all rules which appropriate to root vertices):

For every free process j from P_f do:

For every element i from array $iCurParentsAr$: If $iCurParentsAr[i] = 0$,
then Send(Q_i, i, j); $iCurParentsAr[i] = -1$; $P_f = P_f \setminus \{j\}$; $P_w = P_w \cup \{j\}$.

Block 1 End.

Here Send(Q_i, i, j) is a procedure that sends data set Q_i into process j and informs process j of the necessity to calculate rule i ; $\mu(P)$ is the number of slave processes for calculations; $\mu(P_i)$ is the number of free processes. Data set Q_i contains all the values of the objects included in the condition of rule i .

Block 2 (Receive and synchronize results):

1. Recv(Z, i, j); $P_w = P_w \setminus \{j\}$; $P_f = P_f \cup \{j\}$.

2. $iCurParentsAr[i] = -2$.

3. For all vertices k such as

$iCurParentsAr[k] > 0$, do:

3.1. If IncMatrix[i, k] = 1,

then $iCurParentsAr[k] = iCurParentsAr[k] - 1$;

3.2. If $iLoopAr[i] > 0$ & $iCurParentsAr[k] = -2$ & $iLoopAr[k] > 0$ & THEN(i) \cap IF(k) $\neq \emptyset$

then $iChangedLoopAr[k] = 1$.

4. Data = Data \cup Z;

Block 2 End.

Here Recv(Z, i, j) is a procedure that receives from process j data set Z that are results of computing rule i . Data set Z contains values of the objects included in the consequent of rule i . Data is a data set that contains all the values of all the objects included in the rules.

Block 3 (Assign rules ready for computing to free processes):

For every free process j do:

For every element i from array iCurParentsAr:

If iCurParentsAr[i] = 0

then Send(Q_i, i, j); iCurParentsAr[i] = -1.

If $P_w = \emptyset$ & not exist elements k from array iCurParentsAr such as iCurParentsAr[k] = 0 then:

For every element t from array iCurParentsAr:

If iCurParentsAr[t] = -2 & iLoopAr[t] > 0 then:

For all vertices s :

If LoopMatrix[t, s] = 1 & iCurParentsAr[s] > 0

then iCurParentsAr[s] = 0;

Goto Block 3.

Block 3 End.

Block 4 (Make rules which appropriate to loop vertices ready for repeated calculations):

If exist elements i from array iChangedLoopAr such as iChangedLoopAr[i] = 1 then:

For all rules do:

If k – (distant) child of rule i

then iChangedLoopAr[k] = 1.

For every element t from array iChangedLoopAr:

If iChangedLoopAr[t] = 1

then iCurParentsAr[t] = iParentsAr[t];

If iLoopAr[t] > 1 then iCurParentsAr[t] = 0.

Goto Block 3.

else:

Block 4 End.

Here iChangedLoopAr is an integer array where the element with number i is equal to 1 if loop rule i has been computed and then appear new values for the objects included in its antecedent; otherwise the element with number i is equal to 0. This structure is filled in the course of computing the rules and is used to construct a children list, the children must be computed again.

Scheme 1b (slave process):

Calculations Begin: wRecv(Z, i); wCalc(i, Z, Q); wSend(Q, i); Calculations End

Here wRecv(Z, i) is a procedure that receives from the main process data set Z that contains values of the objects included in the antecedent of rule i ; wSend(Q, i) is a procedure that sends into the main process data set Q that are the results of computing rule i ; wCalc(i, Z, Q) is a procedure that computes rule i with the help of the logical inference.

Tuple Passing at Incomplete Rule Computation

The previous scheme rigidly specifies that the dependent rule cannot be computed until the rules it is dependent on have been computed. This scheme does not have such a restriction – the next rule waits for at least one tuple – the result of the application of the rule it depends on but not the termination of all the rules-parents. If each next tuple appears at the beginning of the rule application, there can be a situation when all the rules are processed parallel regardless of how they are connected informationally. However, due to the restrictions connected with the

number of processes of cluster computer free for computation, the number of rules that work parallel cannot exceed the number of free processes.

Let us complete the above schemes with a series of new operations that will allow to load free processes with those rules the only parents of which are being computed.

Let $iParentsIdAr$ be an integer array the dimensions of which coincide with the number of module rules, the element with number i being equal to the number of the parent if vertex i has the only parent. $SendPFrom(p_{from}, Q, i, j)$ is a procedure that sends data set Q in process j and informs process j of the necessity to calculate rule i , process j must receive from process p_{from} a set of next tuples for the objects included in the antecedent of rule i . Data set Q contains all the values of the objects included in the antecedent of rule i . $SendPTo(p_{to}, p_i)$ is a procedure that sends in process p_i that applies rule i the message about the necessity to send to process p_{to} tuples for those objects included in the antecedent of rule processed by p_{to} .

Block X1 (Assign additional rules to calculations using tuples passing):

For every free process j from P_f do:

For every element i from array $iCurParentsAr$:

If $iCurParentsAr[i] = -1$ then

For every element k from array $iParentsIdAr$:

If $iParentsIdAr[k] = i$ then

$iParentsIdAr[j] = -1$;

$SendPFrom(p_{from}, Q, k, j)$;

$iCurParentsAr[k] = -1$; $P_f = P_f \setminus \{j\}$; $P_w = P_w \cup \{j\}$.

$SendPTo(p_{to}, p_i)$;

Block X1 End.

Scheme 2a (main process):

Calculations Begin:

$\mu(P_f) = \mu(P)$; $P_w = \emptyset$.

Block 1.

Block X1.

LOOP: While exist $iCurParentsAr[i] = 0$ or $P_w \neq \emptyset$,
do:

Block 2.

Block 3.

Block X1.

LOOP End.

Block 4.

Calculations End.

Scheme 2b (slave process):

Calculations Begin:

$flag = 0$; $p_{to_} = 0$;

$wRecv_ (p_{from}, Z, i)$;

If $p_{from} > 0$ then $flag = 1$;

Block 1.

If $flag = 1$ then:

$wRecvPFrom(p_{from}, Z_i)$;

If $Z_i \neq \emptyset$ then $Z = Z \cup Z_i$;

else $flag = 0$;

$wCalc(i, Z, Q)$;

If $wRecvPTo(p_{to}) = 1$ then $p_{to_} = p_{to}$;

If $p_{to_} > 0$ then $wSend_ (p_{to_}, Q)$;

Block 1 End.

If $flag = 1$ then Goto Block 1.

$wSend(Q, i)$;

Calculations End.

Block X1 loads all the free processes with the rules that can receive tuples from their computed parents and informs the processes that compute parents of the necessity to pass tuples to other processes.

The scheme of the workflow of the dependent process is a modified scheme 1b. To describe it we use the following procedures.

$wRecv_ (p_{from}, Z, i)$ is a procedure that is an extended version of procedure $wRecv$. $wRecv_$ receives from the main process data set Z that contains the values of the objects included in the antecedent of rule i and receives the number of process p_{from} from where next tuples can come. If $p_{from} = 0$, tuples from other processes will not be sent. $Z \equiv \{O_1, \dots, O_z\} \equiv \{\{k^1_1, \dots, k^1_{k1}\}, \dots, \{k^z_1, \dots, k^z_{kz}\}\}$.

$wRecvPFrom(p_{from}, Z_i)$ is a procedure that receives from slave process p_{from} data set Z that contains the value of the objects included in the antecedent of rule i . $Z_i \equiv \{O_1, \dots, O_z\} \equiv \{\{k^1_1, \dots, k^1_{k1}\}, \dots, \{k^z_1, \dots, k^z_{kz}\}\}$.

$wRecvPTo(p_{to})$ is a function that receives from the main process the number of slave process p_{to} to which it is necessary to send tuples. The function returns 0 if the number of the process from the main process has not been received, and it returns 1 if it has.

$wSend_ (p_{to}, Q)$ is a procedure that sends to slave process p_{to} data set Q that is the current result of the computed rule i . Data set Q contains all the values of the objects included in the consequent of rule i . $Q \equiv \{O_1, \dots, O_q\} \equiv \{\{k^1_1, \dots, k^1_{k1}\}, \dots, \{k^q_1, \dots, k^q_{kq}\}\}$.

Applying this scheme we can launch in parallel the process that will process the rule dependent on data not when the rule-parent has been performed, but when attributions for the objects found in the course of calculations start to come. Thus, if the dependence on data allows, one can launch all the rules in parallel as the correspondent attributions appear. This implies that the period of applying all the rules can be shorter than the period of applying the rules using the first scheme.

Parallelization of Computations for Rules with Quantifiers

The scheme considered below makes it possible to parallelize computations for a single rule with quantifier. The body of rule with quantifier must be executed as many times as many values the index of quantifier construction receives, i.e. rule with quantifier defines the loop the body of which is the rule body. Therefore rules with quantifiers can be easily parallelized.

The parallelization scheme assumes the following actions. As soon as rule with quantifier becomes accessible for computations, the main process organizes loop on all values of the quantifier index. At each step of the loop, rule with a certain current value of the index is sent to the free process for computation. Each new step of the loop sends the rule to execution in parallel with previous steps as copies of the rules do not contain the dependent data. The results of computations of each copy of the rule are processed by the main process as it is shown on scheme 2.

Let us consider an example of how the scheme works. It is as follows:

$$\langle \langle i, 3, 10 \rangle \rangle a(i) \& b(v) \& v > 10 \rightarrow c(i+v) \>.$$

i variable is the index, the low bound is specified with 3, the upper bound is specified with 10. The body of the rule $a(i) \& b(v) \& v > 10 \rightarrow c(i+v)$.

Using this scheme, we will have a set of 8 rules:

$$a(3) \& b(v) \& v > 10 \rightarrow c(3+v);$$

$$a(4) \& b(v) \& v > 10 \rightarrow c(4+v);$$

...

$$a(10) \& b(v) \& v > 10 \rightarrow c(10+v).$$

Each copy of the rule with a new value of the index is launched if there is a free slave process. If at the initial moment of the execution there are 8 or more free processes, all the copies of the rules can be computed concurrently.

Let there be n free slave processes available for the system at the moment of computation of the rule with quantifier. If the quantifier index receives m values, where $m \geq n$, using the scheme will reduce the execution time of T rule down to n times and down to m times if $m < n$.

Parallelization of Computations through Partitioning Object Value Area inside Rule

This scheme describes parallelization of partitioning rule computation through partitioning area of existing values of objects included in the rule condition and sending each subarea to the correspondent dependent process where searching for the result will take place. If there are free dependent processes, the main process can divide object value area into subareas the number of which equals the number of free processes. In this case, each process receives a copy of the rule with the correspondent subarea of object values. As subareas do not intersect, all copies of the rule can be computed concurrently.

Let there be n items of o objects. For each o_i object, there are a set of tuples of values of ar arguments in quantity of m_i .

$$\begin{array}{lll} ar_1(o_1) & ar_1(o_2) \dots & ar_1(o_n) \\ ar_2(o_1) & ar_2(o_2) \dots & ar_2(o_n) \\ \dots & \dots & \dots \\ ar_{m1}(o_1) & ar_{m2}(o_2) \dots & ar_{mn}(o_n) \end{array}$$

In the process of searching for values satisfying the rule condition a selection and conditional test in a set of Args of all the values:

$$\begin{array}{lll} ar_1(o_1) & ar_1(o_2) & \dots & ar_1(o_n), \\ ar_1(o_1) & ar_1(o_2) & \dots & ar_2(o_n), \\ ar_1(o_1) & ar_1(o_2) & \dots & ar_3(o_n), \\ \dots & \dots & \dots & \\ ar_1(o_1) & ar_2(o_2) & \dots & ar_1(o_n), \\ ar_1(o_1) & ar_2(o_2) & \dots & ar_2(o_n), \\ ar_1(o_1) & ar_2(o_2) & \dots & ar_3(o_n) \end{array}$$

etc. Evidently, each string of values can be tested for compatibility with the rule condition in parallel. This, we can partition a set of object values into nonintersecting subsets:

$$Args = Arg_1 \cup Arg_2 \cup \dots \cup Arg_k.$$

$$Arg_1 \cap Arg_2 \cap \dots \cap Arg_k = \emptyset.$$

Let us exemplify:

$$R_1: a(v_1, v_2) \& b(v_3, v_4) \& condition(v_1) \& condition(v_2) \& condition(v_3) \& condition(v_4) \rightarrow c(v_1+v_3);$$

For each object of R_1 rule, there is a set of values-tuples:

Object «a»		Object «b»	
Arg. 1	Arg. 2	Arg. 1	Arg. 2
a1[1]	a2[1]	b1[1]	b2[1]
a1[2]	a2[2]	b1[2]	b2[2]
...
a1[n]	a2[n]	b1[m]	b2[m]

The scheme of search for all the tuples of objects satisfying the condition of R_1 rule is generally as follows (for execution process):

Begin;

For $i = 1$ to n do:

if condition($a1[i]$) = «true» and condition($a2[i]$) = «true» then

For $j = 1$ to m do:

If condition ($b1[j]$) = «true» and condition($b2[j]$) = «true» then

MakeCalculation($a1[i]$, $a2[i]$, $b1[j]$, $b2[j]$);

If end;

Loop end;

If end;

Loop end;

End;

It is clear that there is a bypass of the tuples of objects by means of nested loops. To parallelize this rule processing, we can partition the search space for tuples for the first object according to the number of free processes. In other words, if we have k of processes available for usage in rule computation, we send the 1st-(n/k)th tuples of the first object to the first process, the ($n/k + 1$)th-($2 \cdot n/k$)th tuples of the first object to the second process, (($n-1$) $\cdot n/k + 1$)th- n th tuples of the first object to the k^{th} process correspondingly.

The example for the t^{th} process ($1 \leq t \leq k$) is as follows:

Begin;

For $i = 1$ to $((t-1) \cdot n/k + 1)$ do:

If condition($a1[i]$) = «true» and condition($a2[i]$) = «true» then

For $j = 1$ to m do:

If condition($b1[j]$) = «true» and condition($b2[j]$) = «true» then

MakeCalculation($a1[i]$, $a2[i]$, $b1[j]$, $b2[j]$);

If end;

Loop end;

If end;

Loop end;

End;

The main process distributes tuples of the first object and sends a copy of the correspondent tuples to the correspondent process.

Method of Control of Scheme Choice

Let us describe the method of control of parallelization scheme choice and consider constraints imposed by the environment of program execution and structure of the information graph (IG).

Let us have $\mu(R_m)$ number of rules in m module. It has P_{opt} defined provided that the rule is executed completely and the number of free slave process equals $\mu(S)$. In real tasks, IG can have up to 100 rules and, at the same time, the branchiness can be very low. On the other hand, real clusters provide not more than a few dozens of processes. In this case, $\mu(S) \ll \mu(R_m)$ but as we are unable to compute more than P_{opt} of rules concurrently, it will be more correct to consider the relation of P_{opt} and $\mu(S)$ numbers. If $P_{opt} < \mu(S)$, there are idle slave process during all the computations. On the other hand, if $P_{opt} > \mu(S)$, there are no idle processes at some points of time. It is clear that graphs have different branchiness at different levels. Hence at one point there are all free slave processes operating, at another one there is only one process with others being idle for a long period of time.

Using the method of sending tuples to processes, the system must have at its disposal the number of slave processes equaling the number of rules while real clusters provide not more than a few dozens of processes.

Additionally, there appear increase expenses for sending tuples to processes and data synchronization system.

We propose to combine two methods of rule parallelization and thus the system of logical inference must use the maximum number of free processes at each point of time. To reach this, the system assigns all ready for execution rules from IG for computation. If there are idle slave processes after that, the system sends computation of rules-children parents of which are being processed. At the same time, there is an exchange of newly computed tuples among processes processing a group of parent-child rules.

Algorithm 1 of rule computation in the main process is given below. The information graph is designed for m module. It has $\mu(R_m)$ number of rules and $\mu(S)$ number of free slave processes. Note that the execution time for each vertex-rule is not known. Consequently, assigning rule ready for execution to a free slave process can only be done dynamically during the logical inference system operation.

1. All IG rule-vertices ready for execution are found. If in the consequent of a rule ready for execution there is a module call, launch of all new rules stops, the main process waits for the results of execution of all current rules and then passes control to the main process of the called module. Transfer to Step 1 of the called module. If in the consequent of a rule ready for execution there is no module call, it is passed to a free process from a set $\mu(S)$ for computation. If there are no free processes after that, transfer to Step 4, otherwise – Step 2.
2. Rule-vertices where their only parent is the vertex that is being processed are chosen from IG. Each found rule is passed to a free slave process for computation; this process can receive resulting tuples from the processes processing the rule-parent. If there are no free processes after that, transfer to Step 4, otherwise – Step 3.
3. If a rule ready for execution contains a quantifier, for each quantifier index there is a copy of the rule with the substituted index value/ the copy is sent to a slave process. All copies of the rule are launched concurrently.

4. If a process has finished the rule computations and sent the results, the main process accepts all resulting tuples and synchronizes them with its data. If all the rules are computed and all slave process are free, transfer to Step 4, otherwise – Step 1.
5. If in IG there are vertices included in loops and, as a result of computations of rules corresponding to these vertices, new values of objects which are included in conditions of loop rules, a subgraph is designed. It contains all the rules which are children of these loop rules. IG is substituted with the designed subgraph and computation starts from Step 1 again. If there are no loops in IG and no values after loop computations, transfer to Step 5.
6. If the module the rules of which were computed at the previous steps has been called from another module, the main process passes control to the process it was called by. Transfer to Step 1. If it is the main module, the computation ends.

Algorithm 2 of a slave process.

1. A slave process accepts the number of a rule and input data from the main process and starts computation.
2. If in the process of computation there is a command to accept tuples from another process, the given process accepts these tuples and synchronizes them with its data.
3. If in the process of computation there is a command to send the computed tuples to another process, all the computed tuples are sent to that process.
4. After the rule computation, all resulting data are sent to the main process.

The algorithms show that all the rules are executed once if there are no loops in the information graph. If there are loops in IG and for one of the computed rules included in the loop there are new values, such rule along with all the rules that are children of the given one is executed again as long as new values appear.

The slave process algorithm describes the computation of the result of the processing of the rule which is search for attributions satisfying the rule condition and execution of the rule consequent. If in the rule consequent there is a module, the main process stops executing the rules of the current module and begins to process the called module according to Algorithm 1.

The search for attributions is search for attributions for all the objects of the rule condition that is passing through the nested loops. Generally, let there be finite sets of values of M , M_1 , M_2 objects and function $f: M_1 \times M_2 \rightarrow M$. Let us consider the problem of computation of over the whole definition range. This problem is solved by means of the following algorithm.

Begin:

For all elements from M_1 do:

Get $x \in M_1$;

For all elements from M_2 do:

Get $y \in M_2$;

Calculate $f(x, y)$;

Loop end;

Loop end;

End.

A loop is used to design computations for each rule to search for values of each object included in the rule condition. Each loop of a new object is embedded into the loop of the previous object in the rule condition. Thus, already found values are excluded from the search range.

In this case, to avoid rescanning of the value range during the recomputation of a certain rule information about what sections of the value range have been scanned before is connected with each rule, i.e. for each rule each set of values is divided into two parts: 'new' and 'old.' If we designate scanning through a new part of the value range of a function or a relation named f_{ri} as $N(f_{ri})$, through the old part – $O(f_{ri})$, through the whole value range – $A(f_{ri})$, Cartesian product set – as $*$, union of sets – $+$, then all the new acceptable values of λ substitution can be resulted from scanning the following set:

$$\begin{aligned} & N(fr1) * A(fr2) * A(fr3) * \dots * A(fr_k) \\ & + O(fr1) * N(fr2) * A(fr3) * A(fr4) * \dots * A(fr_k) \\ & + O(fr1) * O(fr2) * N(fr3) * A(fr4) * \dots * A(fr_k) \\ & + \dots + O(fr1) * O(fr2) * \dots * O(fr_{k-1}) * N(fr_k). \end{aligned}$$

Thus, during the recomputation of loop rules and rules dependent on them, the logical inference system searches for values only for a new range. Labels are stored in special structures. If the rule is a loop one, in case of its recomputation its labels return to zero.

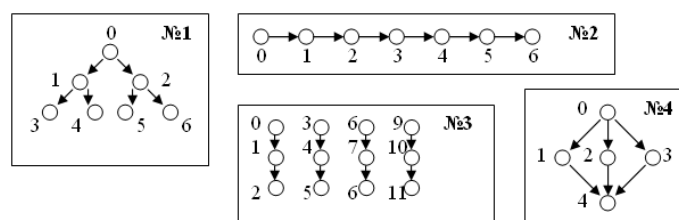
Experimental Study of Parallel Programming System Properties

Some experiments were conducted to evaluate the efficiency of production system operation with parallelization schemes. Each experiment included time measurement for generated programs for cluster with different sets of data. The experiments were aimed at studying the dependency of program execution time on the number of connections among rules and the number of free processes. In the description of the experiments there is average period of program execution time.

Each logical program consists of a set of rules which are connected with each other via data. The experiments cover the programs the connections inside which can be the most frequent. We can assume that all the rest programs are a composition of exemplified programs.

The experiments were conducted on a multiprocessor computer system with 12 processors free for computations. Each processor was given one process. Client-server architecture of the program executing the logical inference requires at least 2 free processes one of which is meant for data control and synchronization; the rest are meant for computations of rules per se. If there are 2 free processes, rules are processed consequently one by one.

To exemplify, we chose the programs the information graphs of which are as follows:



The execution time of each program is shown in Fig. 1. Example 1 demonstrates the maximum execution time of 30.5 sec. with 2 processes (one of them is for control functions, the others – for rule execution), the minimum execution time of 13.2 sec. With 4 and more processes the efficiency grew 2.3 times. Example 2 demonstrates

the maximum execution time of 30.5 sec. with 2 processes, the minimum execution time of 5.3 sec. With 8 and more processes the efficiency grew 5.7 times. Example 3 demonstrates the maximum execution time of 52.5 sec. with 2 processes, the minimum execution time of 8.3 sec. With 12 and more processes the efficiency grew 6.3 times. Example 4 demonstrates the maximum execution time of 21.8 sec. with 2 processes, the minimum execution time of 13.2 sec. With 3 and more processes the efficiency grew 1.6 times.

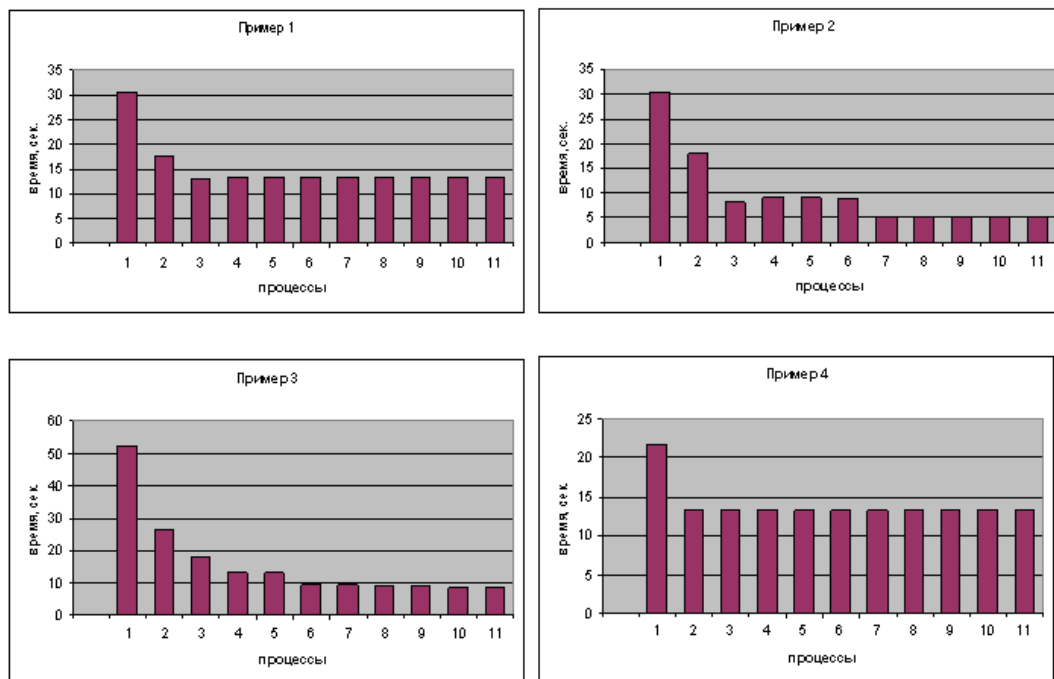


Fig. 1. The dependency of program execution time on the number of connections among rules and the number of free processes.

Apart from this, there were experiments on actual problems, one of which is the problem of organic synthesis. The dependency of the time for solution of this problem on the number of processes is shown in Fig. 2.

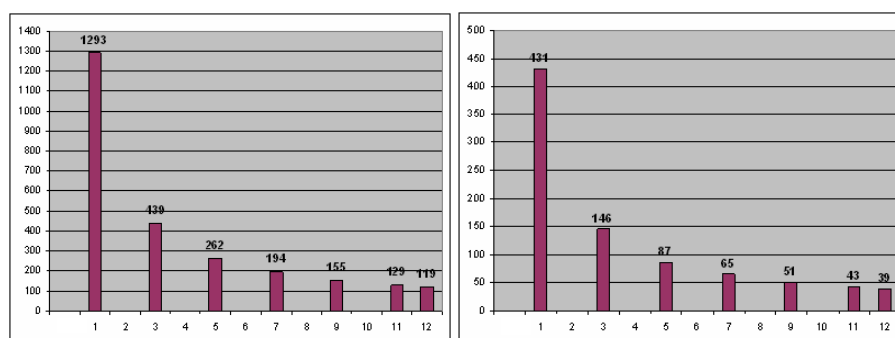


Fig. 2. The dependency of the execution time on the number of processes for the problem of organic synthesis.

The left diagram demonstrates the execution time on one processor of 1293 sec., on 12 processors – 119 sec., the efficiency is 10.8 times. The right diagram presents the program the volume of data of which is three times

less than for the previous program. The maximum execution time – 434 sec., the minimum execution time – 39 sec., the efficiency – 11 times.

Conclusion

The above schemes use such characteristics of the information graph as the number of its vertices, the number of its branches that can be processed in parallel, etc. To choose a scheme of parallelization of the logical inference not only the characteristics of the graph but also architecture and system constraints imposed by the computing environment must be used. They are:

1. The number of free system processes. If the number of the processes is more or equal to the number of the rules of the program, this case is convenient for calculations as one can specify in advance the particular rule for each process. If there are less processes than rules, then the rules are assigned to the processes dynamically.
2. The period of the rule application. In the course of computing a rule there may be a situation when this rule is processed many times longer than any other rule of the logical program. In this case there may be an idle time of the system that awaits the end of calculations of this rule. One of the solutions is to send intermediate results of the rule calculation to other process that process dependent rules.
3. The structure of the program information graph that is assigned by a set of information graphs of modules that are part of the program. The graph characteristics define the number of graph sections that can be executed in parallel. One has to analyze the graph to assign a rule for a free dependent process. The more branches are there in the graph, the stronger is the possibility of parallelizing calculations of rules. As one does not know the exact time of computing a rule, the maximum number of processes P_{opt} that can be performed in parallel with each other is defined in the following. For each vertex of the graph they build a set of vertices into which there are no ways from the given vertex and which are not parents (and far parent as well) of this given vertex. After computing the maximum from the number of the elements of all the sets, one will have the sought P_{opt} .

The closer is P_{opt} to the number of the rules in module $\mu(\Pi_m)$, the more efficient will be the parallelization of the task and quicker will be the calculations, i.e. $E = P_{opt} / \mu(\Pi_m)$ – tends to 1 (E defines the average fraction of the rule calculation by a separate process). From the definition P_{opt} it follows that P_{opt} will be bigger if the graph has more direct children for one parent, i.e. there is wide branchiness of the graph.

For each graph P_{opt} is invariable if the rules are executed completely: first – parents, then – children. However, one can start to pass intermediate results to the dependent rules without awaiting the completion of one rule. In this case if in the rule-parent there is at least one tuple of new values for the object included in the antecedent of the rule-child, the process that maintains the rule-parent sends the tuple to the process assigned to process the dependent rule. Doing this with all the rules and assigning a rule for a process, the system of the logical inference can assign all the rules for execution in parallel with each other. It implies that P_{opt} is always equal to the number of rules $\mu(\Pi_m)$ (and $\mu(P)$ must be equal to $\mu(\Pi_m)$) in module m , $E = 1$.

It is evident from the experiments that execution time of most programs on multiprocessor computer systems is close to theory.

Bibliography

- [1] Gavrilova T.A., Khoroshevsky V.F. Intellectual System Knowledge Bases. (In Russian) – SPb.: Piter, 2001
- [2]. Kleshchev A.S., Artemjeva I.L. Mathematical models of domain ontologies // Int. Journal on Inf. Theories and Appl., 2007, vol 14, № 1. PP. 35-43.
- [3]. Berkeley UPC - Unified Parallel C. <http://upc.lbl.gov/>.

-
-
- [4]. David E. Culler, Andrea Dusseau. Parallel Programming in Split-C. Computer Science Division University of California, Berkeley. 1993. www.eecs.berkeley.edu/~yelick/arvindk/splitc-super93.ps.
- [5]. Ken Kennedy (director), Vikram Adve. Fortran Parallel Programming Systems // Parallel Computing Research Newsletter. 1994, vol. 2, issue 2. <http://www.crpc.rice.edu/newsletters/apr94/resfocus.html>.
- [6]. Modula-3 resource page. <http://www.modula3.org>.
- [7]. Carl Scarbnick. Building your own parallel system with PVM. <http://www.sdsc.edu/GatherScatter/gsnov92/PVMparallel.html>.
- [8]. The Message Passing Interface (MPI) standard. <http://www.mcs.anl.gov/research/projects/mpi>.
- [9]. Message passing with MPL. <http://math.nist.gov/~KRemington/Primer/mp-mpl.html>.
- [10]. OpenMP. <http://openmp.org/wp>.
- [11]. ShMem. http://www.fis.unipr.it/lca/tutorial/hpvm/hpvm.doc_83.html#SEC90.
- [12]. BERT 77. <http://www.basement-supercomputing.com/content/view/25/52/>
- [13]. The PIPS Workbench Project. <http://www.cri.enscm.fr/~pips/>.
- [14]. VAST/Parallel. Crescent Bay Software. http://www.crescentbaysoftware.com/vast_parallel.html.
- [15]. Douglas Kothe, Ricky Kendall. Computational science requirements for leadership computing. 2007. www.nccs.gov/wp-content/media/nccs_reports/ORNLTM-2007_44.pdf.
- [16]. Paralogic Inc. <http://www.plogic.com/index.html>.

Authors' Information

Irene L. Artemieva – artemeva@iacp.dvo.ru

Michael B. Tyutyunnik – michaelhuman@gmail.com

*Institute for Automation & Control Processes, Far Eastern Branch of the Russian Academy of Sciences;
5 Radio Street, Vladivostok, Russia*

WEBLOG CLUSTERING IN MULTILINEAR ALGEBRA PERSPECTIVE

Andri Mirzal

Abstract: This paper describes a clustering method for labeled link network (semantic graph) that can be used to group important nodes (highly connected nodes) along with their relevant link's labels by using a technique borrowed from multilinear algebra known as PARAFAC tensor decomposition. In this kind of network, the adjacency matrix can not be used to fully describe all information about the network structure. We have to expand the matrix into 3-way adjacency tensor, so that not only the information about to which nodes a node connects to but by which link's labels is also included. And by applying PARAFAC decomposition, we get two lists, nodes and link's labels with scores attached to them for each decomposition group. So clustering process to get the important nodes along with their relevant labels can be done simply by sorting the lists in decreasing order. To test the method, we construct labeled link network by using blog's dataset, where the blogs are the nodes and labeled links are the shared words among them. The similarity measures between the results and standard measures look promising, especially for two most important tasks, finding the most relevant words to blogs query and finding the most similar blogs to blogs query, about 0.87.

Keywords: Blogs, Clustering Method, Labeled-link Network, PARAFAC Decomposition.

ACM Classification Keywords: I.7.1 Document management

1. Introduction

The researches on network clustering have a long tradition in computer science, especially on neighborhood-based network clustering, where the nodes being grouped together if they are in the vicinity and have a higher-than-average density of links connecting them [1]. Some real examples of these are in parallel computing and distributed computation where n number of tasks is divided into several processes that carried out by a separate program or thread running on one of m different processors [2].

In addition to the neighborhood-based network clustering, there is another clustering method that works on labeled link network; the nodes are in the same group if they share the same or almost the same link's labels. In online auction networks this method can be used to find similar users [3], and by utilizing user's preferences in buying and selling activities, a recommendation system for effective and efficient advertisements can be proposed [4].

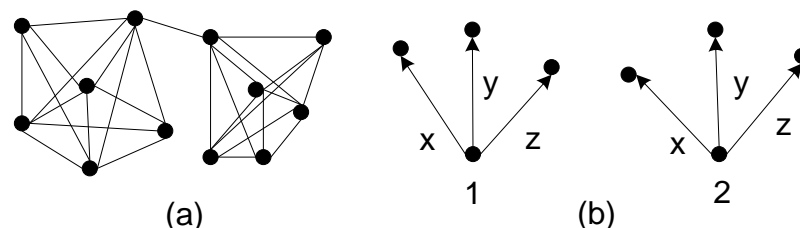


Fig. 1. This figure shows the concept of clustering based on neighborhood (a), and based on link's labels (b). In (a), there are two clusters which well connected within cluster and only has one link between clusters. And in (b) node 1 and 2 are in the same group due to the similarity in their link's labels even though they are not connected at all.

These two clustering methods, however, can only group nodes. When the challenge to group the nodes with their most relevant link's labels comes, we have to utilize characteristic matrix (also known as incidence matrix), a nodes-versus-link's labels matrix, where the entries are the weights of the links. The further discussion about this can be found in [4]. But there are some situations where this approach is not suitable to be used, for example in the situation where we are only interested in finding the relevant link's labels for the most important nodes, nodes that have many links (in web network this will be pages with many inlinks, but other networks like online auction network and international trading network, this can be nodes with many inlinks and/or outlinks [4]). This task is not trivial, for example in the web, the famous web pages more likely to attract many viewers, so that the ability to group the important pages with their relevant anchor texts has real advantage [5].

To provide the natural way of grouping important nodes along with their relevant labeled links; first constructing adjacency tensor of the labeled link network, where the first and second axis are the nodes and the third axis is the link's labels (an example shown in fig. 2). And then apply tensor decomposition to get the node's authority and hub vectors along with link's labels' vectors for each R decomposition groups (see eq. (1) and fig. 3). And because the results produce ranking scores for both nodes and link's labels, we can sort these vectors in descending order to get clustering of important nodes and their relevant labeled links for each R decomposition groups.

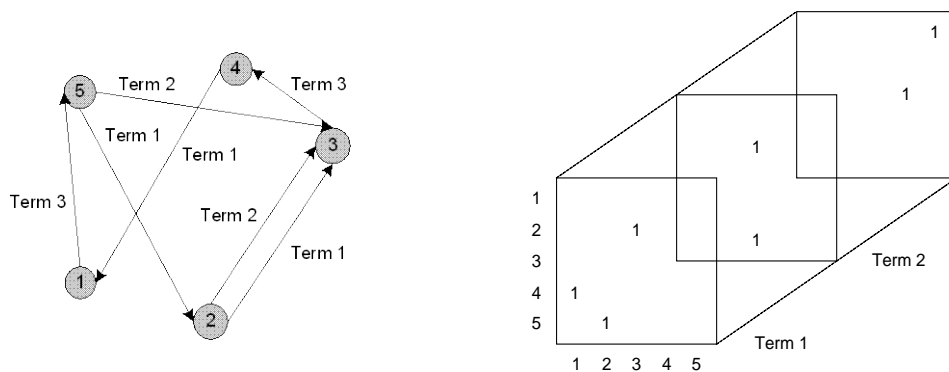


Fig. 2. (a) The labeled link network with five nodes and three terms, and (b) its adjacency tensor.

In this paper, blogs dataset is used to form labeled link network by taking blogs as the nodes and shared words as the link's labels. This work has some possible real applications. For example because the similarity measures for task 1, finding the most relevant blogs to the words query and task 3, finding the most similar blogs to the blogs query, give promising results (see table 4 in section V), one can build a blog search engine that provides users with ability to find the most relevant blogs to the words query or to find the most similar blogs to the blogs query. And because the results provide us with blog-word grouping, the engine can display not only the blogs but also the relevant words. Also one can also build visualization service that shows the blogs with its relevant shared words.

2. PARAFAC Tensor Decomposition

The PARAFAC tensor decomposition is higher-order analogue to the matrix singular value decomposition (SVD), but the singular vectors produced by PARAFAC are not generally orthonormal as the case in SVD [6].

The PARAFAC decomposition approximates a tensor by the sum of R rank-1 outer products of vectors \mathbf{h}_r , \mathbf{a}_r , and \mathbf{t}_r as shown in fig. 3. Vector \mathbf{h}_r is the corresponding hub vectors, \mathbf{a}_r is the corresponding authority vectors, and \mathbf{t}_r is the corresponding term vectors for each rank r . PARAFAC decomposition of the adjacency tensor \mathbf{X} can be written as:

$$\mathbf{X} \approx \lambda \langle \mathbf{H}, \mathbf{A}, \mathbf{T} \rangle \equiv \sum_{r=1}^R \lambda_r \mathbf{h}_r \circ \mathbf{a}_r \circ \mathbf{t}_r \quad (1)$$

where \mathbf{H} , \mathbf{A} , and \mathbf{T} is the hub, authority and term matrices of R rank-1 tensor \mathbf{X} decomposition, \circ is outer vectors product, and λ_r ($\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_R$) is the weight for each group r . \mathbf{H} , \mathbf{A} and \mathbf{T} are formed by arranging vectors \mathbf{h}_r , \mathbf{a}_r and \mathbf{t}_r such that:

$$\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \dots \quad \mathbf{h}_r], \quad \mathbf{A} = [\mathbf{a}_1 \quad \mathbf{a}_2 \quad \dots \quad \mathbf{a}_r], \quad \text{and} \quad \mathbf{T} = [\mathbf{t}_1 \quad \mathbf{t}_2 \quad \dots \quad \mathbf{t}_r] \quad (2)$$

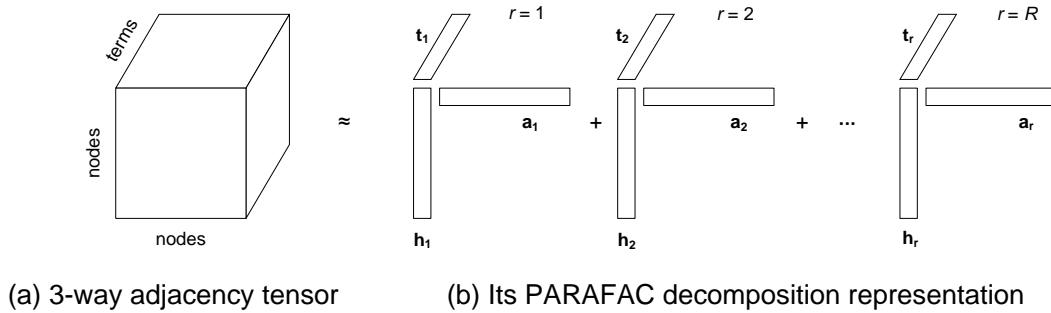


Fig. 3. (a) Network's adjacency tensor, and (b) its R rank-1 PARAFAC decomposition.

To calculate PARAFAC decomposition, greedy PARAFAC algorithm is used [5].

3. Data and Preprocessing

We downloaded the blogs and its contents from technorati's most popular and favorite blogs lists (<http://technorati.com/pop/blogs/>) on November 27th, 2007. The number of blogs is 151 and the number of shared words is 704 (180 words after irrelevant words, like stop words, unrecognized words, and words that don't have clear meaning, are filtered out).

Data preprocessing step manipulates the blogs and their contents into labeled link network. The nodes are the blogs and the links are the shared words. A blog connects to the others if they share the same words. And because the shared words usually don't appear once in the blogs, the labeled links are weighted with the total number of shared words' appearance in all blogs that share those words. Fig. 4 describes the process of constructing labeled links network from blogs dataset.

The blogs and shared words can be represented mathematically by characteristic matrix (see fig. 4 (b)). From this matrix, we can form bipartite graph, where the nodes are blogs and shared words. And then we can manipulate this bipartite graph into labeled link network, which is the form that we need to apply PARAFAC decomposition into its adjacency tensor. Because the network is constructed from bipartite graph, the result is undirected, so each frontal slices of the adjacency tensor (adjacency matrix for each shared words) is a symmetric matrix. Algorithm in fig. 6 below is used to create adjacency tensor from characteristic matrix.

4. Experiment Results

We decompose the adjacency tensor into 2, 4, ..., 14 groups. Our code is written in MATLAB by using MATLAB Tensor Toolbox [7] and run in a notebook with Mobile AMD processor 3000+ and 480 MB DDR RAM. The maximum number of groups, 14, was not chosen but the maximum number that our computer can process due to the memory limitation. The computational time increases rapidly as the number of groups increases, with approximately 15 minutes for the 14-group decomposition. Table 1 shows the result for two-group and table 2 shows for four-group decomposition.

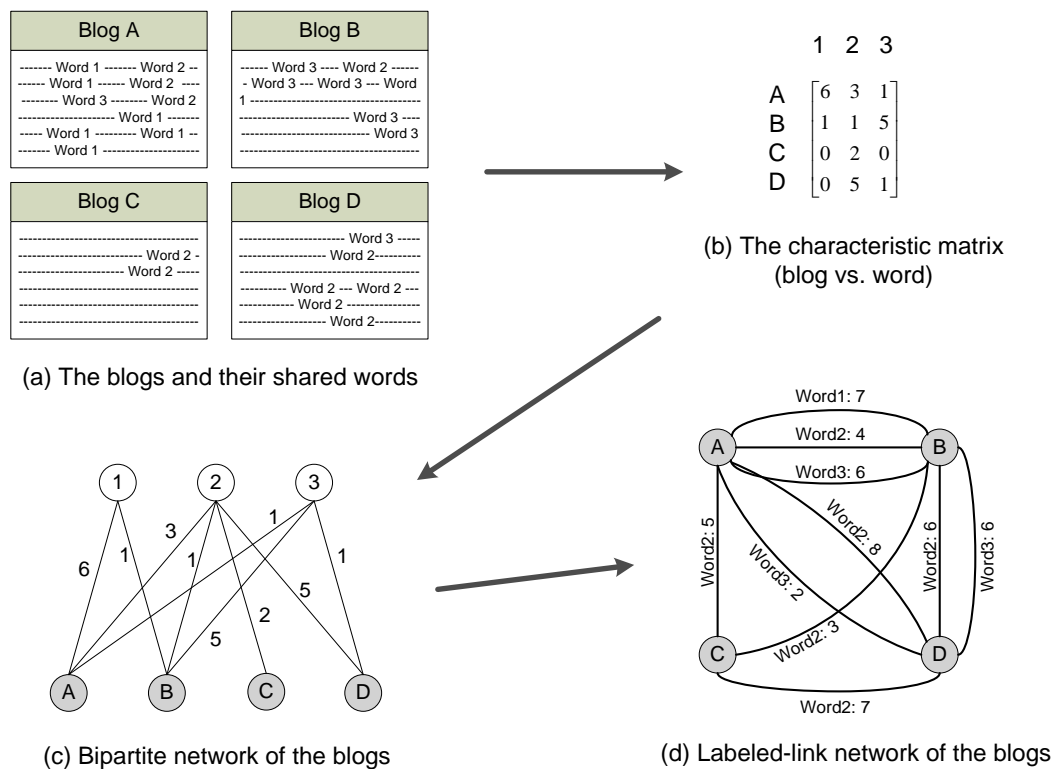


Fig. 4. The labeled link network construction process from blogs dataset.

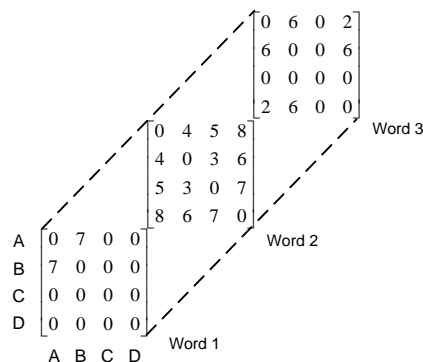


Fig. 5. The adjacency tensor of labeled link network in fig. 4 (d).

In : Blog's Characteristic Matrix C	
Out : Blog's Adjacency Tensor X	
$K = \text{number of column of } C$ $I = \text{number of row of } C$ for $k = 1, 2, \dots, K$, do for $i = 1, 2, \dots, I$, do $X(i, :, k) = C(:, k)$ $X(i, i, k) = 0$ if $C(i, k) == 0$ $X(i, :, k) = 0$ end do end do	$Y = X$ for $k = 1, 2, \dots, K$, do for $j = 1, 2, \dots, I$, do for $i = 1, 2, \dots, I$, do if $X(i, j, k) \neq 0$ $X(i, j, k) = X(i, j, k) + Y(j, i, k);$ end do end do end do end do delete U, Y

Fig. 6. Algorithm to manipulate the characteristic matrix into adjacency tensor

Table 1. Two-group decomposition.

First Group				Second Group			
Blog	Score	Word	Score	Blog	Score	Word	Score
A Consuming Experience	2916.1	Blog	0.90727	Search Engine Land	2114.7	google	0.86815
The Viral Garden	1966.3	those	0.11279	Google Operating System	1668.4	search	0.44746
Search Engine Land	1286.4	Free	0.11032	A Consuming Experience	1219.3	engine	0.063298
Quartz Mountain Weblog	1043.6	video	0.09926	Official Google Blog	931.68	social	0.052491
Tech Gadget Blog	889.5	world	0.096373	The Viral Garden	631.51	online	0.047412
The thinking blog	838.23	Right	0.095685	Search Engine Roundtable	599.15	image	0.042248
i Thought, therefore i Blog	826.66	Top	0.086629	Quartz Mountain Weblog	577.57	mobile	0.041141
Bloggers Blog	812.21	news	0.084289	Search Engine Watch Blog	546.03	news	0.038686
Neil Gaiman's Journal	783.03	media	0.083858	Blog Maverick	544.02	business	0.03801
India PR Blog	780.75	online	0.082614	Valleywag	537.74	information	0.037634
...

Table 2. Four-group decomposition.

First Group				Second Group			
Blog	Score	Word	Score	Blog	Score	Word	Score
A Consuming Experience	2228.3	blog	0.98113	Search Engine Land	2092.1	google	0.87334
The Viral Garden	1773.5	marketing	0.065005	Google Operating System	1686.8	Search	0.44775
Quartz Mountain Weblog	1124.9	Free	0.052199	A Consuming Experience	1036.7	engine	0.066004
Search Engine Land	1084.6	Top	0.048704	Official Google Blog	937.78	social	0.049286
Tech Gadget Blog	961.13	feature	0.029986	Search Engine Roundtable	647.05	mobile	0.040725
i Thought, therefore i Blog	933.98	online	0.025942	Quartz Mountain Weblog	581.14	online	0.035622
Bloggers Blog	887.89	media	0.025532	Search Engine Watch Blog	580.19	yahoo	0.03386
Neil Gaiman's Journal	880.54	reader	0.024918	The Viral Garden	561.53	image	0.032817
The thinking blog	878.92	company	0.023361	Valleywag	552.72	ads	0.030336
India PR Blog	873.41	internet	0.022821	Blog Maverick	542.83	business	0.029824
...

Third Group

Blog	Score	Word	Score
Ask MetaFilter	459.66	blog	0.97118
TreeHugger Radio	451.43	google	0.16145
Techdirt	450.12	marketing	0.058856
Deadspin	447.44	search	0.053256
The Unofficial Apple Weblog	446.67	top	0.048478
Singapore Entrepreneurs	446.59	media	0.048383
Boing Boing	443.47	free	0.046517
43 Folders	442.48	reader	0.042519
Topix.net Weblog	441.5	service	0.03627
Mashable!	440.99	social	0.033971
...

Fourth Group

Blog	Score	Word	Score
NewsBusters.org	882.9	world	0.28811
A Consuming Experience	698.22	right	0.28704
Search Engine Land	668.38	news	0.28023
The Corner	658.24	video	0.24431
Singapore Angle	619.29	america	0.21115
Gothamist	579.5	life	0.1852
we make money not art	567.74	media	0.1562
lifehack.org	557.94	free	0.15519
Blog Maverick	551.35	online	0.14027
the thinking blog	542.45	report	0.13522
...

5. Results Assessment

This section we assess the quality of tensor decomposition by using similarity measure (cosine criterion) between the results and standard measure results. Because decomposition produces two scores, blog and shared word scores for each groups, query vectors that ask both decomposition results and standard measures for relevant/similar blogs or shared words have to be used. Because the query vectors and the groups to be found can be blogs or shared words, there are four possibilities in the query - result as shown in table 3.

Table 3. Query - result relationship.

Query	Relevant/similar group to be found
Blogs	Blogs
Blogs	Shared words
Shared Words	Blogs
Shared words	Shared words

As the standard measure, because it has to be something that doesn't have or produces errors or approximation values, blog's characteristic matrix \mathbf{C} (shown conceptually in fig. 4(b)) is used. Before moving further to find relevant/similar group to the query, the blog's similarity matrix \mathbf{B} and shared word's similarity matrix \mathbf{W} must be calculated in advanced. Let N be the number of blogs and M be the number of shared words, matrix \mathbf{B} is $N \times N$ matrix with its entries defined as:

$$\mathbf{B}(i, j) = \cos \angle(\mathbf{C}(i,:), \mathbf{C}(j,:)), \quad 1 \leq i, j \leq N \quad (3)$$

and matrix \mathbf{W} is $M \times M$ matrix with entries defined as:

$$\mathbf{W}(p, q) = \cos \angle(\mathbf{C}(:, p), \mathbf{C}(:, q)), \quad 1 \leq p, q \leq M \quad (4)$$

The last thing to be considered before the similarity between tensor decomposition results and standard measure results being calculated is the query vectors. There are two query vectors, $N \times 1$ blog's query vector \mathbf{q}_{blog} , and

$M \times 1$ word's query vector \mathbf{q}_{word} , where the entry is one if the blogs/words appear in queries and zero otherwise. Here we are only interested in knowing the quality of decomposition results in average, and not in evaluating specific cases, so all entries of vector \mathbf{q} is set to one.

5.1 Task 1: Finding the Most Relevant Blogs to Words Query

For standard measure case, matrix \mathbf{C} is being used.

$$\mathbf{b}_{standard} = \mathbf{C}\mathbf{q}_{word}, \text{ where } \mathbf{q}_{word} = \text{ones}(M,1) \quad (5)$$

For tensor decomposition results we calculate:

$$\mathbf{m} = \mathbf{T}^T \mathbf{q}_{word} \text{ and } \mathbf{b}_{decomp.} = \mathbf{H}\mathbf{m} \quad (6)$$

The similarity between ranking vector of standard measure, $\mathbf{b}_{standard}$ and tensor decomposition results, $\mathbf{b}_{decomp.}$ is calculated using cosine criterion.

$$\text{similarity}(\mathbf{b}_{standard}, \mathbf{b}_{decomp.}) = \cos \angle(\mathbf{b}_{standard}, \mathbf{b}_{decomp.}) \quad (7)$$

5.2 Task 2: Finding the Most Relevant Words to Blogs Query

For standard measure case, matrix \mathbf{C} is being used.

$$\mathbf{w}_{standard} = \mathbf{C}^T \mathbf{q}_{blog}, \text{ where } \mathbf{q}_{blog} = \text{ones}(N,1) \quad (8)$$

For tensor decomposition results we calculate:

$$\mathbf{m} = \mathbf{H}^T \mathbf{q}_{blog} \text{ and } \mathbf{w}_{decomp.} = \mathbf{T}\mathbf{m} \quad (9)$$

The similarity between ranking vector of standard measure, $\mathbf{w}_{standard}$ and tensor decomposition results, $\mathbf{w}_{decomp.}$ is calculated using cosine criterion.

$$\text{similarity}(\mathbf{w}_{standard}, \mathbf{w}_{decomp.}) = \cos \angle(\mathbf{w}_{standard}, \mathbf{w}_{decomp.}) \quad (10)$$

5.3 Task 3: Finding the Most Similar Blogs to Blogs Query

For standard measure case, matrix \mathbf{B} is being used.

$$\mathbf{b}_{standard} = \mathbf{B}\mathbf{q}_{blog}, \text{ where } \mathbf{q}_{blog} = \text{ones}(N,1) \quad (11)$$

For tensor decomposition results we calculate:

$$\mathbf{m} = \mathbf{H}^T \mathbf{q}_{blog} \text{ and } \mathbf{b}_{decomp.} = \mathbf{H}\mathbf{m} \quad (12)$$

The similarity between ranking vector of standard measure, $\mathbf{b}_{standard}$ and tensor decomposition results, $\mathbf{b}_{decomp.}$ is calculated using cosine criterion.

$$similarity(\mathbf{b}_{standard}, \mathbf{b}_{decomp.}) = \cos \angle(\mathbf{b}_{standard}, \mathbf{b}_{decomp.}) \quad (13)$$

5.4 Task 4: Finding the Most Similar Words to Words Query

For standard measure case, matrix \mathbf{W} is being used.

$$\mathbf{w}_{standard} = \mathbf{W}\mathbf{q}_{word}, \text{ where } \mathbf{q}_{word} = ones(M,1) \quad (14)$$

For tensor decomposition results we calculate:

$$\mathbf{m} = \mathbf{T}^T \mathbf{q}_{word} \text{ and } \mathbf{w}_{decomp.} = \mathbf{T}\mathbf{m} \quad (15)$$

The similarity between ranking vector of standard measure, $\mathbf{w}_{standard}$ and tensor decomposition results, $\mathbf{w}_{decomp.}$ is calculated using cosine criterion.

$$similarity(\mathbf{w}_{standard}, \mathbf{w}_{decomp.}) = \cos \angle(\mathbf{w}_{standard}, \mathbf{w}_{decomp.}) \quad (16)$$

5.5 The Similarity Measures

Table 4 summarizes the similarity measures calculation in eq. (7), (10), (13), and (16). The total similarity measures for task 1 and 3 give good results, about 87%. And because in real situation users usually want to find set of blogs that match with their keywords (task 1, find the most relevant blogs to words query), and also the most similar set of blogs to the their favorite blogs (task 3, find the most similar blogs to blogs query) the high similarity measures for task 1 and 3 are promising. Also the relatively good result in task 3, finding the most relevant words to the blogs query means that tensor decomposition can also be used to give description of the contents of queried blogs.

Table 4. The summary of similarity measures.

	Group 2	Group 4	Group 6	Group 8	Group 10	Group 12	Group 14	Av.
Task 1	0.8854	0.8748	0.8772	0.8477	0.9077	0.8561	0.8839	0.8761
Task 2	0.8715	0.7924	0.8244	0.6905	0.9223	0.7325	0.7994	0.8047
Task 3	0.8421	0.9461	0.8384	0.9460	0.7329	0.9426	0.8812	0.8756
Task 4	0.4255	0.7666	0.7257	0.7155	0.6320	0.5674	0.5660	0.6284
Av.	0.7561	0.8450	0.8164	0.7999	0.7987	0.7747	0.7826	0.7962

5. Conclusions

This paper discusses the possibility of using tensor decomposition in grouping blogs and their contents. From results assessment, it can be concluded that decomposition on 3-way adjacency tensor of blogs network can be used to group the blogs with their most descriptive terms and the blogs with other similar blogs.

Bibliography

1. M.E.J. Newman, "Clustering and Preferential Attachment in Growing Networks," in *Phys. Rev. E* 64, 025102, 2001.
 2. M.E.J. Newman, "Finding Community Structure in Networks Using the Eigenvectors of Matrices," in *Phys. Rev. E* 74, 036104, 2006.
 3. Y. Kawachi, S. Yoshii, and M. Furukawa, "Labeled Link Analysis for Extracting User Characteristics in E-commerce Activity Network," in *Proceeding of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence*, 2006, pp. 73-80.
 4. Mirzal, *Development of Link Structure Ranking Algorithms and Clustering Methods for Networks*, Master Thesis, Graduate School of Information Science and Technology, Hokkaido University, 2008.
 5. T.G. Kolda, B.W. Bader, and J.P. Kenny, "Higher-Order Web Link Analysis Using Multilinear Algebra," in *Proceeding of the 5th IEEE International Conference on Data Mining*, 2005, pp. 242-249.
 6. T.G. Kolda and B.W. Bader, "The TOPHITS Model for Higher-Order Web Link Analysis," in *Workshop on Link Analysis, Counterterrorism and Security*, 2006.
 7. Andersson and R. Bro. "The N-way toolbox for MATLAB," in *Chemometrics & Intelligent Laboratory Systems*, 52(1):1-4, 2000. (<http://www.models.kvl.dk/source/nwaytoolbox/>)
-

Authors' Information

Andri Mirzal – PhD Student; Graduate School of Information Science and Technology, Hokkaido University, Kita 14 Nishi 9, Kita-Ku, Sapporo 060-0814, Japan; e-mail: andri@complex.eng.hokudai.ac.jp

PRESENTATION OF ONTOLOGIES AND OPERATIONS ON ONTOLOGIES IN FINITE-STATE MACHINES THEORY

Sergii Kryvyi, Oleksandr Khodzinskyi

Abstract: A representation of ontology by using finite-state machine is considered. This representation allows introducing the operations on ontologies by using regular algebra of languages. The operations over ontologies allow automating the process of analysis and synthesis for ontologies and their component parts.

Keywords: ontology, operations, finite automata.

ACM Classification Keywords: I.2.4 Knowledge Representation Formalisms and Methods; F.4.1 Finite Automata

Introduction

Of late years in natural sciences and in particular in the theoretical programming so much different directions, trends and theoretical results appeared that it becomes problematic to overcome all field of scientific activity as a whole even in the separately taken areas. One of the approaches to understand interconnections between different trends and theories is the ontological approach [1, 2]. In addition, due to increase of complication of software and hardware systems "intellectualization" of these processes is required and in opinion of many specialists the attaining of such "intellectualization" is possible by using of the ontology-driven systems of searching, mining and processing of knowledge, which are contained in ontologies. Ontological approach of the development of connections between the concepts of some data domain usually is based on the determination of the relation «the data domain – properties – models – applications». In this work the method of representation of ontology via finite-state machines and relations lying in the basis of every ontology are considered. This approach allows entering operations on ontologies using operations on languages and finite state machines. At such approach types of ontologies and their hierarchy are not detailed with the purpose of underlining community of the examined operations. Operations are illustrated by the simple examples of ontologies, related to computer mathematics [3].

1. Presentation of Ontologies in Finite-state Machines Theory

Let us represent ontologies as directed graph of $G = (V, E)$, where the set of vertexes V is the set of data domains and the set of edges E is a binary relation between these data domains.

With each directed graph $G = (V, E)$ we associate finit-state (generally speaking) partial determined machine $A = (V, X=v, f, S, F)$ without outputs, where V is a set of the states which also serves as the source alphabet of this machine, S is a subset of the initial states, F is a subset of the final states (which, in particular, can be empty), and the function of transitions of this machine is determined as follows: $f(u,v) = v$ only in the case when $(u,v) \in E$ and it is not determined in other cases.

Let us consider the example of representation of the fragment of ontology for the data domain «Combinatorics» [5].

Example 1. Let us set the ontology representing small part of the data domain «Combinatorics» as following directed graph:

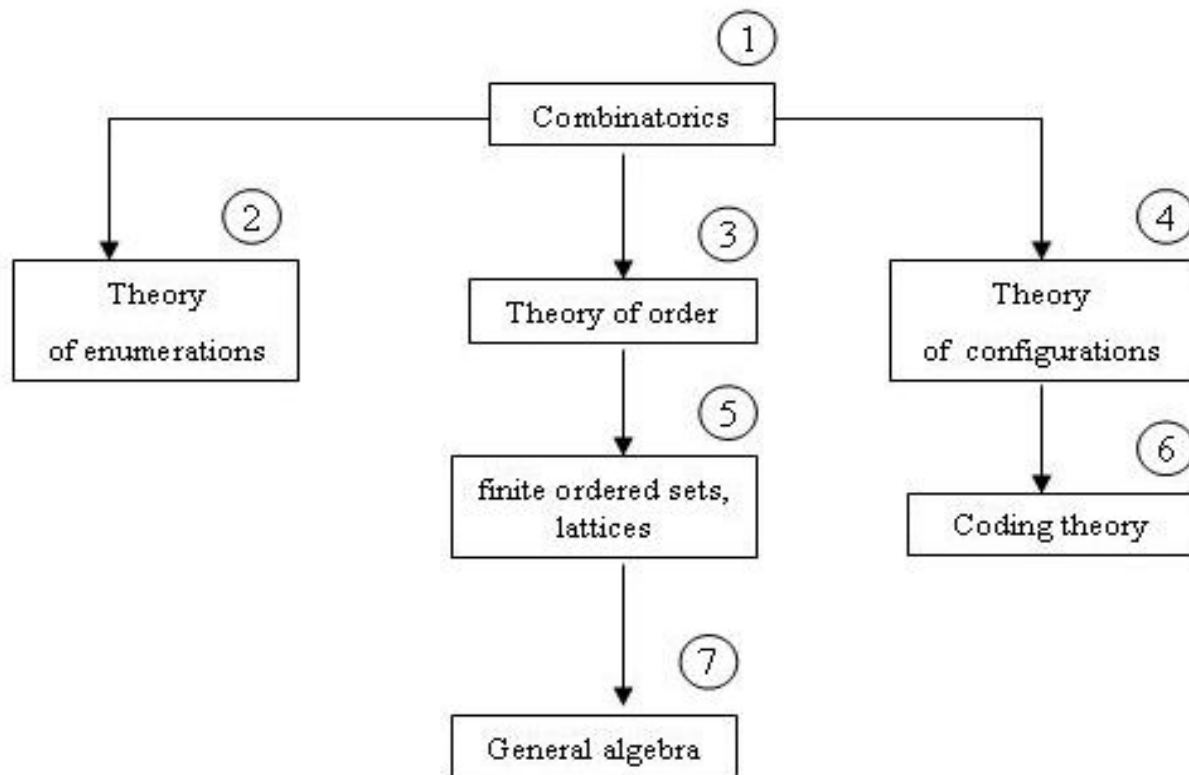


Fig. 1. Ontology O

Finite-state machine, which corresponds to this ontology, is of the following form $A = (V = \{1, 2, 3, 4, 5, 6, 7\}, X = \{1, 2, 3, 4, 5, 6, 7\}, f, \{1\}, \{7\})$, where f is determined as following transition graph:

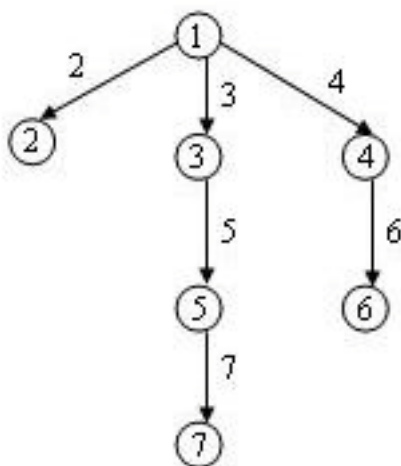


Fig. 2. Finite state machine A for O

It means that $f(1,2)=2$, $f(1,3)=3$, $f(1,4)=4$, $f(3,5)=5$, $f(5,7)=7$, $f(4,6)=6$. Other transitions in this machine are not determined.

2. Operations on Ontologies in Finite-state Machines Representation

Representation of ontology as finite-state machine without output allows entering operations on ontologies. Operations on machines mean operations on regular languages which are accepted by these machines. There are following basic operations:

union is a set-theoretic union of the set of states and the set of transitions of these machines-arguments;

intersection is the set-theoretic intersection of the set of the states and the set of transitions replenished by the transitive closure of relation of attainability on machines-arguments;

appending or **multiplication** of two machines is the special case of operation of union, when an association is executed only on the great number of the initial states of the second machine;

iteration – repeated eventual number of one times operation of increase, applied within the framework of one ontology with the purpose of clarification and addition to this ontology (this operation means the ontology incremental refinement and addition);

an appeal is an opposite orientation of transitions in an automat, presenting this ontology, i.e. construction of the function of transitions of $g(v,u)=u$ if and only if, when $f(u,v)=v$ and indefinitely in other cases.

Example 2. Let ontology of kind is given

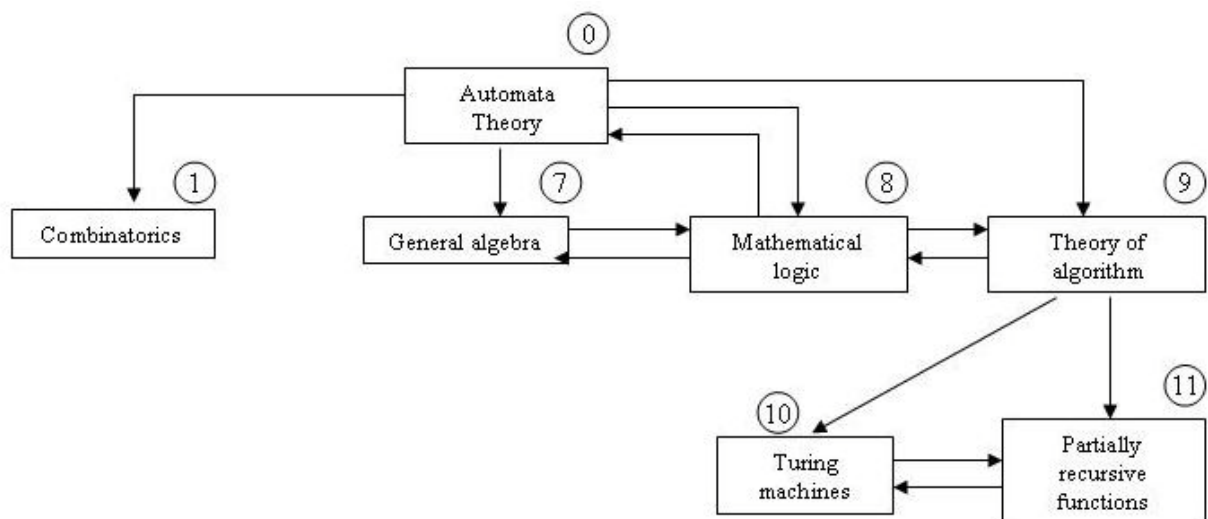


Fig. 3. Ontology O_1

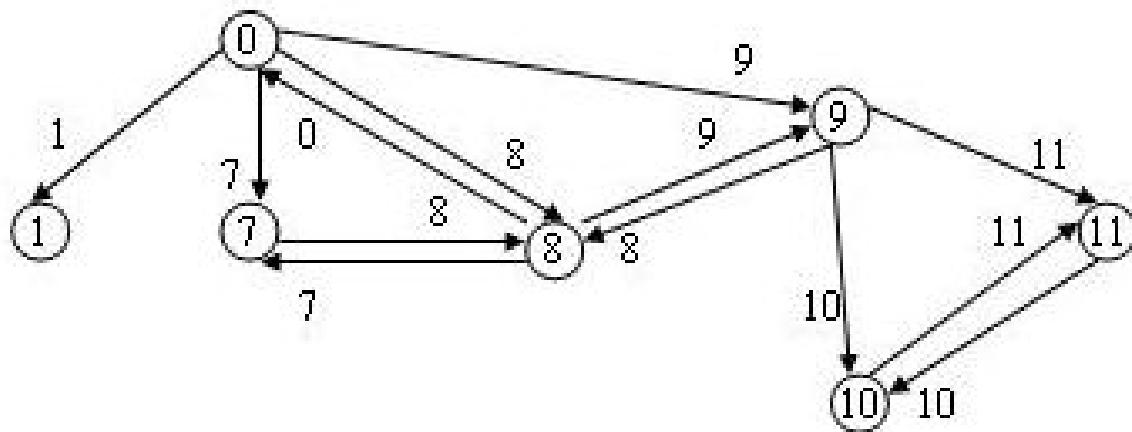


Fig. 4. Finite state machine A_1 for O_1

where $A_1 = (\{0, 1, 7, 8, 9, 10, 11\}, \{0, 1, \dots, 11\}, g, \{0\}, \{11\})$.

If one applies the operations entered above to the automats of A_1 and A from a previous example they give such results.

Union:

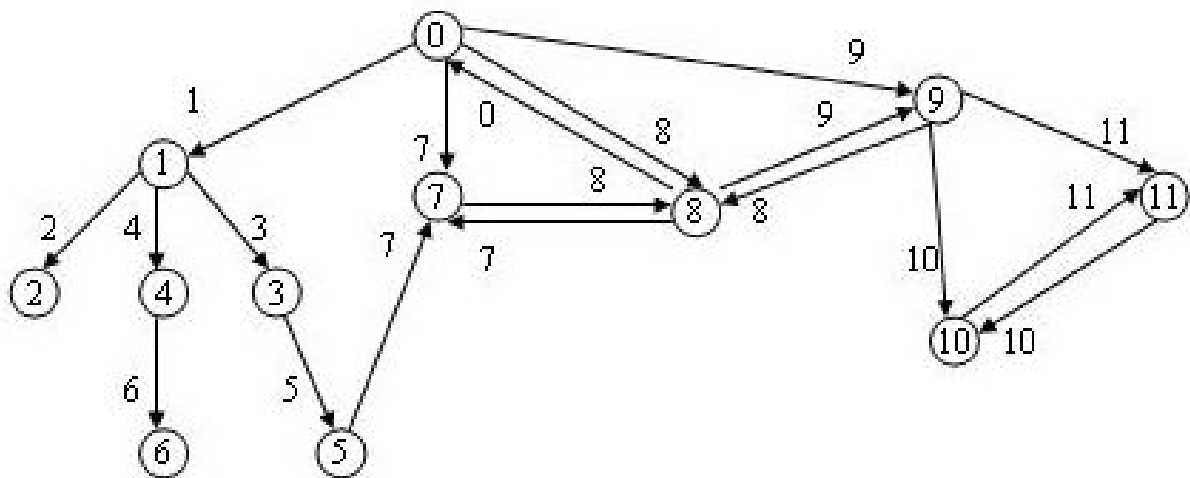


Fig. 5. Finite state machine $A \cup A_1$

Intersection:



Fig. 6. Finite state machine $A \cap A_1$

Iteration: clarification of ontology of O_1 :

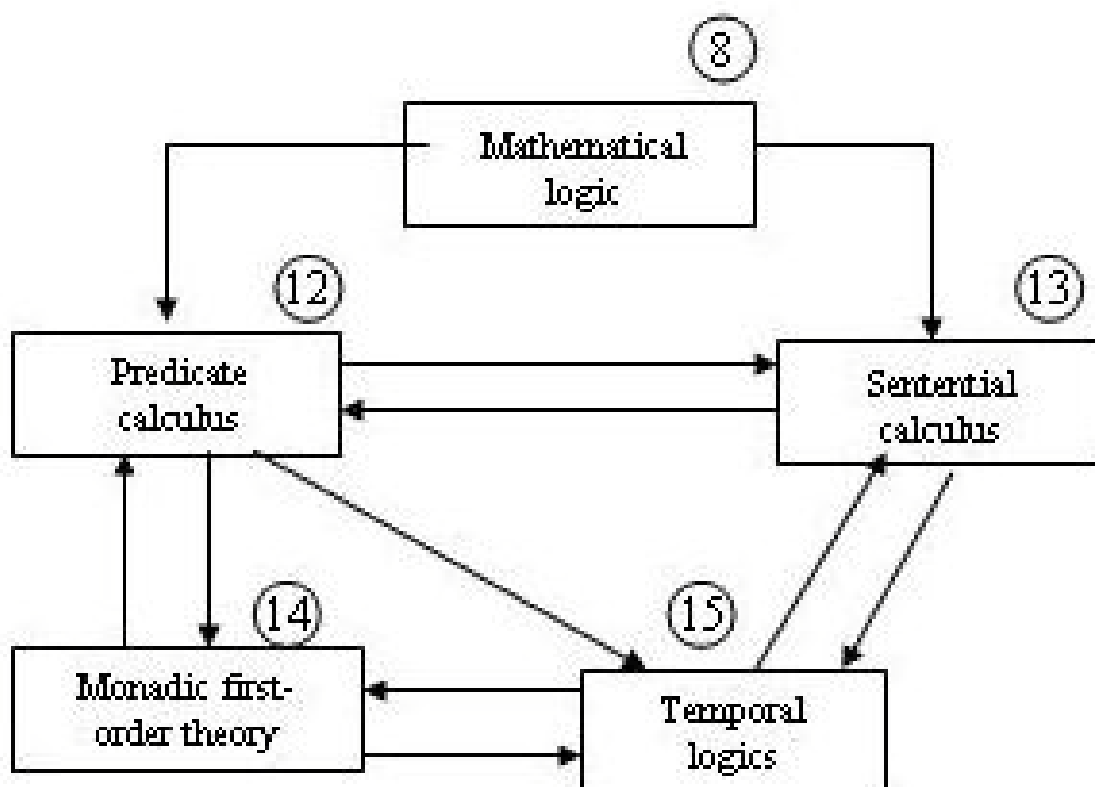


Fig. 7. clarification O_2 of ontology of O_1

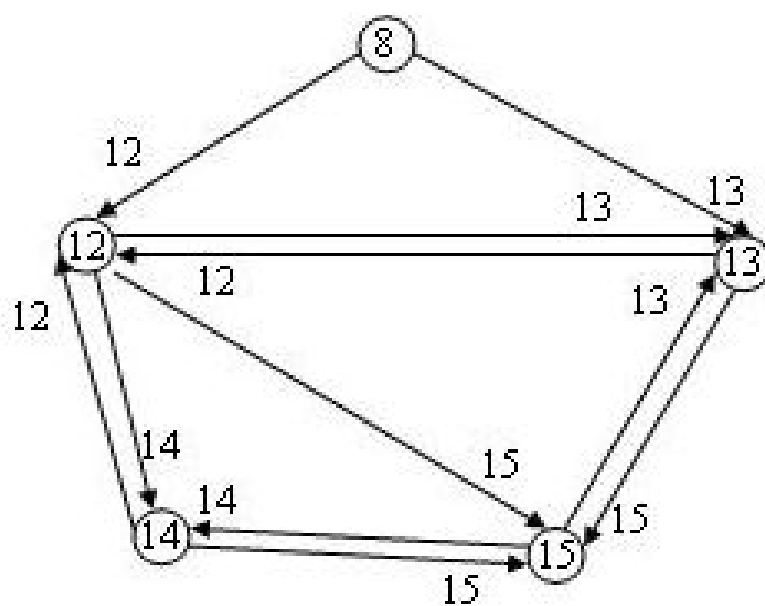


Fig. 8. Finite state machine A_2 for ontology of O_1

Concatenating automats A_1 and A_2 on the initial state 8 automat A_2 , get an automat, presenting the specified ontology $O_1 * O_2$.

Appeal: applying this operation to A_1 , get an automat:

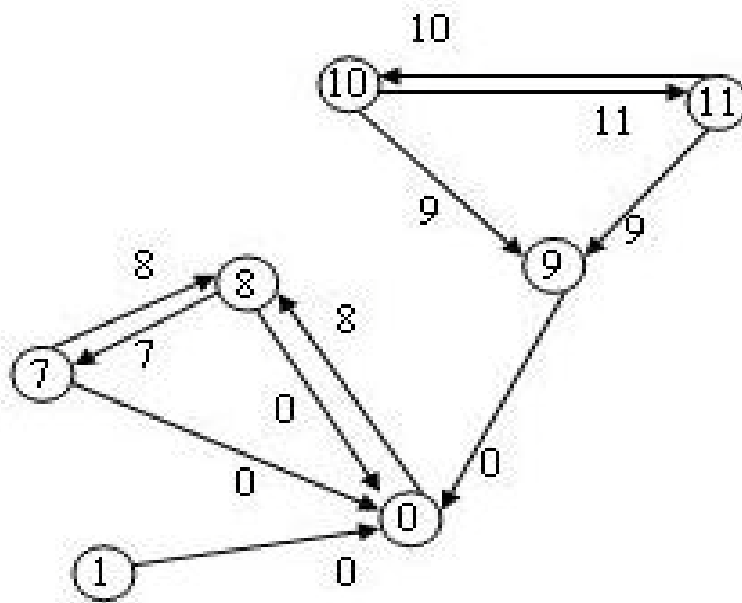


Fig. 9. Converting finite state machine for ontology of O_1

3. Short-story Description of Operations

Algebraic properties of the entered operations on ontologies follow from the proper properties of operations of algebra of regular languages. It means that these operations satisfy the followings laws: commutativity and associativeness of operations of association and crossing, associativeness of increase, distributivity of operation of increase in relation to the operations of association and crossing.

This great number of operations (in the case of necessity) can be extended at least in two directions. One of such directions is expansion operations on columns (introduction and delete of top and rib, connection of counts, isomorphic connection [6], cartesian product, etc.). Other direction is algebra of relations. As every ontology is presentation as some aggregate of relations (in particular: one), it is possible to enter the operations of relational algebra.

What from possible directions will be chosen, depends on the practical necessities of the use of ontologies. To forecast anything does not make sense on that score, because practice appears always richer than any theory. Authors hope that the presented operations above ontologies will appear useful at an analysis, synthesis and manipulation by ontologies and ontological objects.

4. Problems of Realization of Operations

Let us consider now some problems, arising up on the way of realization of these operations.

The first problem (and possibly basic during work with ontologies) is related to that correct implementation of the operations described higher requires creation of some general glossary of subject domains and concepts, which it would be possible to identify proper objects by. Presumably, this problem is not the problem of realization of the entered operations only but also a general issue on the way of construction of ontologies and works with ontologies.

The second problem, arising up during realization of operations, is related to the present hierarchy of areas and concepts. The point is that in different ontologies the same concepts and objects can be on the different levels of hierarchy and it must be taken into account at application of operations. In offered approach this problem is decided by the construction of the transitive closure of accessibility relation on the states of automats, presenting ontology data. However, authors are not sure that this closure is enough for the problem decision. Here it seems that experiments are necessary on real ontologies and their representations.

The third problem is related to completeness of knowledges, presented in given ontologies. This problem is basic in the process of specification and verification of software and hardware systems. Here this problem is related to possibility of the construction of the information system which is in some way complete and guided-by-ontology.

5. Summary

The topic of this work originated from the lectures which were presented on the conferences of KDS-2005 and KDS-2007 (Varna, Bulgaria) [1-4]. An ontologies section of these conferences was one of the greatest and lectures presented in these sections were encouraged for development of presentation of ontology and operations on ontologies with the purpose of automation of process of planning and manipulation by these objects. Possibly, after this attempt to enter operations on ontologies other approaches will appear in order to construct the algebra of ontologies. It would be very desirable and fruitful for further development of this knowledge domain. Our approach, presumably, is not the best, as requires the decision of the problems presented above.

Bibliography

1. Gavrilova T., Puuronen S. In Search of a Vision: Ontological View on User Modeling Conferences' Scope. XII-th International Conference KDS 2007, ITHEA, Sofia, 2007. Volume 2, p.422-427.
2. Gribova V. Automatic Generation of Context-sensitive Help Using a User Interface Project. XII-th International Conference KDS 2007, ITHEA, Sofia, 2007. Volume 2, p.417-422.
3. Кривой С., Матвеева Л., Лукьянова Е., Седлецкая О. Онтологический взгляд на теорию автоматов. XII-th International Conference KDS 2007, ITHEA, Sofia, 2007. Volume 2, p.427-436.
4. Artemieva I. XII-th International Conference KDS 2007, ITHEA, Sofia, 2007. Volume 2, p.403-411.
5. Кривий С. Л. Дискретна математика. Вибрані питання. Київ. Видавничий дім „Києво-Могилянська академія”. - 2007. - 572 с.

Authors' Information

Kryvyi Sergii – *Glushkov Institute of Cybernetics of NSA of Ukraine, Ukraine, Kiev, 03187, 40 Glushkova Street, Institute of Cybernetics*, e-mail: krivoi@i.com.ua

Khodzinskyi Oleksandr – *Glushkov Institute of Cybernetics of NSA of Ukraine, Ukraine, Kiev, 03187, 40 Glushkova Street, Institute of Cybernetics*, e-mail: okhodz@gmail.com

COGNITIVE APPROACH IN CASTINGS' QUALITY CONTROL

Irina Polyakova, Jürgen Bast, Valeriy Kamaev,
Natalia Kudashova, Andrey Tikhonin

Abstract: Every year production volume of castings grows, especially grows production volume of non-ferrous metals, thanks to aluminium. As a result, requirements to castings quality also increase. Foundry men from all over the world put all their efforts to manage the problem of casting defects. The authors suggest using cognitive approach to modeling and simulation. Cognitive approach gives us a unique opportunity to bind all the discovered factors into a single cognitive model and work with them jointly and simultaneously. The method of cognitive modeling (simulation) should provide the foundry industry experts a comprehensive instrument that will help them to solve complex problems such as: predicting a probability of the defects' occurrence; visualizing the process of the defects' forming (by using a cognitive map); investigating and analyzing direct or indirect "cause-and-effect" relations. The cognitive models mentioned comprise a diverse network of factors and their relations, which together thoroughly describe all the details of the foundry process and their influence on the appearance of castings' defects and other aspects. Moreover, the article contains an example of a simple die casting model and results of simulation. Implementation of the proposed method will help foundry men reveal the mechanism and the main reasons of casting defects formation.

Keywords: castings quality management, casting defects, expert systems, computer diagnostics, cognitive model, modeling, simulation.

ACM Classification Keywords: I.6.5 [Computing Methodologies - Simulation and Modelling] Model Development - Modeling methodologies

Introduction

Every year production volume of castings grows, especially grows production volume of non-ferrous metals, thanks to aluminium. Aluminium and its alloys have been the prime material of construction for the aircraft industry, car production and also attractive in other areas of transport thanks to the combination of acceptable cost, low component mass, appropriate mechanical properties, structural integrity and ease of fabrication.

Aluminium alloys are processed with the help of all possible techniques of metal casting. In order to choose the right technique all technical and economic requirements (casting size, weight, wall width, casting complexity, required quantity of castings) should be attended [VDS, 1988]. The most commonly used techniques are pressure die casting (about 58 % in 1998 in Germany), gravity die casting (about 33 %) and sand mould casting (9 %) [Drossel, 1999].

Because of continuous growing of castings output from aluminium alloys, requirements to the castings quality also increase. Aluminum casting alloys have a number of typical casting defects. Among them are building of oxides, porosity, misrun etc. By casting defect we understand a technical characteristics mismatch of produced castings and technical requirements which the castings should meet.

While we still get castings with defects that make castings more expensive, foundry men from all over the world put all their efforts to manage the problem of casting defects.

1. Identifying the Problem

Because of the high oxidation susceptibility, aluminium alloys form oxide films on the surface, which affect the quality of the melt and make it viscous. Consequently, inclusions of oxides can be observed in the form of casting defects. In practice the aluminium alloys, which tend to high oxygen content, also tend to high hydrogen content as well. High hydrogen content in the melt is one of the immediate causes of porosity defect. The other typical casting defect of aluminium alloys is misrun (when the melt become hard too early, before it reaches all edges).

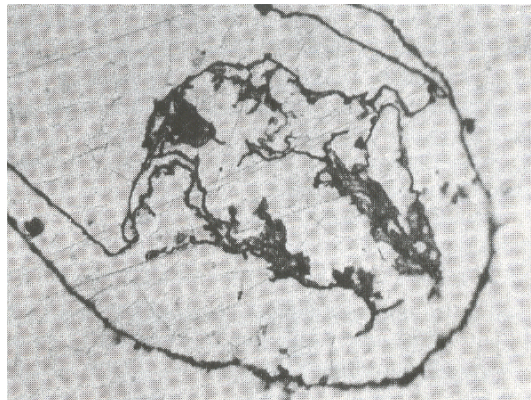


Fig. 1. Casting defect - oxide film [Altenpohl, 1994]

The process of casting defects formation depends on a great amount of factors and parameters. Very often, these dependencies are very complex by their nature; they could only be described by numerous factors, which in their turn could have varying interactions with others. Moreover, we experience a lack of knowledge about the collection of factors in general and have limited opportunities of their quality understanding.

Nowadays we see a large number of simulation methods that work with complex dynamic systems and processes. The choice of a method depends on the level of complexity of a system and on the volume of information about it. Today there are four main groups of methods in the sphere of "fighting" with casting imperfections:

- the first method is based on the use of atlases of casting defects. Many research centers are still working on them. It is difficult to find right reasons with the help of such atlases, especially when there is more than one reason or more than one defect;
- the second method deals with classical expert systems. Though many scientific groups work on them, such kind of software systems does not give us an opportunity to observe the behavior of all system parameters when we want to improve values of some of them;
- the third group works on a so-called black box technology. The black box technologies comprise methods that are based for example on neural networks. Nevertheless, the problem of neural networks lies in the lack of transparency of the process of making inferences. This means that it is impossible to understand the grounds of the conclusion being made;

- the forth method is a simulation. This is a common method, when a system is represented by a number of differential equations, which describe energy conservation laws that occur in the system. Moreover, it is the most laborious and complicated method, which is also rather costly and time consuming.

Though simulation is the most powerful method very often there is no need to spend time and money to build a model. Frequently the problem lies in understanding of the occurring processes. Moreover, in the field of "fighting" with casting defects, it is impossible to build a fully adequate mathematical model, which summarizes all the factors and their interrelations. Even now, there are a lot of interferences, which are still not described mathematically and could only have verbal interpretation.

To solve the problem of understanding of processes that occurred the authors suggest using cognitive approach to modeling and simulation. The approach is based on the usage of cognitive maps in computer simulation. With the help of cognitive maps, it is possible

- to make visible and transparent all the occurred processes and the whole complex network of reasons of casting defect formation;
- give plausible results without a need to answer difficult questions about cause-and-effect relations;
- capture and visually present the structure of expert knowledge – a set of factors from the problem domain and cause-and-effect relations between them, –and also generate structure-related information about possible consequences [Axelrod, 1976];
- easily make certain evaluations [Axelrod, 1976].

It is very important in the area of "fighting" with casting defects, because the change of one parameter usually cause changes in other manufacturing parameters. Very oft it is very difficult to suppose the consequences of problem solving. Possibly a new problem (Fig.2)?

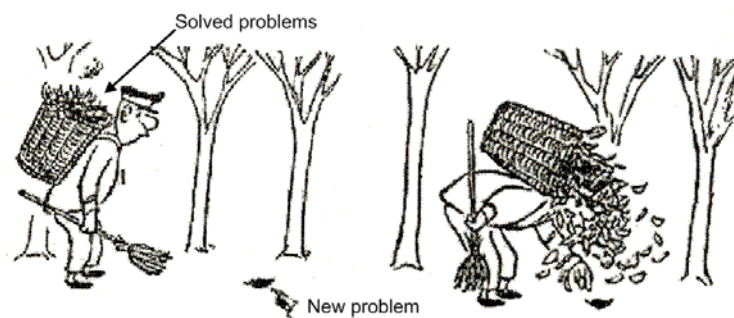


Fig. 2. Problem solving

The fight with casting defects can lead to undesirable after-effects: appearance of other imperfections, loss of time, money, resources and finally to loss of productivity. Cognitive modeling makes it possible to conduct fast and more or less exact quantitative virtual experiments with the help of proper software and to get the required information.

Cognitive approach gives us a unique opportunity to bind all the discovered factors into a single cognitive model and work with them jointly and simultaneously. Moreover, it is possible to use not only well-known exact interactions, but also interactions, which are only supposed to occur.

Cognitive maps were initially suggested by American psychologist Tolman E. C. to describe behaviour of mice [Tolman, 1948]. Later Axelrod R. used them in politics (model of British politics in Persia) [Axelrod, 1976]. Roberts F. used them in economics (model of energy consumption) [Roberts, 1986].

Commonly a cognitive map is represented as a directed graph $G(V, A)$, where V - factors, A - cause-and-effect relations between factors.

Significant contribution to the development of cognitive maps' theory was made by B. Kosko [Kosko, 1992]. He proposed the most popular modification of cognitive maps - so-called fuzzy cognitive maps (FCM), where values of factors vary from -1 to +1 and some scale is in use.

Cognitive modeling has already been tested with socio-economic systems like regional economy management, industrial safety and so on [Kosko, 1992]. All such systems are complicated and semi-structured systems, which have a large quantity of interacting factors. These interactions could be changed dynamically. The system of casting defect formation has similar characteristics; therefore the cognitive approach should be rather efficient here.

Like many other scientists the authors see further development of cognitive maps' approach in the direction of joining them with fuzzy logic, where factors are linguistic variables and relations represent data banks of fuzzy rules.

Cognitive modeling has already been tested on complicated and semi-structured systems, which have a large quantity of interacting factors. The system of casting defect formation has similar characteristics; therefore the cognitive approach should be rather efficient here.

The method of cognitive modeling (simulation), discussed on the conference, should provide the foundry industry experts a comprehensive instrument that will help them to solve complex problems such as:

- predicting a probability of the defects' occurrence;
- visualizing the process of the defects' forming (by using a cognitive map);
- investigating and analyzing direct or indirect "cause-and-effect" relations.

2. Example of the Cognitive Model

For example, let us consider a cognitive map of AlSi12-alloy and die casting process. In our research, we investigate a great number of factors, which described properties of aluminium-alloy, die, work process-related parameters, work of personnel and machines. In order to arrange reasons of casting defect formation logically and evaluate system reasons a so-called Ishikawa diagram (cause-and-effect diagram; Fig. 3.) is used. In order to find all factors each group was analyzed in detail.

In the classical cognitive approach, all factors's values are qualitatively described (for example "weak", "strong", "reasonably", "high", "low"). Each factor has an own scale from -1 to 1 and ranges within the scale, the boundary values of selected for the example factors are shown in the next table (Tab. 1). In spite of the fact, that some factors can be described quantitatively (like die temperature or width of coating) and others can be evaluated only qualitatively (like continuity of the metal-flow or competence of personal), all factors in the example will be described qualitatively and scaled from -1 to 1.

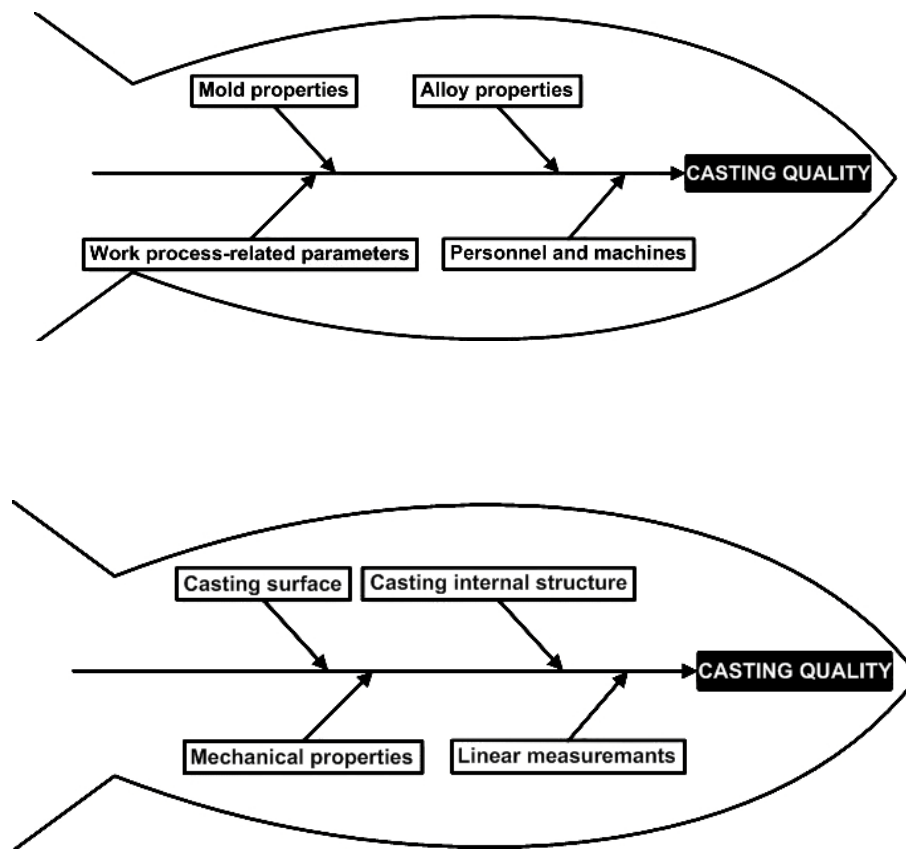


Fig. 3. General view of the Ishikawa-diagrams

Table 1. Boundary values of selected factors

Name of the factor	Min value(-1)	Max value(+1)	Unit
Die temperature	low (300)	high (400)	°C
Pouring rate	low	high	m ³ /s
Width of coating	thin (0,1)	thick (0,2)	mm
Die durability	low	high	Castings
Width of the die wall	thin (2*D, D-wall thickness of the casting)	thick (5*D, D-wall thickness of the casting)	sm.
Shrinkage of the alloy	high (0,8 %)	small (0,5 %)	%
Fluidity	low	high	mm
Mouldfilling ability	low (0,1)	high (1,1)	mm-1
Fe content	low (0,0)	high (1,0)	%

Mn content	low (0,001)	high (0,4)	%
Cu content	low (0,0)	high (0,1)	%
Gas absorbtion of the alloy	low	high	-
Casting strength	low (110)	high (150)	N/mm ²
Porosity	low	high	-
Cold cracks	low	high	-
Misrun	low	high	-
Die ventilation	poor	good	-
Width of the casting wall	thin	thick	sm.
Internal stress	low	high	-
Surface conditions of the die	bad (damaged)	good (undamaged)	-
Evaporation of coating	high	poor	-
Poring temperature	low (680)	high (780)	°C
Oxide	low	high	-
Oxide in melt	little	many	-
Purity of metal melt	bad	good	-
Idle time of the melt	short (10 min)	long (120 Min)	min
Surface conditions of the tank	dirty	clean	-
Melting time	short	long	min
Melt temperature in idle time	low (780)	high (800)	°C
Oxide formation during mold-filling	low	High	-
Pouring height	low	high	sm
Speed of the melt in the pouring gate	low	high	sm/s
Cross-section of the pouring gate	small	big	sm ²
Air absorption during pouring	poor	high	-
Fullness of the pouring gate	empty	full	-
Competence of personal	low	high	-
Continuity of the metal-flow	unsteady	steady	-

If there are no quantitative data about the interactions, it is possible to use a qualitative value (strong, moderate, weak) to define the interactions together with the Harrington's scale [Diligensky, 2004] from -1 to +1. We used the following Harrington's scale to measure the interaction force for all factors not to complicate the model: "lightly"- [0-0.2], "weakly"-[0.2-0.37], "moderate" - [0.37-0.63], "essentially" - [0.63-0.8], "strong" - [0.8-1].

Using qualitative relations, we can also distinguish positive and negative interactions. In other words, a positive interaction shows us that a rise of the factor, which is at the beginning of the arrow, increases the factor value on

the arrow end; a negative interaction on the contrary shows us that a rise of the factor, which is at the beginning of an arrow, decreases the factor value on the arrow end.

Example the qualitative relations according to Harrington's scale:

- purity of metal melt strongly decreases (-1) oxides in the melt;
- temperature difference in the die essentially decreases (-0.7) die durability;
- Cu-presents essentially decreases (-0.7) shrinkage of the alloy;
- pouring temperature essentially increases (0.8) fluidity of the melt;
- continuity of the metal-flow lightly increases (0.2) oxide formation during mold-filling.

An example of possible verbal interpretation sounds: "moderate increase of the die ventilation strongly increases the probability of porosity defect" (Fig. 3).

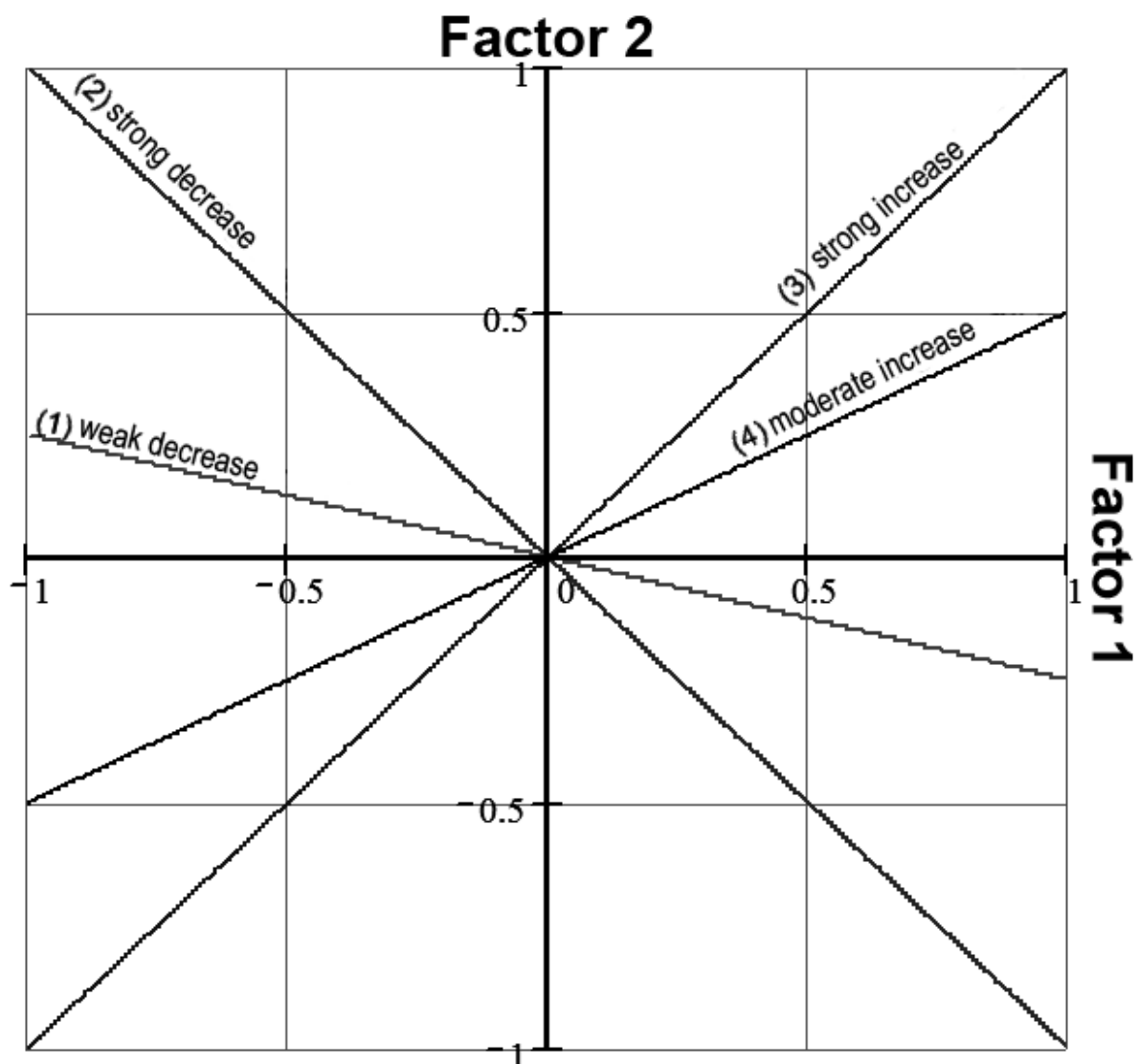


Fig. 3. Qualitative interactions

A fragment of classical cognitive map is shown on Fig. 4. The fragment consists of 38 factors and even more interactions. In the given model, many factors are not attended to reduce a model size for the article.

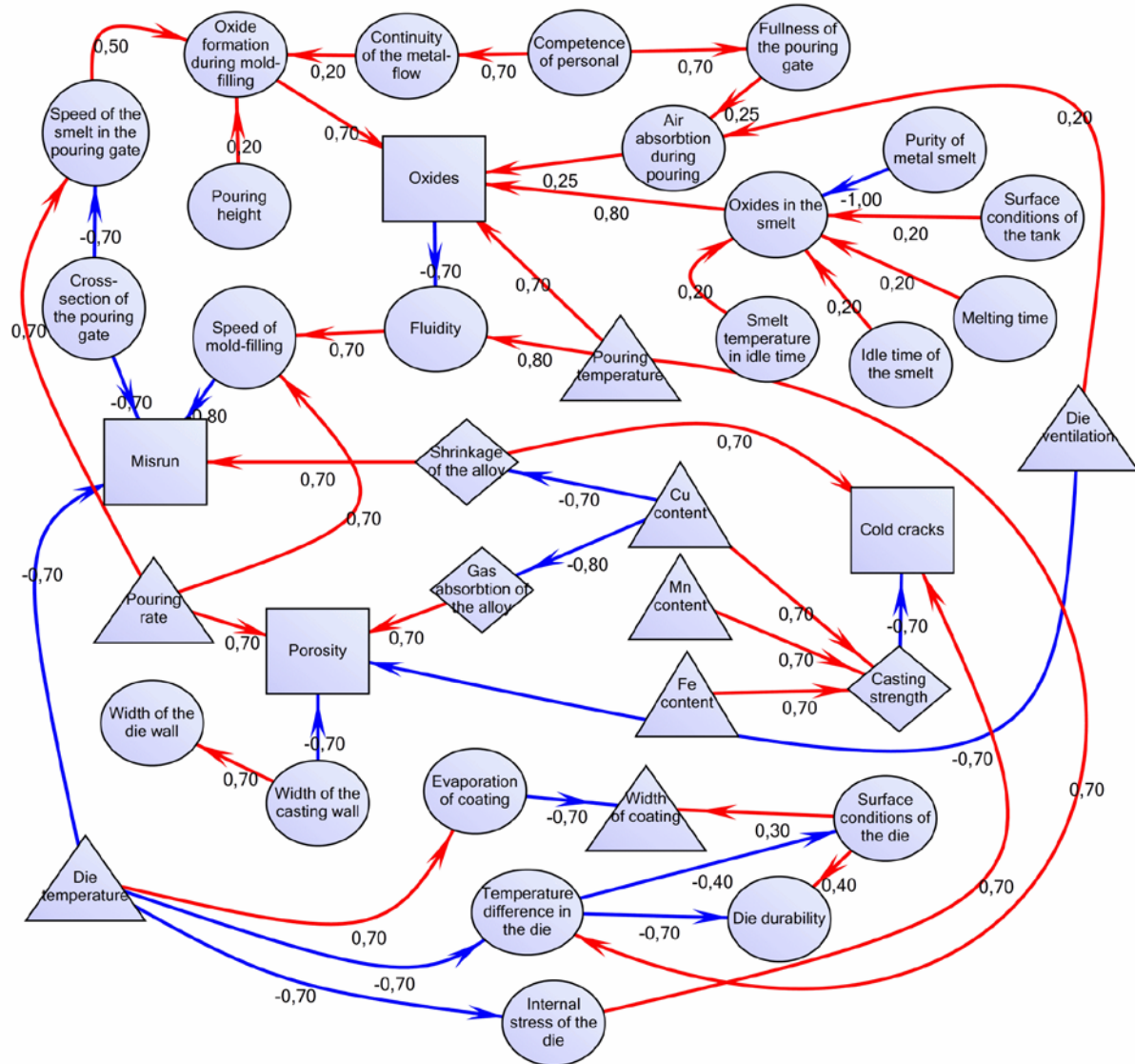


Fig. 4. Fragment of the cognitive map with 38 factors (Δ - control factors, \square - target factors, \circ - other factors)

Among these factors, we can choose so-called target factors, which should alter in a desired direction (shown with squares at Fig. 4). Moreover, we can also select so-called control factors (shown with triangles in Fig. 4), which values we can change in order to achieve desired directions of target factors.

The information about factors is represented in the software system as a matrix. This allows us to run a simulation, which virtually shows possible consequences of alterations of one or more factors.

3. Simulation Example

Let us demonstrate a simple simulation example on the above-mentioned model. The behavior of some factors when adding Cu into the alloy is observed. It can be observed, that the addition of Cu at first influences the gas-absorption ability of the alloy, shrinkage of the alloy and casting strength. Alterations of these factors lead in their turn to the reduction of misruns, porosity and cold cracks. The results of simulation are shown in Fig. 5.

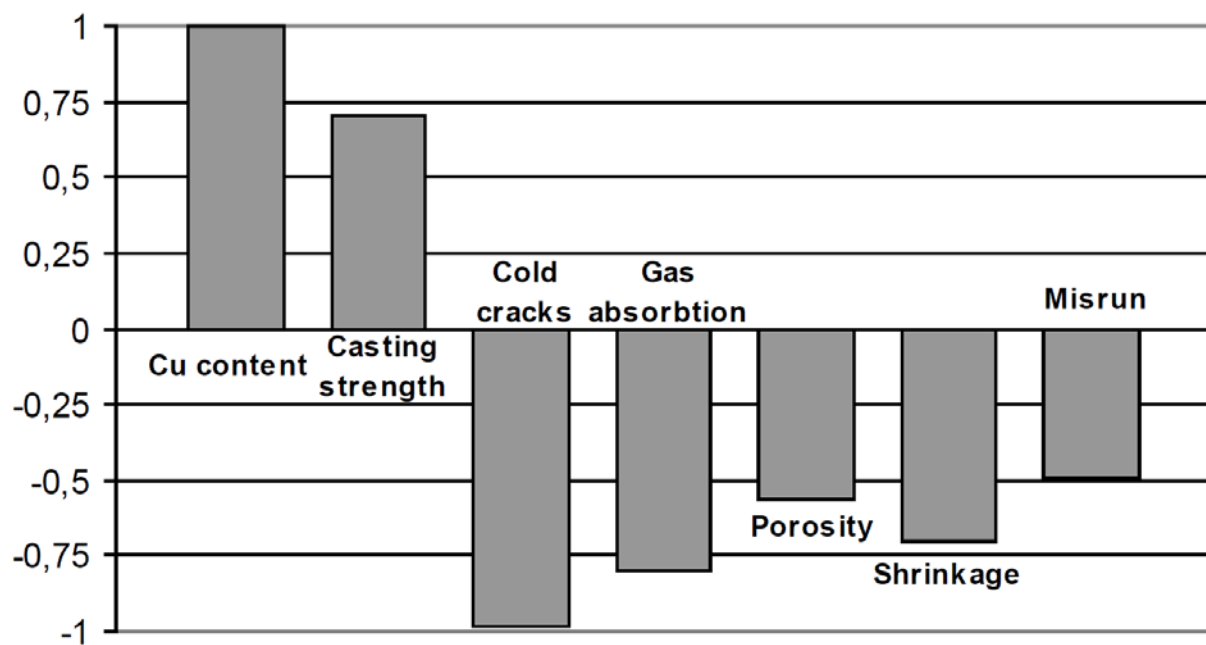


Fig. 5. Simulation example "Cu content – cold cracks, porosity, misrun"

The next example will demonstrate conflicting goals. This time the factor "pouring rate" is chosen. If we look at the graph (Fig. 4), we can see that the increasing of pouring rate of the melt will increase speed of the melt in the pouring gate, which in its turn will lead to increase of oxide formation during mould-filling and consequently we could get oxide film casting defect (see Fig. 6). Otherwise, the increase of pouring rate of the melt increases speed of mould filling, what in its turn decreases the probability of misrun.

This example illustrates how the "fight" with one casting defect can influence on the others. Therefore, it is highly important to consider the whole network of factors, but not concentrate on a single factor or even a group of factors.

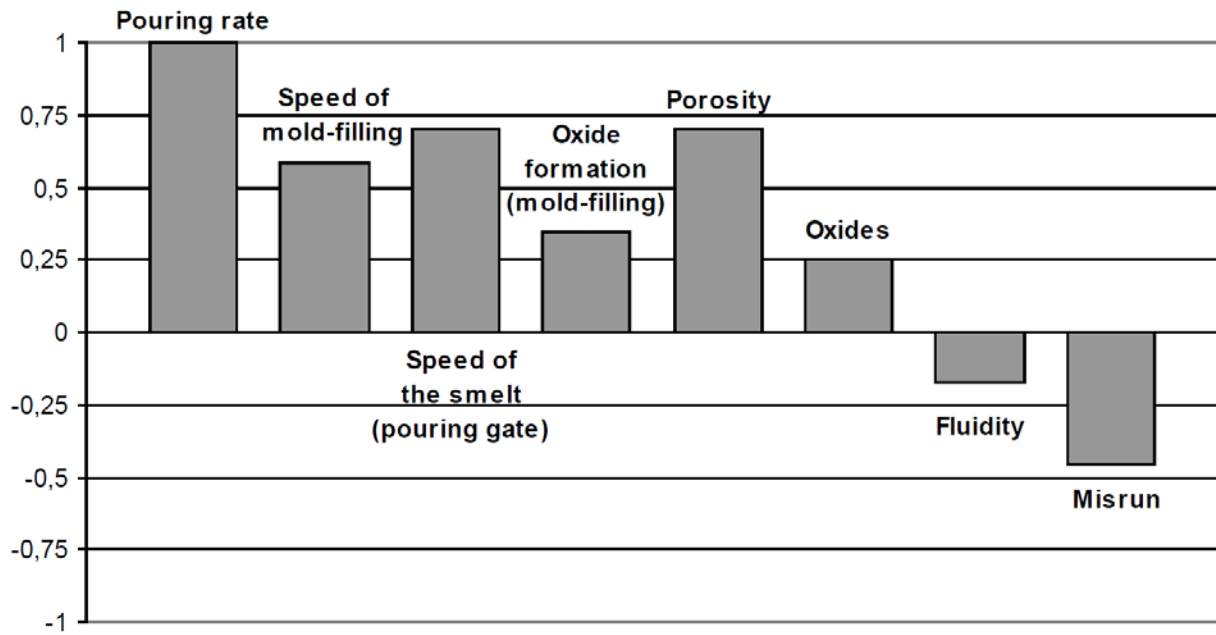


Fig. 6. Example "pouring rate – oxides, porosity, misruns"

4. Software

In order to facilitate and simplify the process of creating this cognitive model of casting defects' formations (and cognitive models in general) we use a software analytical system "Strategist" (Fig. 7).



Fig. 7. Logo of "Strategist".

This software is developed in the context of doctoral thesis at the institute of mechanical engineering at the technical university Freiberg and Volgograd state technical university. The „Strategist“ is used for visual representation of complex casual relations between factors during the analysis of complicated semi-structured problems. With the help of this software system all the available experience and knowledge of the factory employees and experts can be easily transferred into factors and interactions.

The software is designed for simulation of the further development of the systems situation with or without control actions. It is also assigned to simulate scenarios, to create and analyze different strategies, with are rather costly to realize in a real system (Fig. 8).

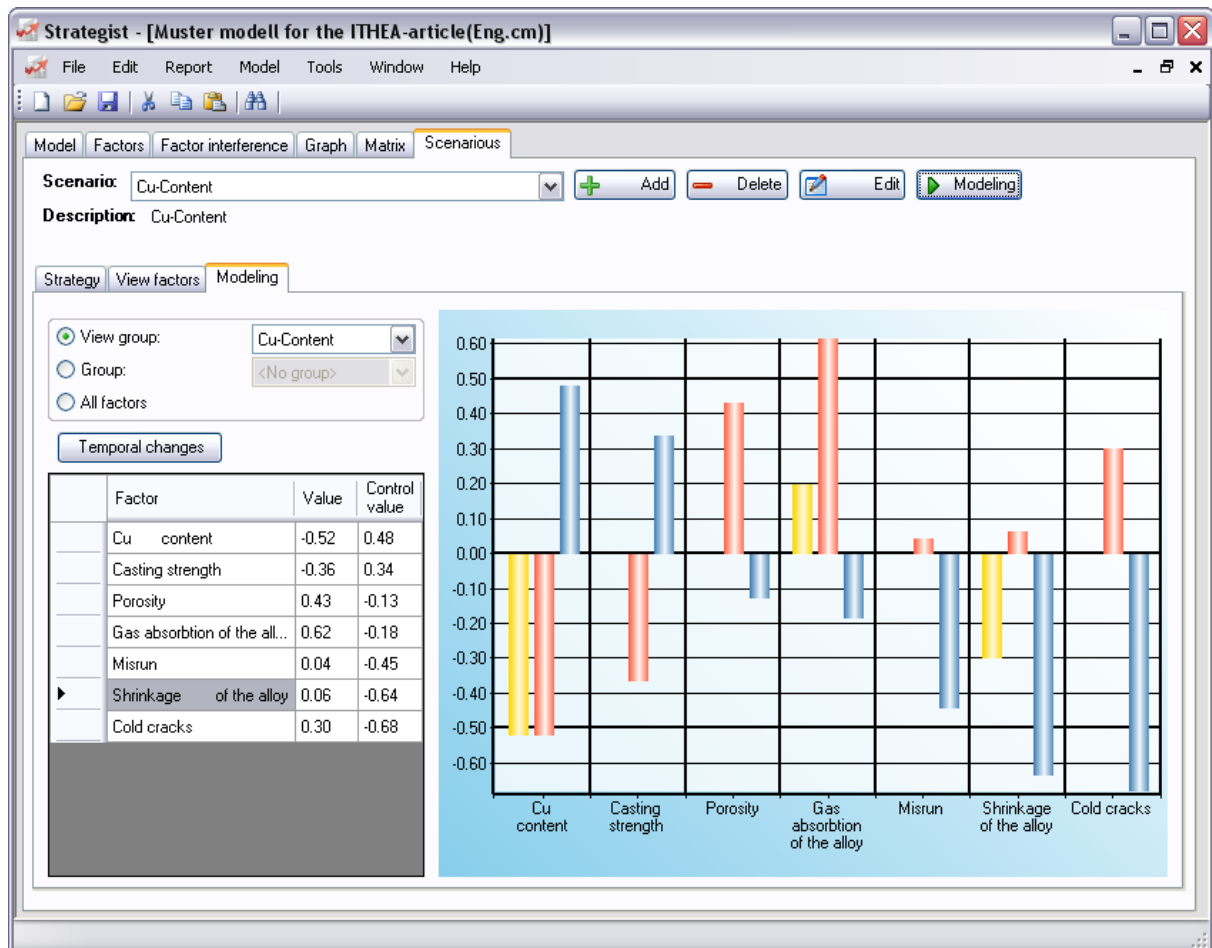


Fig. 8. Window of the simulation results in "Strategist"

Besides the "common" features of the cognitive simulation (representation of the model as matrix or graph, self-development, development with control, and so on) in "Strategist" are also realized some new like:

- determination of the interconnections between factors with the help of correlation analyses on the basis of the statistical information;
- determination of the common (direct and indirect) influences of one factor to all others and of the all factors to the selected one;
- visual coloration of the influences levels on the graph;
- search of all possible chains between 2 selected factors;
- Pareto-analyze of the interconnections between all the factors.

The user can choose from 4 different methods to build a cognitive map: using a graph, using connected lists of factors, using a matrix or using integrated library of factors.

5. Further Scientific Plans

With the help of the common cognitive map of the defects formation of the AlSi12 the authors are going to analyze indirect influences between factors deeper. The authors plan to analyze three types of the influences separately: "cause" – "casting defect", "casting defect" – "casting defect", "cause" – "cause" (Fig. 9). Such kind of relations has hardly ever been investigated and analyzed. But with the help of cognitive approach we have got a unique opportunity to see the real picture with all "cause-and-effect" relations – either direct or indirect.

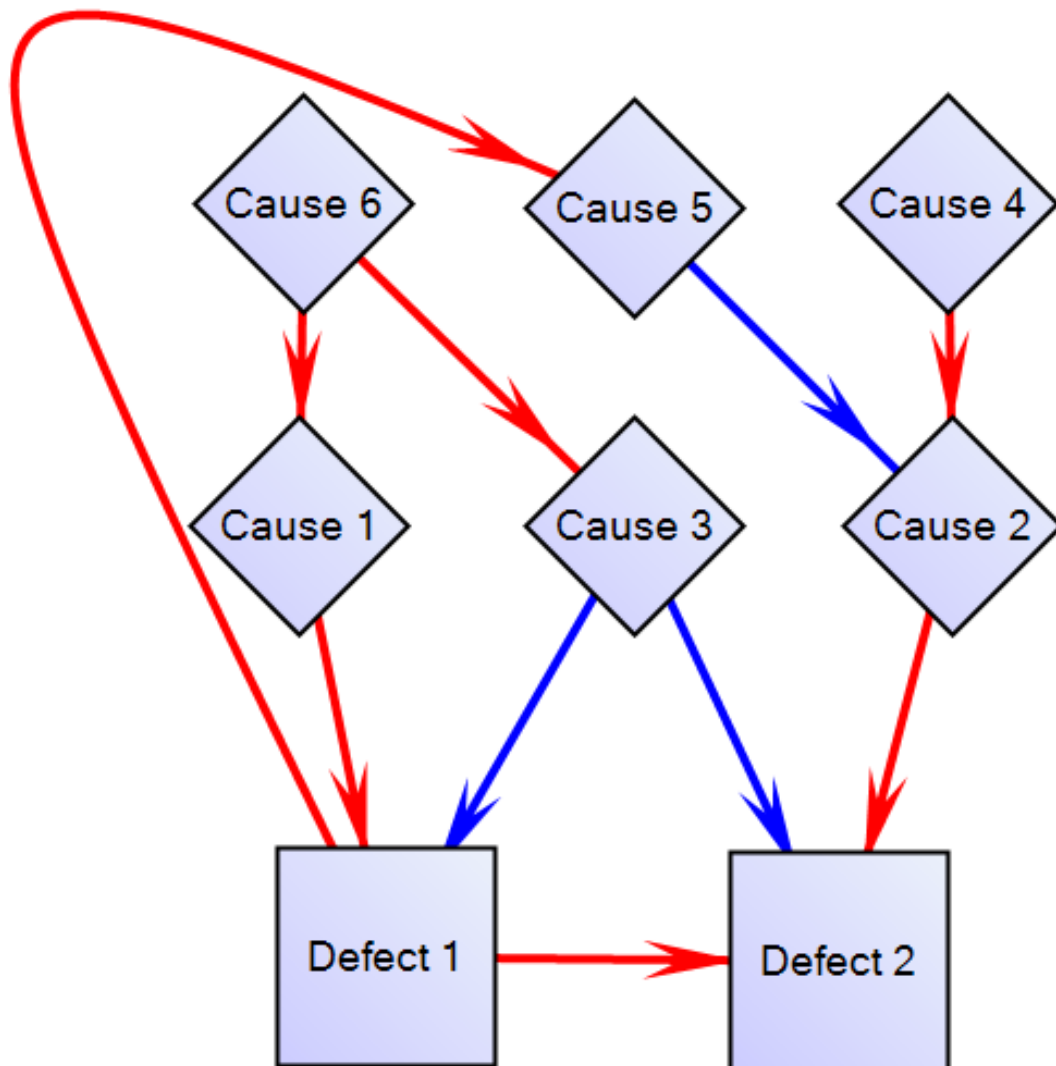


Fig. 9. Possible relations

Conclusion

So, the software system, which is being developed and improved, could be used by foundry men in their every day practice and by experts or managers in making fast decisions without a need to conduct long tests and complicated researches.

One of the main advantages of the proposed method is that a foundry man can extend a cognitive model and improve results of simulation himself (without any help from the outside), according to his own knowledge and

experience about castings formation and their avoidance. The use of the cognitive approach helps us to manage a large quantity of factors. It is highly competitive in the situations, when other methods could either be very expensive or do not show the logic of the occurred processes. This method will help foundry men to reveal the mechanism and the main reasons of casting defects' formation and take preventive measures in time.

Nowadays authors make researches in different areas of the foundry process:

- aluminium die casting;
- penetration defects by cast iron in sand forms;
- aluminium high pressure casting.

The authors continuously improve the software adding new tools and functions to make the cognitive approach widely available.

Acknowledgements

The authors would like to acknowledge financial support, provided by the DAAD (Deutscher Akademischer Austausch Dienst - German Academic Exchange Service) in the form of research grants.

Bibliography

- [VDS, 1988] Aluminium Gusslegierungen. VDS – Vereinigung deutscher Schmelzhütten. ISBN 3-87260-092-3. 5. Auflage 1988.
- [Drossel, 1999] G. Drossel, S. Friedrich, etc. Aluminium-Zentrale e.V., Aluminium-Taschenbuch. 15. Auflage. Band 2: Umformen von Aluminium Werkstoffen... ISBN 3-87017-242-8. 1999.
- [Altenpohl, 1994] D. Altenpohl. Aluminium von innen. Das Profil eines modernen Metalles. 5. Auflage. Aluminium-Verlag, Düsseldorf, 1994. ISBN: 3-87017-235-5.
- [Diligensky, 2004] Diligensky N.V., Dimova L.G., Sevastyanov P.V. Nechetkoe modelirovanie i mnogokriterialnaya optimizaciya proizvodstvennih system v usloviyah neopredelennosti: tehnologiya, ekonomika, ekologiya. Moskva, «Izdatelstvo Mashinostroenie n. 1», 2004
- [Tolman, 1948] E. C. Tolman. Cognitive maps in rats and men. Psychological Review, 55, 189-208, (1948).
- [Axelrod, 1976] Axelrod, Roberts (ed.), Structure of Decision (Princeton, NJ: Princeton University Press, 1976).
- [Kosko, 1992] Kosko, Bart, Neural networks and fuzzy systems, a dynamical systems approach to machine intelligence (Englewood Cliffs, NJ [u.a.], 1992)
- [Roberts, 1986] Roberts, Fred, Discrete mathematical models with application to social, biological and environmental problems (Rutgers University, Prentice-Hall, Inc, Englewood Cliffs, NJ, 1986)
- [Brunhuber, 1958] Ernst Brunhuber. Leichtmetall- und Schwermetall- Kokillenguss. Fachverlag Schiele und Schön GmbH, Berlin. 1958
- [Schneider, 1986] Philipp Schneider. Kokillen für Leichtmetallguß, Gießerei-Verlag. 1986.
- [Irmann, 1959] Dr.-Ing habil. Roland Irmann. Aluminiumguss in Sand und Kokille. Aluminium-Verlag GmbH Düsseldorf 1959
- [Polyakova, 2005] Polyakova I.A., Kamaev V. A., Zabolotsky M. A., Tikhonin A. V., region's situation analysis and development management with the help of cognitive modeling IT in science, education and business: Works of international conference, Gurzuf, 2005
- [Frank, 1993] Gregory, Frank: Cause, Effect, Efficiency and Soft Systems Models, in: The Journal of the Operational Research Society, Vol. 44, No. 4, New Research Directions (Apr. 1993), S. 333-344

Authors' Information



Polyakova, Irina – – PhD student; Institute of mechanical engineering Technical University Bergakademie Freiberg, Bernhard-von-Cotta-Strasse 4, D-09596 Freiberg; e-mail: ipolyakova@gmail.com



Bast, Jürgen – Professor Dr. Eng., Head of the Department, Institute of Mechanical Engineering Technical University Bergakademie Freiberg, Bernhard-von-Cotta-Strasse 4, D-09596 Freiberg; e-mail: bast@imb.tu-freiberg.de



Kamaev, Valeriy – Professor Dr., Head of the CAD Department, Volgograd State Technical University; e-mail: cad@vstu.ru



Kudashov, Natalia – PhD student; Institute of mechanical engineering Technical University Bergakademie Freiberg, e-mail: natalia@kudaschov.de



Tikhonin, Andrey – PhD student; Volgograd State Technical University; e-mail: andrey.tikhonin@gmail.com

EXTENDED ALGORITHM FOR TRANSLATION OF MSC-DIAGRAMS INTO PETRI NETS

Sergiy Kryvyy, Oleksiy Chugayenko

Abstract: *The article presents an algorithm for translation the system, described by MSC document into ordinary Petri Net modulo strong bisimulation. Only the statical properties of MSC document are explored – condition values are ignored (guarding conditions are considered always true) and all loop boundaries are interpreted as $\langle 1, \text{inf} \rangle$. Obtained Petri Net can be later used for determining various system's properties as statical as dynamic (e.g. liveness, boundness, fairness, trap and mutual exclusion detection). This net regains forth and back traceability with the original MSC document, so detected errors can be easily traced back to the original system. Presented algorithm is implemented as a working prototype and can be used for automatic or semi-automatic detection of system properties. The algorithm can be used for early defect detection in telecommunication, software and middleware developing. The article contains the example of algorithm usage for correction error in producer-consumer system.*

Keywords: *MSC, Petri Net, model checking, verification, RAD.*

ACM Classification Keywords: *D.2.4 Software/Program Verification - Formal methods, Model checking*

Introduction

The growing topicality of modern software systems and the rapidity imposed by pressing time-to-market demands for new approaches to the development process of high-performance and low-cost software. Namely, design productivity should be improved by means of new methodologies implementation due to the increase of the complexity of the systems. Formal methods begin to play a crucial role during design process. The use of formal methods during the initial stages of the development process can help to improve the quality of the later software, even if formal methods are not used in subsequent phases of development.

Protocol design is one of the most critical problems in distributed communication systems. Effective design methodology for protocol design requires formal models which are able to capture the inherent aspects of a system specification and verification tools that allow the designer to verify that a system satisfies its specification, to check the correctness of the system specification, and quickly explore alternative solutions.

Backgrounds

MSC is a modeling technique that uses a graphical interface, which was standardized by ITU (International Telecommunication Union) [ITU-TS, 2000], [ITU-TS, 2001]. It is usually applied to applications of the telecommunication domain, since they have properties of distributed reactive real-time systems. MSC diagrams are widely used in the early design stages of the systems development to capture system requirements. So, MSC is extremely suitable to capture the scenarios that a designer might want the system to exhibit (or avoid). MSC describes message flow between the instances, which present asynchronously communicating objects of the system or system entities like blocks, services or processes of the system. One MSC diagram describes a certain portion of system behavior or a scenario of communication between the instances. The set of the scenarios

(MSC-diagrams) are captured as requirements which constitute a complete behavioral description of the system. Let us describe briefly the most basic MSC-constructs.

Instances and Messages. Instance is a basic primitive of MSC, which in graphics is presented as vertical line with its name. Message transmissions, which are acts of communication between instances, are presented by horizontal arrows with possible curve or tilt under angle for reflecting "overtaking" or "intersection" of messages. The beginning of the vector marks a sending of the message and its ending marks receiving of the message. Events of sending and receiving of the messages are ordered along the instances so that sending of the message always happens earlier than its receiving. There is one more rule in standard MSC'2000 [ITU-TS, 2000] for ordering events along the instances: everything located above happens earlier than that located below (except **coregion** part, where events are not ordered).

Creation and termination of instances may be specified within MSCs. An instance may be created by another instance. No message events before the creation can be attached to the created instance. The instance stop is the counterpart to the instance creation, except that an instance can only stop itself whereas an instance is created by another instance.

Conditions. Condition is used as for restricting or defining a set of MSC traces through indicating states of the system so for defining the composition of one MSC diagram from the several MSCs. Namely, the standard MSC'2000 [ITU-TS, 2000] defines conditions of two types: setting condition and guarding condition.

Conditions of the first type are those which describe the current global system state (global condition), or some non-global state (nonglobal condition). In the latter case the condition may be local, i.e. attached to just one instance. Instances presenting dynamic objects can be began and finished, so far globality of a state considers dynamically changing set of instances.

Conditions of the second type restrict behavior of the MSC to execution of events in a certain part of MSC depending on the value of the given guarding condition.

Besides of composition role, conditions according to the standard [ITU-TS, 2000] are the means of events synchronization. For example, if two instances share one and the same condition, then for each message between these instance its sending and receiving events shall happen both before or both after setting of the condition. If two conditions are ordered directly sharing the common instance, or indirectly through conditions on other instances, then this order must be respected on all instances that share these two conditions.

General Ordering. General ordering is used to impose additional orderings upon events that are not defined by the normal ordering given by the MSC semantics. For example, it may be used to specify that an event on one instance must happen before an otherwise unrelated event on another instance.

There cannot be both upwards and downwards steps on the same ordering. This means that it may consist of consecutive vertical and horizontal segments. The textual grammar defines the partial order relations by the keywords before and after. They indicate directly in what order the involved events must come in the legal traces.

Inlines. Inline expressions used to create various composition of events inside MSC diagram. They allow creating of alternative, parallel, loop compositions and exceptional and optional regions. Last two are special cases of alternative composition.

Environment and Gates. The gates represent the interface between the MSC and its environment. Any message or order relation attached to the MSC frame constitutes a gate. Due to possibility of MSC nesting, gates can be defined for MSC diagrams, MSC diagram references (reference expressions) and for inlines. For gate connections the associated gate definition must correspond with the actual gate. Message gates are used for message events and order gates are used for causal ordering.

Order gates represent uncompleted order relations where an event inside the MSC will be ordered relative to an event in the environment. Order gates are always explicitly named. Order gates are considered to have direction - from the event ordered first to the event coming after it. Also order gates are used on references to MSC diagrams (MSCs) in other MSCs. The actual order gates of the MSC reference are connected to other order gates or to events.

Message gates define the connection points of messages with the environment. The message gates are used when references to the MSC are put in a wider context in another MSC. A message gate always has a name. The name can be defined explicitly by a name associated with the gate on the frame. The actual message gates on the MSC reference are then connected to other message gates or instances. Similar to gate definitions, actual gates may have explicit or implicit names.

MSC Semantics. MSC is a language with formally defined semantics, which is based on the process algebra. Applying this formal semantics to MSC a process term can be derived for each MSC and each MSC specification. MSC language semantics based on process algebra was defined first for textual representation of MSC diagrams in the form of expressions of process algebra that was called denotation semantics. Operational semantics is defined via transitional rules added to algebraic expressions.

Petri Net. Ordinary Petri net is used as a formal model to define the semantics of MSC system and support analysis [Kryvyi, 2007], [Kryvyi, 2007].

Definition 1. A net is a triple $N = (P, T, F)$, such that P and T are disjoint sets of places and transitions respectively, and $F \subseteq (P \times T) \cup (T \times P)$ is binary incidence relation between places and transitions (flow relation).

On the basis of incidence relation F (flow relation) characteristic function $\bar{F} : (P \times T) \cup (T \times P) \rightarrow N$ is introduced, where N is the set of natural numbers.

The sets $\bullet x = \{y / yFx\}$ and $x^\bullet = \{y / xFy\}$ denote the pre- and post set of arbitrary element $x \in P \cup T$ of net.

The following three conditions are required for the net $N = (P, T, F)$:

- C1) $P \cap T = \emptyset$,
- C2) $(F \neq \emptyset) \wedge [(\forall x \in P \cup T)(\exists y \in P \cup T) : xFy \vee yFx]$,
- C3) $(\forall p_1, p_2 \in P) (p_1 = p_2 \wedge p^1 = p^2 \Rightarrow p_1 = p_2)$.

Definition 2. A marking of the net $N = (P, T, F)$ is the function $\mu : P \rightarrow N$. The equation $\mu(p) = k \in N$ means that place $p \in P$ has k tokens.

Definition 3. Petri net (PN) is a triple (N, μ_0, W) , comprising a certain net N , certain initial marking μ_0 , and W is weight function (or multiplicity of an arc).

A transition $t \in T$ is enabled at a marking μ of PN (N, μ_0) iff $\forall p \in \bullet t : \mu(p) \geq \bar{F}(p, t)$. Such a transition can be fired, leading to the marking μ' according the following rule: $\forall p \in P : \mu'(p) = \mu(p) - \bar{F}(p, t) + \bar{F}(t, p)$. A sequence of transitions $\sigma = t_1, t_2, \dots, t_n$ is an occurrence sequence of a PN iff there exist marking $\mu_0, \mu_1, \dots, \mu_n$ such that $\mu_0 \xrightarrow{t_1} \mu_1 \xrightarrow{t_2} \dots \xrightarrow{t_n} \mu_n$. It is said that μ is reachable from μ_0 by the occurrence of σ . The marking obtained by enabled sequences are said to be reachable.

A PN is said to be safe if any reachable marking has at most one token at each place.

A PN is said to be ordinary if all of its arc weights are 1's.

Definition 4. Marked Petri net is a pair (N, Σ) , where N — Petri net and $\Sigma: T \rightarrow A$ — markup function over the alphabet A . If Σ is a partial function, unmarked transition are called " λ -transitions" and marked by the "empty symbol" λ .

Algorithm of Translation of MSC Document to Petri Net

MSC subset used by algorithm. Presented algorithm processes the MSC 2000 constructions set with the following limitations:

1. Time constraints are ignored.
2. Timers are processed only as separate events.
3. MSC references are allowed only in HMSC.
4. MSC reference expressions are not processed.
5. Loop boundaries are ignored as treated as $\langle 1, \text{inf} \rangle$ (or $\langle 0, \text{inf} \rangle$ by user's choice).
6. Sequential MSC diagram connection is treated as strong sequence (i.e. all events in the first diagram shall be finished before the second one will start).

Algorithm assumes that source MSC document is syntactically and statically correct. Relation between MSC diagrams shall be given explicitly by HMSC.

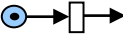
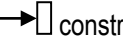
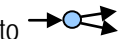

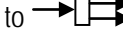


Algorithm Description

Stage1 (Building of Trace Graph). During the first stage of the algorithm the Trace Graph will be build. This graph represents the traces, which a structurally possible in original MSC document (save loop iterations, which will be added during the next stage). Trace Graph creation consists of the following steps:

1. HMSC start construction translates to «start» node.
2. HMSC end construction translates to «end» node.
3. HMSC points of alternative branching translates to «alt-in» nodes.
4. HMSC points of branch joining translates to «alt-out» nodes.
5. HMSC par frame translates to «par-in» and «par-out» pair of nodes.
6. HMSC lines translates to edges between corresponding nodes.
7. MSC references in HMSC translates according the following rules (8 — 19).
8. For each MSC diagram created the nodes pair «msc-in» and «msc-out». This pair forms new synchronization zone. All edges, which corresponds to HMSC lines, drawn to this MSC reference, connect to the «msc-in» node. All edges, which corresponds to HMSC lines, drawn from this MSC reference, connect to the «msc-out» node.
9. Nodes, which corresponds to instance create events of MSC diagram, connect with «msc-in» node.
10. Nodes, which corresponds to instance end events of MSC diagram, connect with «msc-out» node.
11. Inline of «exc» and «opt» types translates as corresponding «alt» inlines.
12. Inline of «alt» type translates to «alt-in» and «alt-out» pair of nodes. Each of alternatives in this inline forms new synchronization zone.
13. Inline of «part» type translates to «par-in» and «par-out» pair of nodes. Each of alternatives in this inline forms new synchronization zone.

14. Inline of «loop» type translates to «loop-in» and «loop-out» pair of nodes. Inline body forms new synchronization zone.
15. All edges, which represent MSC lines, drawn to inline, connect to the corresponding «-in» nodes. All edges, which represent HMSC lines, drawn from inline, connect to the corresponding «-out» nodes.
16. Coreions are treated as par inline for all it's events.
17. All other valid MSC events translates to graph nodes; invalid events are skipped.
18. Edges which represent the order, explicitly shows in MSC diagram, added to graph.
19. Gates between MSC diagrams and between inlines resolved and new edges, which corresponds to gated messages and gated order relation, are added to graph.
20. In each synchronization zone edges, which do not correspond to domination relation, are removed. (Nested synchronization zone is treated as one node for outer zone.)

Stage2 (Building of Petri Net). On the second stage Trace Graph is used for resulting Petri Net building. Petri Net creation consists of the following steps:

1. Node «start» translated to  construction. Transition is marked by λ .
2. Node «end» translated to  construction. Transition is marked by λ .
3. Node «alt-in» translated to  construction. Number of out edges corresponds to number of alternatives.
4. Node «alt-out» translated to  construction. Number of in edges corresponds to number of alternatives.
5. Node «par-in» translated to  construction. Number of out edges corresponds to number of alternatives. Transition is marked marks by λ .
6. Node «par-out» translated to  construction. Number of in edges corresponds to number of alternatives. Transition is marked by λ .
7. «msc-in» and «msc-out» nodes are translated to transitions and marked by λ .
8. «loop-in» and «loop-out» pair translated to  construction. Transition is marked by λ . (If user choose $\langle 0, \text{inf} \rangle$ loop boundaries, additional pass-through transition with λ mark added.)
9. Rest of graph nodes are translated to transitions marked by names of source MSC document elements.
10. Edges of the Trace Graph are translated to edges in Petri Net, which connects the corresponding nodes. If such edge connects transition with transition, addition place inserted in this edge.

Example

Let's show the algorithm work and usage on simple example. One of the simplest producer-consumer models as shown on the next MSC document with one MSC and one HMSC diagram (figures 1 and 2).

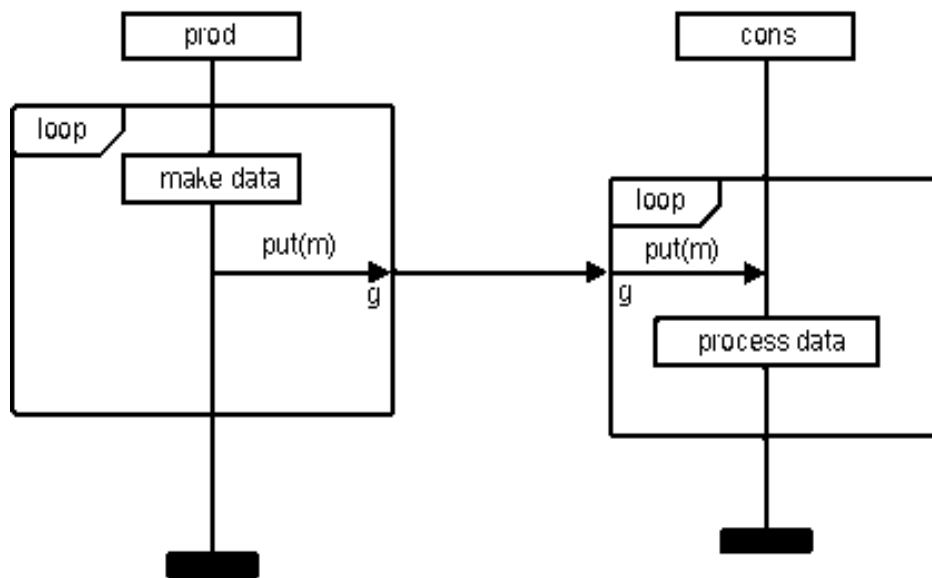


Fig. 1.

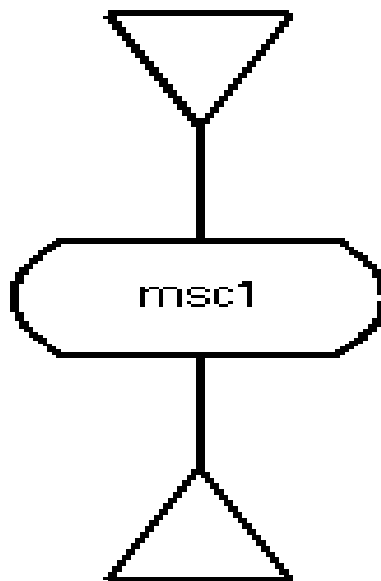


Fig. 2.

This diagram has two instances — producent (prod) and consument (cons). Producent infinitely produces some date (action «make data» in loop) and sends it to consument by message put(m) through gate g. Consument also has a loop, which allows it to infinitely process data («process data» action). After performing steps 1 — 19 we obtain the graph, shown on figure 3 and, after removing the non-dominated edges (Step 20), on figure 4.

Then, after performing the Step 2 of algorithm, we obtain PN, shown on figure 5.

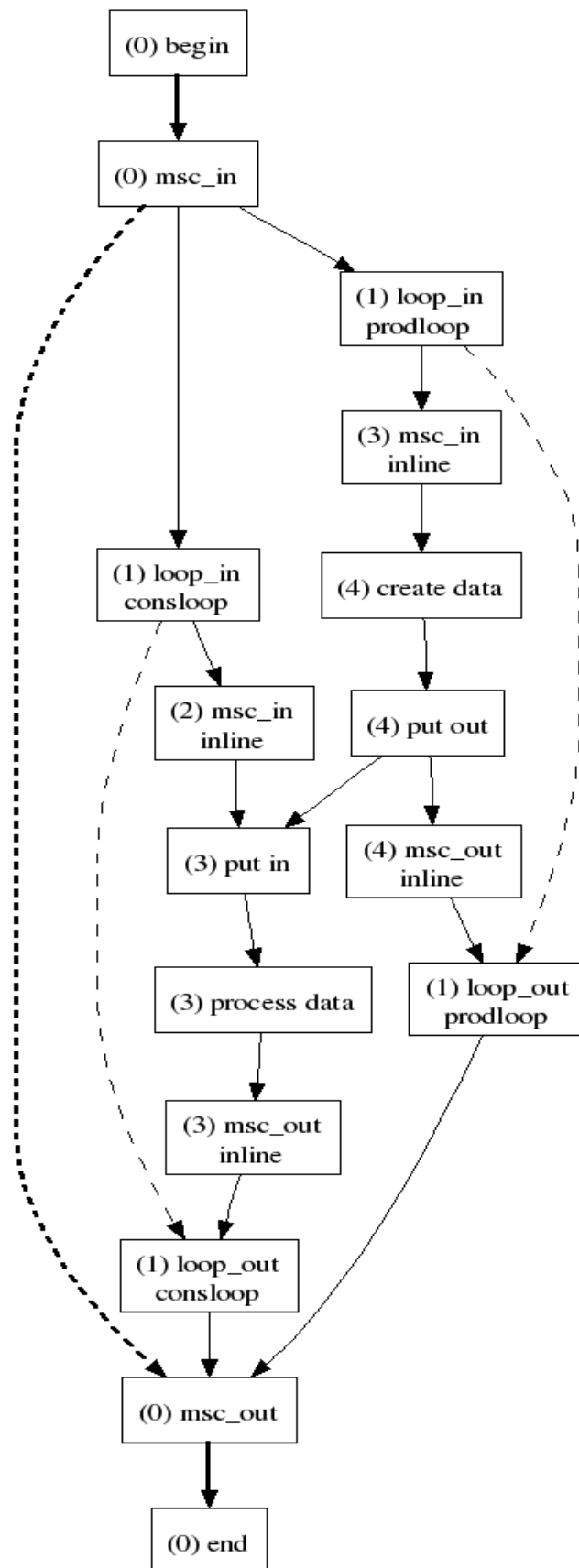


Fig. 3

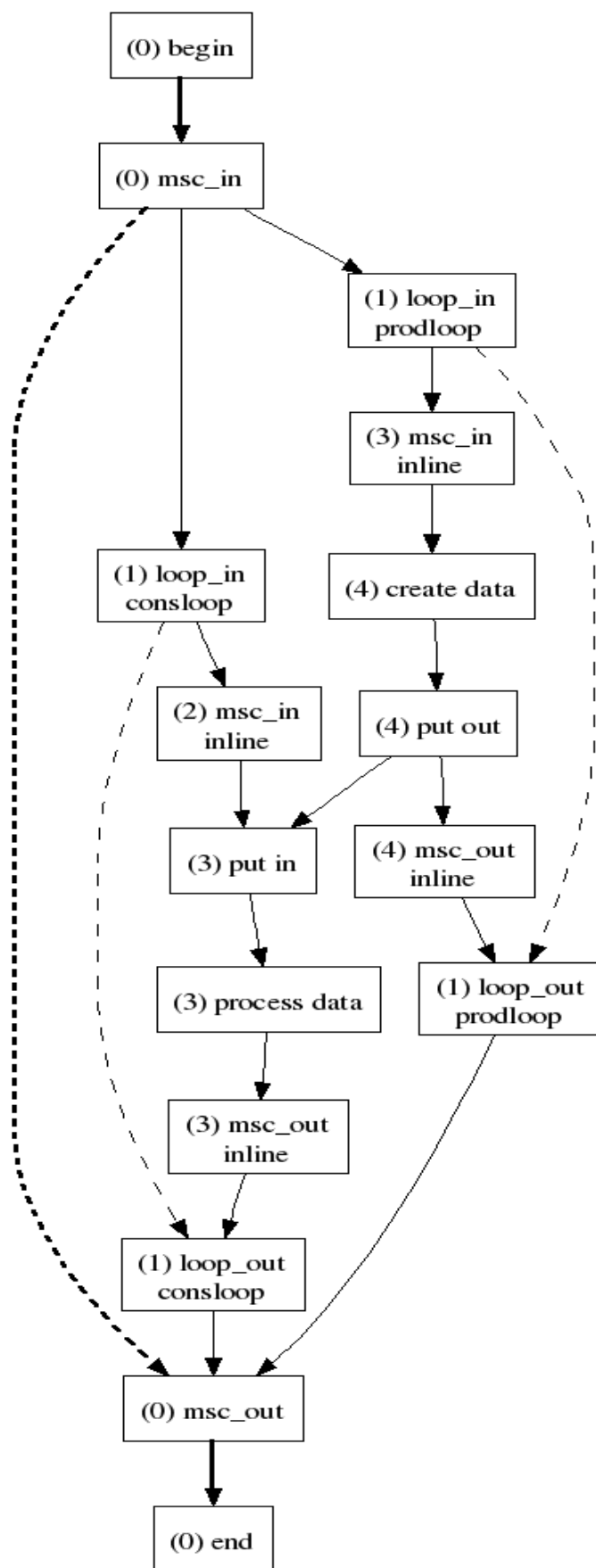


Fig. 4

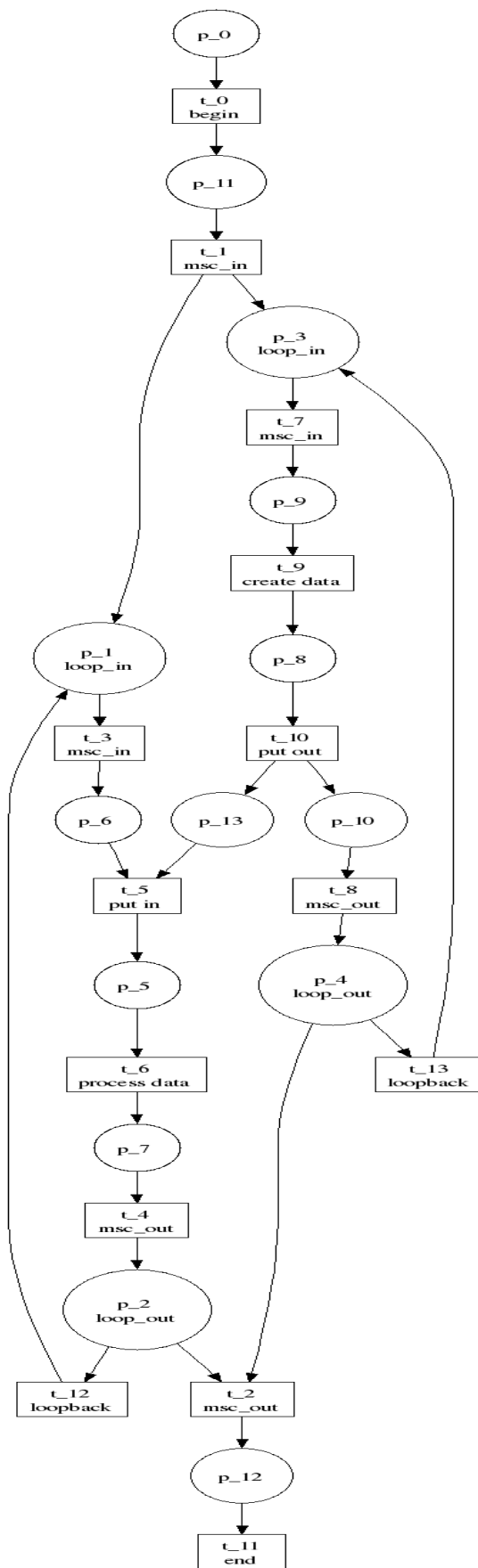


Fig. 5

Now, let's try to analyze our system. First of all check liveness; to do this, calculate incidence matrix of obtained PN and find it's T-invariants. Incidence matrix is:

```

-100000000000000
010-100000000010
00-1010000000-10
0100000-1000001
00-10000010000-1
000001-10000000
00010-100000000
0000-1010000000
0000000001-1000
000000010-10000
00000000-101000
1-10000000000000
001000000000-100
00000-100001000

```

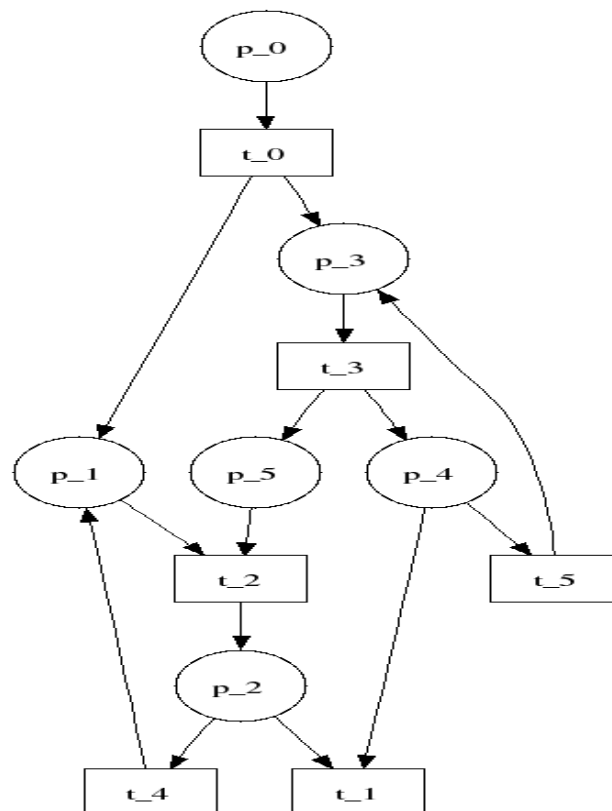


Fig. 6

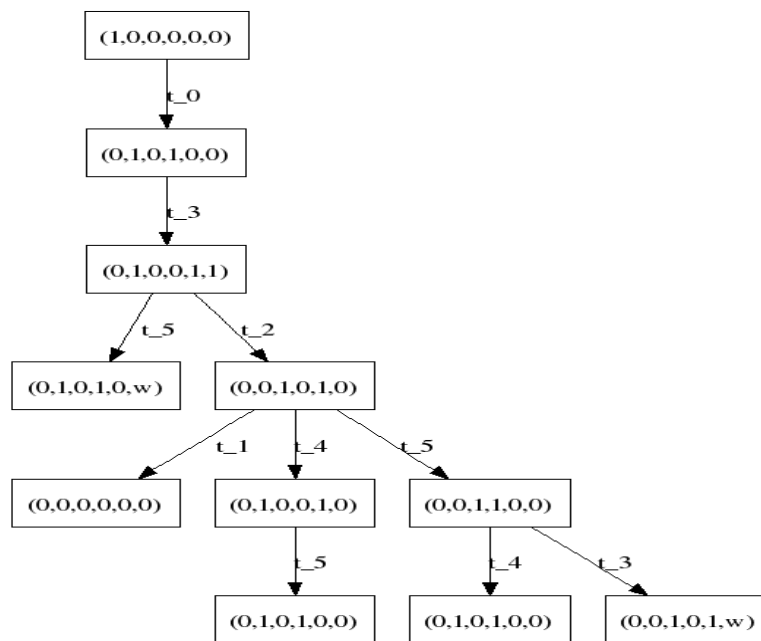


Fig. 7

And truncated solution set of the equation $Ax=0$ is $(0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1)$, which shows, that transitions in our PN are alive except t_0 (begin MSC document), t_1 (begin MSC diagram) and t_{11} (end MSC document) — just as expected. So, all events in the source can be executed infinitely. Then, check for PN's boundness. To perform this check, reduce PN as shown in [Murata, 1989] and create it's covering tree (figures 6,7).

As we can see on covering tree, place p_5 of reduced PN is unbounded. It corresponds to place p_{13} of original PN and means, that infinite number of messages $put(m)$ can be stacked in original MSC (it can happens if consumer process data slower than producer creates it). To fix this, let's change original MSC by adding synchronization between consumer and producer (figure 8):

Now let's see on reduced PN and covering tree for corrected document (other steps have been omitted to save the article space).

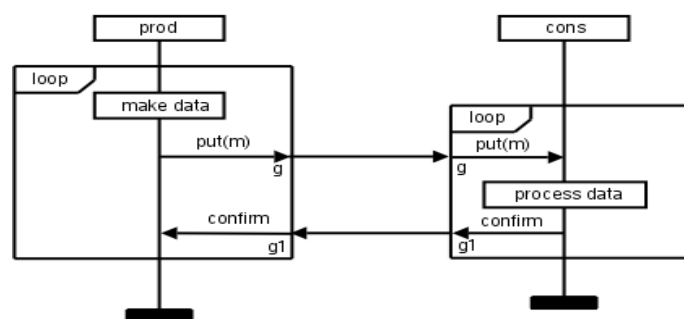


Fig. 8.

Now we can see (figure 10) that our PN is bounded, so synchronization was enough to fix the found problem.

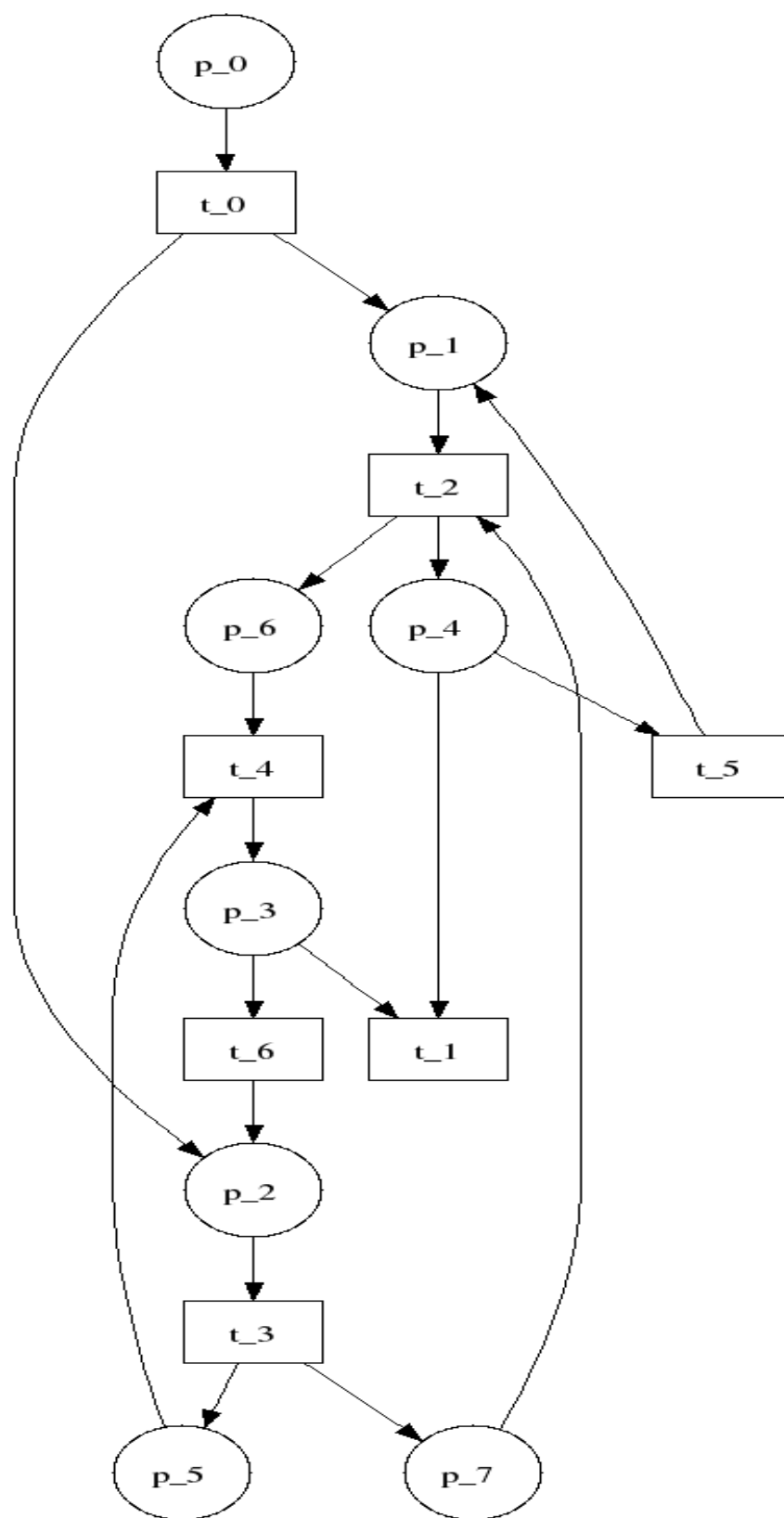


Fig. 9

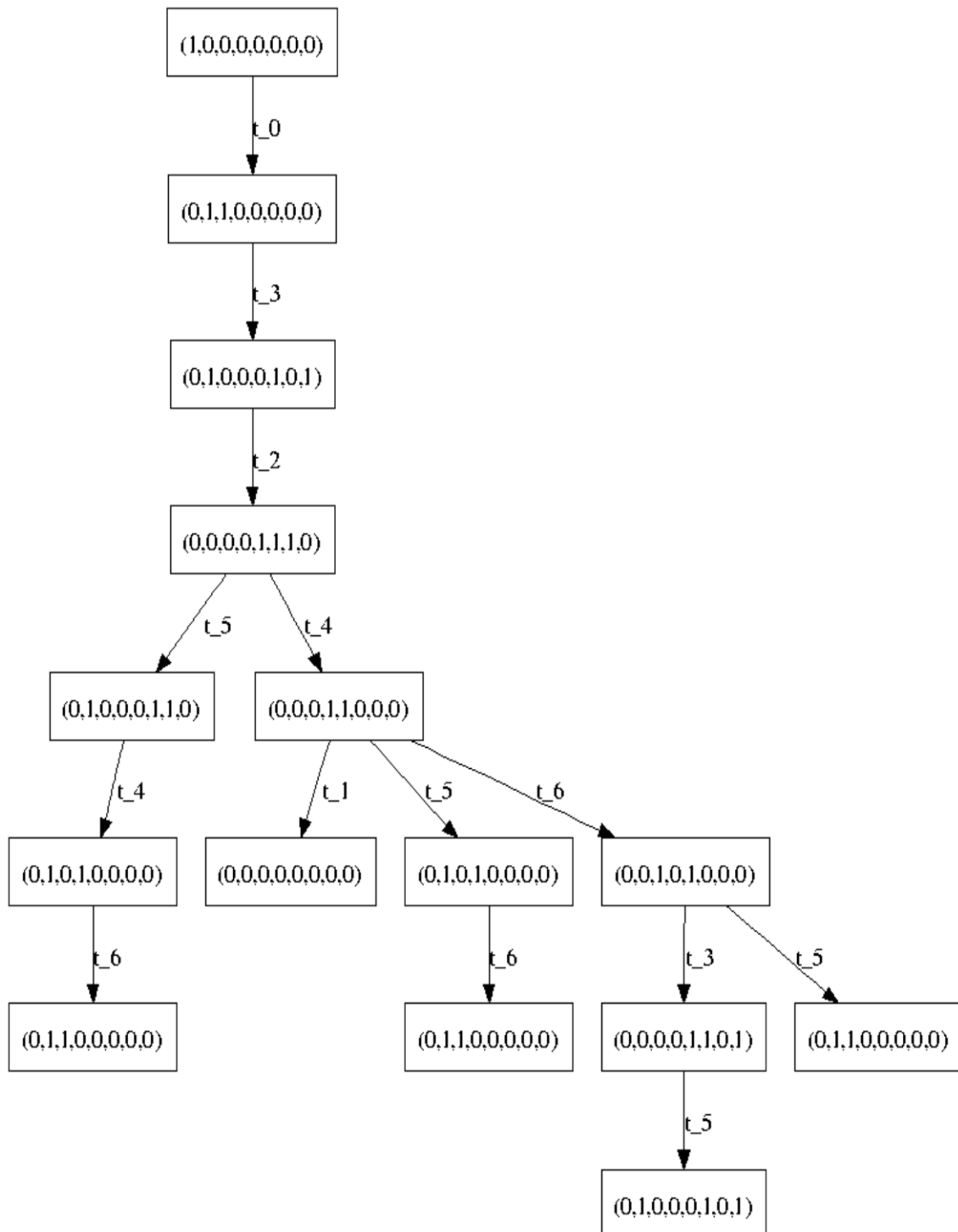


Fig. 10

Conclusion

For conclusion we note, that the most significant feature of the given algorithm is its increased usability. The input set of MSC elements has been extended. Most of the existing design approaches require decomposing from the beginning the overall functionality into components. Extended input set of MSC elements in the given version of the translation algorithm makes it possible to apply different composing/decomposing techniques for subsequent PNs analysis.

The algorithm of translation MSC diagrams into Petri net, presented in the paper, considers only the subset of MSC language, therefore full implementation of reference MSC expressions and processing time constraints are our further research direction.

Bibliography

- [ITU-TS, 2000] ITU-TS Recommendation Z.120: Message Sequence Chart (MSC). ITU-TS, Geneva (2000).
- [ITU-TS, 2001] ITU-TS Recommendation Z.120 Annex B: Message Sequence Chart Annex B: Formal semantics of Message Sequence Charts. ITU-TS, Geneva (2001).
- [Kryvyi, 2007] Kryvyi, S., Matveyeva, L.: Algorithm of Translation of MSC-specified System into Petri Net. *Fundamenta Informaticae*, Vol.79 (2007), 1–15.
- [Kryvyi1, 2007] Kryvyi, S., Matveyeva, L., Chugayenko O.: Extension of Algorithm of Translation of MSC specified system into Petri Net. *Proc. of the CS&P'2007*, 2007 – v2 – p.376–388
- [Murata, 1989] Murata T. Petri Nets: Properties, Analysis and Verification. *Proc. of the IEEE*, 1989 — 77 — N4, p.65–74

Authors' Information



Kryvyi Sergii – *Glushkov Institute of Cybernetics NAS Ukraine, Ukraine, Kiev, 03187, 40 Glushkova Street, Institute of Cybernetics, e-mail: krivoj@i.com.ua*

Major Fields of Scientific Research: Verification, Simulation, Discrete systems, Diophantine equations.



Chugayenko Oleksiy – *Institute of Cybernetics NAS of Ukraine, Ukraine, Kiev, 03187, 40 Glushkova Street, Institute of Cybernetics, e-mail: avch@avch.org.ua*

Major Fields of Scientific Research: Automatic reasoning, Petri Nets, Software technologies.

METHODS OF COMPARATIVE ANALYSIS OF BANKS FUNCTIONING: CLASSIC AND NEW APPROACHES

Alexander Kuzemin, Vyacheslav Lyashenko

Abstract: *General aspects of carrying out of the comparative analysis of functioning and development of banks are considered. Classical interpretation of an estimation of efficient control by bank from the point of view of interrelation of its liquidity and profitableness is considered. Questions of existential dynamics in a system of comparative analysis of difficult economic processes and objects are generalised.*

Keywords: *bank, analysis, microsituation, statistical conclusion, nonlinear dynamics, Wilcoxon criterion.*

ACM Classification Keywords: *H.4.2. Information system Applications: Types of Systems Decision Support*

Introduction

When considering directions of solving different aspects of analysis of the market economy objects spatial-temporal dynamics the necessity arises, in any event, to carry out the comparison between the characteristics of the subjects being studied. The performance of such analysis is related, first of all, to elucidation of the arisen situation in the estimation of the economic entity functioning being studied and comparison of such object development with other similar market objects development. As a whole, this contributes not only to revealing of the reasonable approaches to the arising problems solution, but to the possibility to justify look-ahead actions in decision-making relating to the stable functioning and development of the economic entity being studied.

In this work the banks are chosen as an example of the specific economic entities. This is motivated by the fact that the stable and systematic development of the banking sector has a profound impact on the reproduction structure of economy, as through the banking the flow and reallocation of monetary resources and capital funds are organized. At the same time, the analysis of the finance flows both of the banking system, as a whole, and individual banks, in particular, is one of the key components of building of the adequate economic security system of the economic entity operating in the market economy [1, 2]. Eventually, it is precisely this that defines the urgency of the given direction of the investigation, the importance and utility of its consideration as far as the banks is concerned.

Primary Purpose of Research

The basis for performance of the banks functioning comparative analysis involves, as a rule:

- the models based on generalization and consideration of the normative indices, coefficients of economic agents activity standards with the subsequent estimation of their rating [3, 4];
- the methods of statistical conclusion based on econometric models and methods having in their origin the game theory approaches [5, 6];
- the methods of the fuzzy sets theory [1, 7].

By widespread approaches, comparative analysis, as a rule, is:

- either relative generalization of dynamics of the corresponding activity indices of the economic entity being analyzed [8],

- or construction of cluster models which allow to rank the degree of development of the objects being compared [9, 10].

But in any case the mathematical basis of investigations consists, mainly, of the probabilistic methods for data analysis. In this case the solution key aspect consists in determination of the analyzed data distribution parameters with a view to obtain an adequate model, often this is connected with violation of the distribution normality law of the data sampling involving such objects. At the same time, the main problem, arising when constructing an adequate model of the banking comparative analysis, is connected with that the economic development laws assume the presence of such interaction between different subject of the market and account for the action on these interactions of various surroundings manifestations, not having a definite statistical nature in the classical meaning. Solution of the given problem is reached, in a way, through introduction of different aspects of information saturation, banking indices being considered, into the problem treatment. But in the given case another problem arises associated with the necessity to consider the procedure of various manifestations ranking of information saturation of one or other banking indices.

Thus, the openness of a question of construction or a choice of model of the comparative analysis of functioning and development of banks induces not only to carrying out of new researches directed on its decision, but also consideration of the reached results on the basis of construction of a generalised conceptual model.

Classical Interpretation of the Estimate of the Bank Management Efficient with Respect to Interaction of its Liquidity and Profitability.

One of approaches for carrying out of a comparative estimation of development, both bank system, and separate banks is comparison of their levels of liquidity and profitableness. A basis of such comparison reflects the interrelation between liquidity and profitableness of the bank activity which essence reveals that more risky bank operations can bring even higher incomes. Thus, when considering the probabilistic interpretation of banking activity management starting from a definite liquidity level one should take into account the fact that the bank tends to support the liquid assets volume at the level sufficient to ensure meeting previously taken commitments. At the same time the bank defines the probability of the need for loan resources to meet its commitments. Then, for example, the interpretation of the banking system development based on the liquidity analysis can be considered as a probability for a random two-dimensional value to penetrate into some specified field where acceptable and admissible liquidity and profitability levels parameters manifest themselves as boundaries of such a field.

At the same time the classical interpretation of the bank management efficiency can be considered in terms of the fuzzy sets theory. The given approach becomes possible through introduction into consideration of the ownership function of some set of the bank liquidity and profitability indices corresponding to a subset of efficient managing actions of the given indices.

Then, for example, the fuzzy interpretation of the bank management efficiency in the specified phase space is limited to building and estimation of the corresponding ownership functions characterizing the degree of reaching the bank efficient management in the specified variation intervals of the banking activity being analyzed. In this case it is expedient to choose a fuzzy interpretation of the intended parameters variations in the limits of the admissible values of liquidity and profitability indices presented in the probabilistic model by the corresponding probabilistic curve as a formal description of such functions. The advisability of such a transition is motivated by that the fuzzy formalization of the corresponding probabilistic curve is possible on the basis of the concept of the fuzzy number of L-R type, which in the given case can be regarded as a trapezoidal fuzzy number. Such an

interpretation of the ownership function makes it possible not only to describe the processes under investigation formally but to take into account existing economical aspects in their development.

Nevertheless, a prominent aspect of carrying out of the comparative analysis of functioning of banks is the account of dynamics of investigated processes that allows speaking about possibility of use of a method of space-time analysis in comparison of development of banks [2].

Space-time Dynamics in System of the Comparative Analysis of Difficult Economic Processes and Objects

The most simple and abundant example of the spatial-temporal dynamics in the economic systems may be considered a set of the data characterizing development of some process (phenomenon) in time having regard to variety of available economic managing subjects. By description of such processes (phenomena), in particular, is meant dynamics of different indices of the socioeconomic development of the country in connection of its separate regions or development of some sector of the economy taking into account functioning of its separate economic components. Dynamics of the banking sector of economy development both taking into account regional features of separate administrative territorial units of the country and presence of a definite number of economic managing subjects defining the corresponding activity in this or that region can exemplify such a description.

The spatial-temporal dynamics analysis in the economic systems amounts either to the cross-section regression, or to the temporal series regression. The first type of regression makes it possible to estimate the interconnection between different data being analyzed at a definite moment of time; the second type is the interconnection between the data of one (or several) parameter during some interval of time. In this case application of the first type of regression, as a rule, doesn't take into account the dynamics of data being analyzed, application of the second type of regression doesn't take into account the presence of interdependent influence between the studied parameters with respect to different economic managing subjects. In the total the generalized model of analysis can assume the structured form [2]:

$$Y = F(X_1, X_2, \dots, X_n) \Leftrightarrow \begin{cases} y^1 = f^1(X_1), \\ y^2 = f^1(X_2), \\ \dots \\ y^n = f^n(X_n), \end{cases} \quad (1)$$

or

$$Y = F(X_1, X_2, \dots, X_n) \Leftrightarrow \begin{cases} y^1 = f^1(x_1^1, x_2^1, \dots, x_n^1), \\ y^2 = f^2(x_1^2, x_2^2, \dots, x_n^2), \\ \dots \\ y^i = f^i(x_1^i, x_2^i, \dots, x_n^i), \end{cases} \quad (2)$$

where

Y – dependent variable characterizing some generalized its value;

$\{X_n\}$ – set of independent variables characterizing some generalized their values;

$F(\dots)$ – function representing the kind of regression dependence between the generalized values of variables being investigated;

y^n and y^i – dependent variables with regard to the analysis of the action of one independent variable X_n on the whole interval of time being investigated or taking into account analysis of the action of all independent variables $\{x_n^i\}$ for some definite interval of time;

$f^n(\dots)$ and $f^i(\dots)$ – function representing the kind of regression dependence between the dependent and independent variables represented with non-generalized values.

Such a representation of spatial-temporal dynamics in the form of the regression dependence makes it possible to present the interconnections existing between the data being analyzed in the combined-structured form and to investigate them in greater detail.

Nevertheless, it should be noticed that the basis of a method of analysis of space-time dynamics of difficult economic processes and phenomena builds the concept of a financial stream dominating in a number of approaches for carrying out of the comparative analysis of functioning of banks.

Analysis of Financial Streams as Basis of Conducting of Comparative Analysis of Functioning of Banks

As be marked before, for conducting of comparative analysis of functioning of banks an important instrument is the use of the finance flows, which makes it possible to give the most complete description of the banking on the basis of multiple presentation of the initial data (separate indices of activity) x_t^γ of their sets of γ at a certain temporal interval t in terms of the finance flows – $\{x_t^\gamma\}$.

This is associated with that the basis of the flow approach comprises the possibility to realize the structuring of data for complex dynamic systems; it is precisely the structuring that opens different directions for carrying out the necessary analysis [11].

At the same time, the flow processes involve all spheres of the market economy, this is rather important as far as the banks is concerned as the centers of redistribution of monetary and reallocation of capital. This also allows taking into account the degree of various environment factors action, governing thereby the information saturation of the indices being considered.

It should be noted in this case that the flow approach can serve not only as the set of instruments for the banks functioning and development, but also act as the combining center of various approaches applications for carrying such analysis.

At the same time one of the shortages of the flow approach consists in performance of the banks generalizing comparative analysis as the financial flow concept assumes only consideration of some sets of such flows while their structuring is also significant. Therefore, the following part of the given investigation is just devoted to the processes of more precise bank finance flows structuring.

Visualisation of Processes of the Comparative Analysis of Functioning of Banks

However, before to pass to consideration of the questions, concerning direct carrying out of the comparative analysis of functioning of banks, we will stop on a problem, concerning visualisation of the investigated facts and received results. Validity of such consideration is connected by that decision-making in area, a task in view of the given research, are connected with necessity of carrying out of the analysis and interpretation of the multidimensional data for a time scale approached to the real. Thus the purpose of such visualisation is, including identification of a current condition of investigated object (bank) during its life cycle on the basis of the intellectual

analysis of the data. Therefore, on a number with classical approaches about representation of the statistical data about investigated economic object (bank) in the form of diagrams and schedules, also it is necessary to use more difficult models of interpretation of the received data.

In particular one of perspective directions of research, concerning visualisation of carrying out of the comparative analysis of functioning of banks is the representation of results of such analysis in a kind of equal-distance points from the beginning used systems of co-ordinates in various metrics. The offered approach gives the chance to interpret results of the comparative analysis, as in the form of various geometrical figures, as well as various lines which reflect dynamics of the investigated data and change of such dynamics.

Thus the task of the beginning of co-ordinates (by means of a choice of the certain metrics) for equal-distance points with the subsequent transformation of the received curve (or figures) by means of group of affine transformations allows to carry out comparative analysis for the investigated objects forming one economic cluster. For example, if in two-dimensional space of signs of an income-expense scheme display an arrangement of various banks they form a curve which reminds one of ellipse quarters (fig. 1).

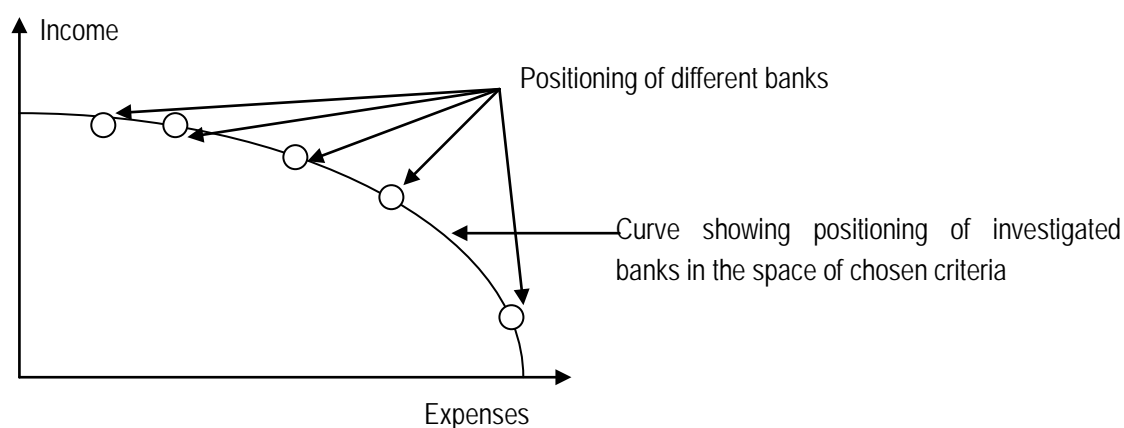


Fig. 1 shows the investigated objects (banks) as a curve in the space of chosen criteria (income – expenses)

In particular such representation (see fig. 1) allows not only to compare initial data (the received income and the realized expenses), but also to predict dynamics of change of probable profit in the framework of one model. The general toolkit for carrying out of the comparative analysis of functioning of various subjects of managing thus extends. On the other hand decomposition of the income and profit of each of banks allows to speak about certain situations in their development and therefore to apply concept of a microsituation to research of the chosen subjects of managing.

Microsituation Concept as the Foundation for Performance of the Banks Functioning Comparative Analysis and Development

Some problems of finance flows structuring for carrying out of the economical processes dynamics comparative analysis were considered in [1]. Nevertheless, the problematic aspects concerning comparison of special and general finance flows for their further structuring and analysis remain beyond the scope of investigations. First and foremost, such a generalization concerns, first of all, the problems of description of the situation of functioning of the banks as a complex system and the banks taken individually.

One of the specified problem solution directions can be the use of the microsituation concept which found the proper application when solving a number of problems arising in emergency situations [12].

In the given aspect, to perform the comparative analysis of the banks functioning and development based on the flow approach, by the microsituation, variety of the banking description with the help of the corresponding parameters and indices should be meant. In this case the concrete microsituation S^L can be described in the form of a separate finance flow or some set of them being defined with a set of data $\gamma, (\gamma = \overline{1, m})$, characterizing the banking of some bank $L, (L = \overline{1, n})$:

$$S^L = (\{x_t^{k1}\}^L, k1 \in \gamma, \quad (3)$$

$$S^L = (\{x_t^{k1}\}, \{x_t^{k2}\}, \{x_t^{k3}\})^L, k1, k2, k3 \in \gamma. \quad (4)$$

Thus, comparing banks between themselves we, first of all, compare the micro situations which in the given case describe the state of the banks functioning and development in terms of some parameter or their totality:

$$S^1 = (\{x_t^{k1}\}, \{x_t^{k2}\}, \{x_t^{k3}\})^1 \approx S^2 = (\{x_t^{k1}\}, \{x_t^{k2}\}, \{x_t^{k3}\})^2, 1, 2 \in L, \quad (5)$$

where S^1 – is the microsituation describing the first of the banks being analyzed,

S^2 – is the microsituation describing the second of the banks being analyzed.

At the same time it is possible to carry out comparison of the banks development and functioning as a whole fixing parameter t . Then, in the given case, variation of some of the banking parameters being analyzed x_{tp}^γ for a fixed date tp in terms of the whole variety of banks – $\{x_{tp}^\gamma\}^L, L, (L = \overline{1, n})$ is considered as a finance flow.

In this case the concrete microsituation can be presented in the following form:

$$S_{tp}^L = (\{x_{tp}^\gamma\}^L, tp \in t, \gamma, (\gamma = \overline{1, m}), \quad (6)$$

or

$$S_{tp}^L = (\{x_{tp}^\gamma\}, \{x_{tp}^\gamma\}, \{x_{tp}^\gamma\})^L, tp \in t, \gamma, (\gamma = \overline{1, m}). \quad (7)$$

Then the comparison consists in performance of the analysis between the microsituations describing the state of the banking system functioning as a whole at some fixed dates of time:

$$S_{tp1}^L = (\{x_{tp1}^\gamma\}, \{x_{tp1}^\gamma\}, \{x_{tp1}^\gamma\})^L \approx S_{tp2}^L = (\{x_{tp2}^\gamma\}, \{x_{tp2}^\gamma\}, \{x_{tp2}^\gamma\})^L, tp1, tp2 \in t. \quad (8)$$

Ultimately, we receive some set of microsituations $\Omega = \{S^L, S_{tp}^S\}$, completely describing functioning and development of the banking system. Since, as mentioned above, not all microsituations can have the normal distribution, then we shall consider nonparametric tests to verify the hypothesis for coincidence of the microsituations being investigated. In the given case it is expedient to use the test Wilcoxon for bound samplings [13], which answers the question: whether some event essentially changing the microstructure hierarchy took place in the analyzed data which characterize different samplings.

In other words, when carrying out the comparative analysis of banks functioning and development the analyzed microsituations distinguishability is studied. Then the value of the Wilcoxon test can be used as the measure of distinction (agreement) of the microsituations being considered. The greater is the value of the test being considered, the more distinguishable as a whole are the microsituations being considered and vice versa, the less is the value of the test being considered the closer are the microsituations being considered.

The Initial Data and Results of the Comparative Analysis of Banks Functioning in Ukraine

The foregoing approach is being considered as an example of the banking in Ukraine in terms of such index as a share of the granted credits in the overall totality of bank assets. The paramount importance of consideration of such banking values is associated with that just the credits

- on the one hand, constitute a considerable part of bank operations and, respectively, operating profits in total gains of a bank from such operations,
- on the other hand, the granted credits growth results in credit risks and, consequently, in the banks development destabilization.

Thus the problem associated with the succession of the development dynamics of relation between the granted credits and total volume of banks assets both for the banking system as a whole, and in terms of separate banks functioning is rather significant. The more so the generalized dynamics of the relation between the granted credits and total volume of banks assets as a whole is indicative of the rise in the banks preferred weight with the increased part of the granted credits in their assets volume (Fig.2, generalized using the site www.finance.ua).

Hence the essence of the first question as to carrying out the comparative analysis of the banking activity consists in estimation of the succession in variation of the granted credits preferred weight in their assets volume during each year of the period being investigated. To analyze such a succession is possible on the basis of investigation of the microsituations each of them describes the state of the banking system functioning as a whole for the fixed date of time t_p in terms of the banking activity index x_{tp}^{γ} – the credits preferred weight in the banks assets (see Eq. 8). The results of such investigation obtained within the periods of 2004, 2005, 2006 and 2007 years in section of each month represent a separate microsituation shown in Figs.3-6 (generalized on the basis of the above approach and data of the site www.finance.ua). In this case the black circles mark the microsituations the most consistent between themselves, the microsituations less consistent are not shown at all.

As can be seen from the data in Fig. 3 the corresponding consistency between microsituations in the banking system development of Ukraine by the results of 2004 in terms of the credits specific weight in the banks assets is the least one.

At the same time the analysis of data from Fig.3-Fig.6 testifies that year after year the consistency between the microsituations becomes stronger. This is apparent both from the increase in the microsituations number and from the increase in such consistency, the decrease in the circles dimensions demonstrates this. Hence a dangerous situation forms in the banking system development in Ukraine as a whole, which is marked by the rise in the credits specific weight in the banks assets structure, this can cause the rise in credits risk level. Moreover, continuity in such development is observed.

In other hand the analysis performed according to the above methods of consistency in development of separate banks is not less interesting in the considered aspect. To perform such an analysis let us consider a group of 12 banks representing those representing and operating in the same region that makes it possible to consider indirectly the action of various factors on their functioning and development. For the microsituations, their comparison will represent consistency of separate banks development, generalization of their finance flows appears, this represents the specific weight of credits in the structure of such banks assets. Further comparison is carried out on the basis of Wilcoxon criterion according to Eq.5. Fig.7 shows results of such consistency.

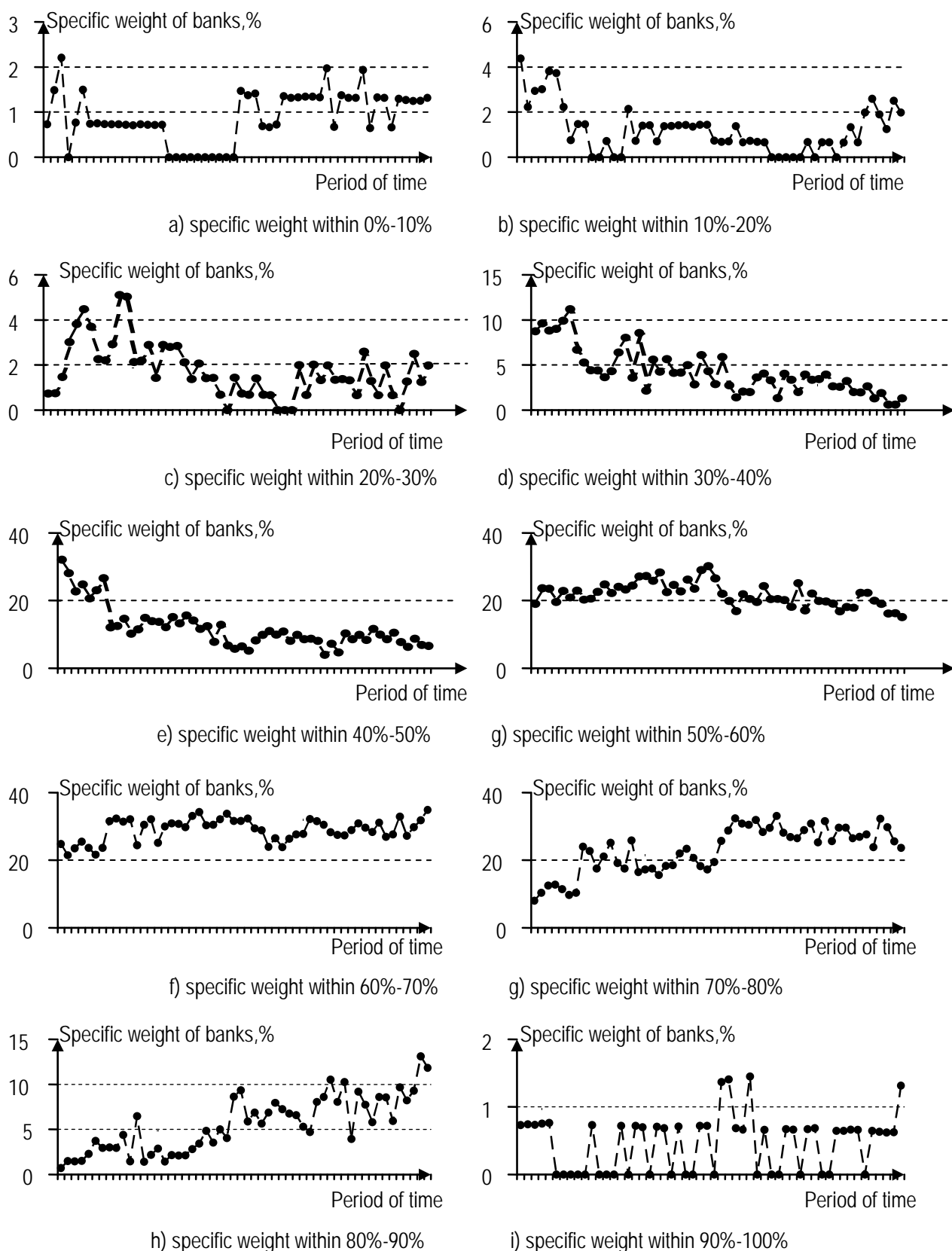


Fig.2 The specific weight dynamics of the granted credits to the total assets volume in the banking system as a whole during the period from 01.01.2004 till 01.05.2008

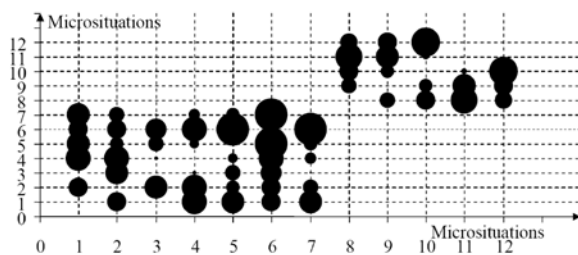


Fig.3 Consistency of microsituations representing variation of the credits specific weight in the banks assets volume according to the results of the banking system work in 2004.

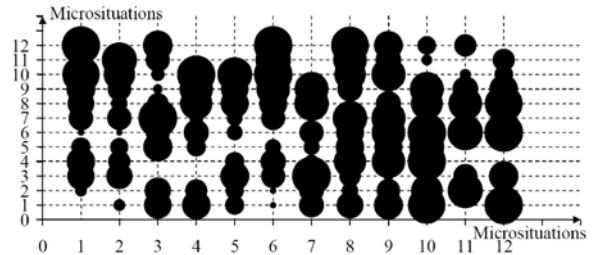


Fig.4 Consistency of microsituations representing variation of the credits specific weight in the banks assets volume according to the results of the banking system work in 2005.

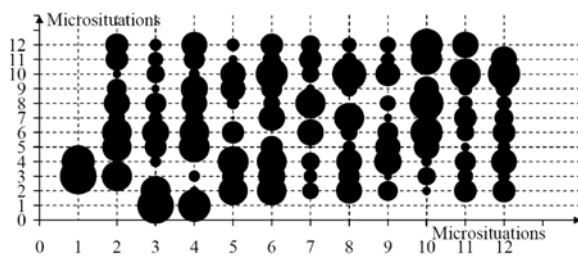


Fig.5 Consistency of microsituations representing variation of the credits specific weight in the banks assets volume according to the results of the banking system work in 2006.

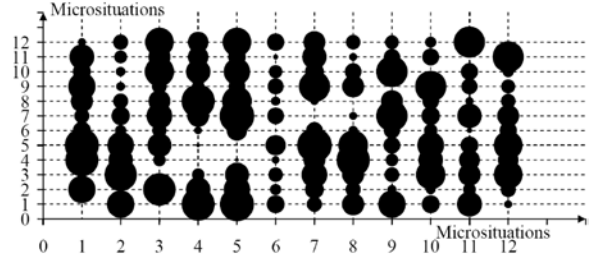


Fig.6 Consistency of microsituations representing variation of the credits specific weight in the banks assets volume according to the results of the banking system work in 2007.

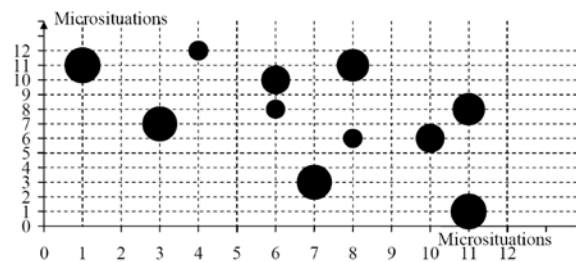


Fig.7 Consistency of microsituations representing variation of the credits specific weight in the banks assets volume according to separate banks of the group being studied by the results of their work within the period from 01.01.2004 till 01.05.2008

As evident from the data in Fig.7 the microsituations consistency in the considered aspect is not observed for the group of banks under study. Thus it may be concluded that each of the banks chooses its own strategy of increase of the credits being granted. Nevertheless, according to the data from Figs.3-6 such a strategy as a whole is aimed at increasing the credits specific weight in the banks assets structure. Consequently, the problem of the credit risk rise remains an urgent one.

If we speak about direct visualisation of results of the comparative analysis of functioning of banks it is possible to take advantage also of their representation in the form of some curve that has been described in subsection visualisation of processes of the comparative analysis of functioning of banks of the given work. In particular useful is the consideration of difference between the received coordination of considered microsituations according to fig. 3 – fig. 6 where such distance is considered for various metrics from the point of view of change of degree of the coordination considered microsituations (in this case the size of each circle reflects conformity

degree (a coordination) of investigated microsituations). In the formalized expression it can be written down as follows:

$$R = M(S_n^m, S_{n+1}^{m+1}) . \quad (9)$$

where R is distance of coincidence between investigated microsituations in the framework of some metrics M ; S_n^m is dimension of certain circle characterising microsituation nm in the former period of analysis of the whole set of microsituations, describing the actions of investigated economic subjects; S_{n+1}^{m+1} is size of a certain circle characterising a microsituation during the subsequent period of the analysis of a cumulative set of microsituations, describing investigated subjects of managing activities.

The received curve by means of the formula 9 will reflect dynamics of change of a coordination of microsituations of the investigated subject of managing and may be used as some profile of its dynamics. On the other hand you may use methods of nonlinear dynamics for consideration of such dynamics which also find wide application for carrying out of the comparative analysis of functioning of objects in economic researches.

Methods of Nonlinear Dynamics are in the Estimation of Development of Banks

Methods of nonlinear dynamics are widely used in analysis and forecasting of parameters showing the development of stock exchange market, insurance market, dynamics of investment handling. Simultaneously analyses of bank segments of finance market based on methods of nonlinear dynamics are not sufficient explored in scientific publications. One of boundaries of such approach to such type of markets is the necessary amount of sample data collected, which may characterise the development of bank sector. Even for such markets the investigation of discontinuities of economic processes is quite important for taking into account existing dynamics and the possibility of weakening regarding to further development of banks.

Phase portrait of statistical data series is the key term of nonlinear dynamics, characterising main parameters of bank's processing and their time induced changes. Such series are e.g.

KI – data series, defining dynamics of bank's loan-investment portfolio;

KR – data series defining dynamics of loans handed over;

MK – data series, defining bank's activity on the markets of interbank loans;

ZP – data series, characterising dynamics of amount of bank's securities;

D – data series, defining the general amount of resources, acquired as deposits;

DF – data series, generalizing amount of resources acquired as deposits of physical persons;

DY – time series generalizing amount of resources acquired as deposits of legal persons;

In this way, bank's activity may be described as an amount of data series mentioned above, which can be generalised as follows:

Data series, defining dynamics of bank's loan-investment portfolio as:

$$KI(x_1, x_2, \dots, x_t) = KR(y_1, y_2, \dots, y_t) + MK(z_1, z_2, \dots, z_t) + ZP(d_1, d_2, \dots, d_t) , \quad (10)$$

and data series, defining the overall amount of resources, acquired as deposits:

$$D(e_1, e_2, \dots, e_t) = DF(ef_1, ef_2, \dots, ef_t) + DY(ey_1, ey_2, \dots, ey_t) , \quad (11)$$

Where $x_t, y_t, z_t, d_t, e_t, ef_t, ey_t$ are values of according series at a fixed time moment t .

Then, in a phase space of dimension 2 using cartesian coordinates the phase portrait of statistical data series may be defined as a set of points:

$$\Phi(CHR) = \{(r_i, r_{i+1})\}, i = \overline{1, t-1}, \quad (12)$$

where CHR – one of series shown above according to equ. 10 and 11.

r_i, r_{i+1} – are the values of series shown, in defined time intervals.

According to the fundamentals of rating of bank's development with methods of nonlinear dynamics in pic. 8 are shown phase portraits of data series, reflecting dynamics of interbank loans, taking into account the specifics of activities of different Ukrainian banks (values are taken from www.finance.ua).

As seen from fig. 8 generally for banks are characteristic different phase portraits of investigated data series. Simultaneously you may see that the dynamics of phase portraits of Basis and Grant banks are most correlated compared with the dynamics of phase portraits of investigated series for Big Energy and Nadra. This fact may be first of all explained by existing bank's strategy to act on market of interbank loans.

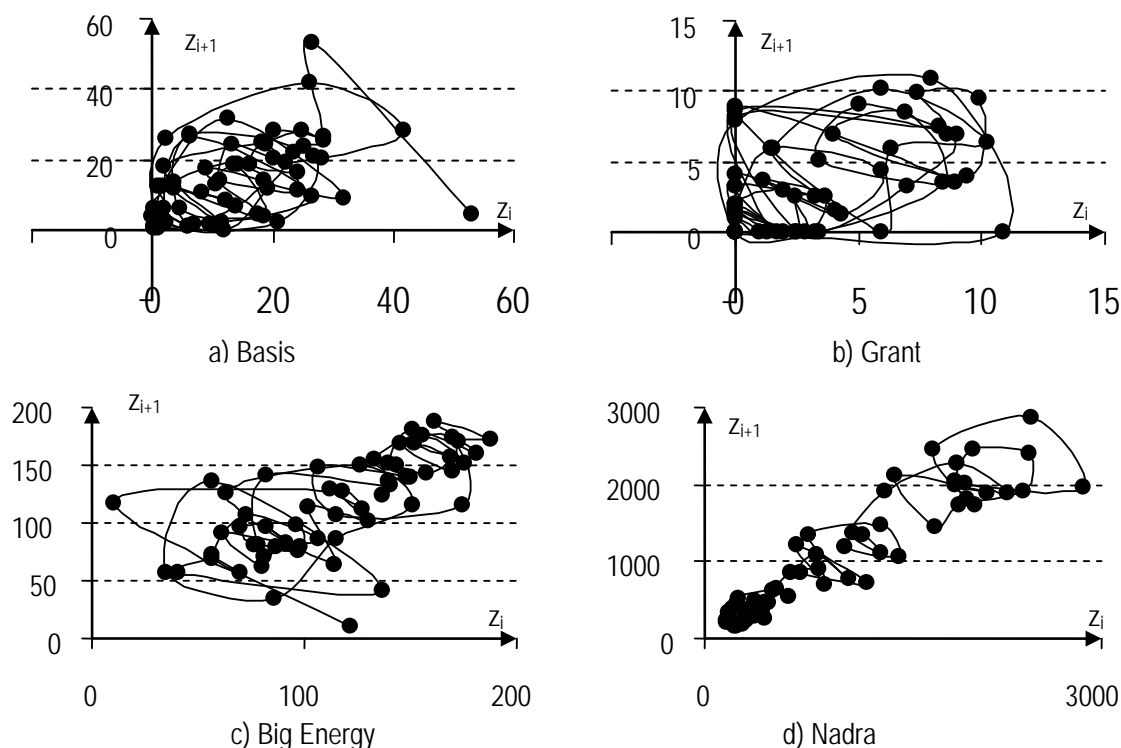


Fig. 8. Phase portraits of data series defining bank's activities on the market of interbank loans for 2004–2008 (monthly)

Such strategy, however, will be defined based on existing conditions and factors, influencing bank's activities. This fact is reflected in the phase portraits of investigated data series shown above. So the Basis bank and the Grant bank are related to the same group of banks, which are additional administrated by intermediate management. Big Energy and Nadra banks also are administrated by intermediate management (www.bank.gov.ua). Therefore, one may state, that phase portraits of data series, shown above, reflect existing conditions of functioning, belonging to different banks. With other words, methods of nonlinear dynamics may be used on equal rights for investigation and analysis of development processes of complex economic systems, banks belonging to.

Conclusions

Thus, in the given work a number of methods developed by authors throughout last four years for carrying out of the comparative analysis of functioning of banks which it is represented perspective is considered. In particular such methods are:

- Formalisation of management efficiency by bank from the point of view of interrelation of its liquidity and profitability on the basis of the theory of indistinct sets;
- Representation of space-time dynamics of the investigated phenomena and processes as conjoint-structured ones;
- Technique of carrying out of the comparative analysis of functioning and development both bank system as a whole, and separate banks in particular, on the basis of the generalised concept of a set of microsituations, each of which characterises such activity proceeding from certain financial streams which in turn reflect those or other indicators of activity of separate banks. Thus for comparison of microsituations nonparametric tests on the basis of Wilcoxon's criterion are used;
- Carrying out of the comparative analysis of functioning of banks by nonlinear dynamics which is defined by concept of a phase portrait, the presence of cyclic changes in time series of the data from the point of view of different indicators of activity of banks is shown, that allows to note features of development of the investigated time series of data.

In the given aspect work presents a general concept of carrying out of comparative analysis of functioning of banks where prevailing focus is given the concept of microsituation, and also the use of approaches of nonlinear dynamics.

Adequacy and working capacity of the offered approaches is approved on the real data, concerning various directions of bank activity. It allows applying the offered approaches to carrying out of the expanded comparative analysis of various directions of activity, both separate banks, and bank systems.

Bibliography

1. Kuzemin A., Lyashenko V. Fuzzy set theory approach as the basis of analysis of financial flows in the economical security system // International Journal "INFORMATION THEORIES & APPLICATIONS" – 2006. – Vol. 13, Num. 1. – P. 45–51.
2. Kuzemin A., Lyashenko V. Analysis of Spatial-temporal Dynamics in the System of Economic Security of Different Subjects of Economic Management // International Journal "Information Technologies and Knowledge" – 2008. – Vol. 2, Num. 3. – P. 234–238.
3. Kuzmenko E.S. Methods for banks safety rating estimation// Urgent problems of economics. – 2007. – № 1. – P. 164-175.
4. Indicators of financial stability. Instruction manual. – Washington, Columbia District, USA: International Monetary Fund, 2007. – 326 p.
5. Azarenkova G.M. Models and methods for financial flows analysis. – KharkovX: БКФ "Grif", 2005. – 119 p.
6. Watsheh T.G., Parramaw K. Quantitative methods in finances. – M.: Finances, ЮНИТИ, 1999. – 527 p.
7. Nedosekin A.O. Application of fuzzy models to banks finances management // <http://sedok.narod.ru>.
8. E.V. Comparative analysis of the banking systems of Russia and Czech Republic under condition of the transitional period in economics // Materials of the V International scientific-practical conference "Countries with transitional economy under conditions of globalization". – M.: РУДН, 2006.
9. Golovan' S.V., Karminsky A.M., Kopylov A.V., Peresetsky A.A. Models of the Russian banks default probability. Preliminary banks partitioning into clusters // Preprint # 2003 XXX. – M.: РЭШ, 2003. – 49 p.

10. Snityuk V. Evolutionary clusterization of complex objects and processes // XI-th International Conference «Knowledge-Dialogue-Solution» – Varna, 2005. – Vol. 1. – P. 232–237.
11. Kuzemin A., Lyashenko V., Bulavina E., Torojev A. Analysis of movement of financial flows of economical agents as the basis for designing the system of economical security (general conception) // Third international conference «Information research, applications, and education». 27-30 June. Varna, Bulgaria. – Sofia: FOI-COMMERCE – 2005. – P. 204–209.
12. Kuzemin A., Lyashenko V. Conceptual Foundations of Construction of the Models and Procedures for Prediction of the Avalanche-dangerous Situations Initiation // International Journal INFORMATION THEORIES & APPLICATIONS. – 2008. – Volume 15. – №2. – P. 153–158.
13. General theory of statistics: Manual// T.V. Ryabushkin, M.R. Efimova, I.M. Ipatova, N.I. Yakovleva. — M.: Finance and statistics, 1981.

Authors' Information

Kuzemin A.Ya. –*Doctor of Technical Sciences, Prof. of Information Department, Kharkov National University of Radio Electronics, Head of IMD, (Ukraine), kuzy@kture.kharkov.ua*

Lyashenko V.V. – *senior scientific employee, Kharkov National University of Radio Electron (Ukraine), kuzy@kture.kharkov.*

INDEX OF AUTHORS OF IJ ITA, VOLUME 16

Aleksandr Tugaenko	193	Juan Castellanos	203
Alexander Kuzemin	384	Jürgen Bast	356
Ana Martinez	238	Liudmila Cheremisinova	145
Andrey Tikhonin	356	Luis Fernández	222
Andri Mirzal	340	Luis Fernando de Mingo	233
Andrzej Smykla	74	Michael Tyutyunnik	323
Angel Castellanos	238	Miguel A. Peña	233
Angel Goni Moreno	203	Natalia Kudashova	356
Arcadio Sotto	238	Nuria Gómez Blas	233
Arkadij Zakrevskij	25	Oleksandr Khodzinskyi	349
Dimiter Skordev	90	Oleksiy Chugayenko	370
Dimitrina Polimirova	114	Olga Siedlecka	53
Dmitri A. Rachkovskij	176, 269	Paula Cordero	203
Dmitry Cheremisinov	145	Serge V. Slipchenko	269
Dmitry Kinoshenko	260	Sergii Sirenko	303
Elena G. Revunova	176	Sergiy Kryvyy	349, 370
Elena Ilina	65	Tatyana Romanenko	193
Elena Yegorova	260	Tatyana Stupina	43
Eugene Nickolov	114	Valeriy Kamaev	356
Francisco Gisbert	233	Valeriya Gribova	103
Francisco Javier Gil	222	Varun Gupta	136
Igor Gorban	5	Velin Krlev	291
Igor Karelin	161	Vitaly Vishnevsky	193
Irena Nowotyńska	74	Vladimir Kalmykov	193
Irene Artemieva	323	Vladimir Mashtalir	260
Irina Polyakova	356	Vyacheslav Lyashenko	384
Jitender Kumar Chhabra	136	Yevgen Viktorov	245
Jorge Tejedor	222	Yevgeniy Bodyanskiy	245

TABLE OF CONTENTS OF IJ ITA, VOLUME 16

Number 1

Preface	3
Cognition Horizon and the Theory of Hyper-Random Phenomena <i>Igor Gorban</i>	5
Solving Large Systems of Boolean Equations <i>Arkadij Zakrevskij</i>	25
Adaptive Approach to Static Correction with Respect to Aprioristic Information in Problem of Seismic Data Processing <i>Tatyana Stupina</i>	43
Method and Algorithm for Minimization of Probabilistic Automata <i>Olga Siedlecka</i>	53
Application of Information Theories to Safety of Nuclear Power Plants <i>Elena Ilina</i>	65
Increasing Reliability and Improving the Process of Accumulator Charging Based on the Development of PCGRAPH Application <i>Irena Nowotyńska, Andrzej Smykla</i>	74
Transliteration and Longest Match Strategy <i>Dimiter Skordev</i>	90

Number 2

Methods for Automated Design and Maintenance of User Interfaces <i>Valeriya Gribova</i>	103
Investigation on Compression Methods Used as a Protection Instrument of File Objects <i>Dimitrina Polimirova, Eugene Nickolov</i>	114
Object Level Run-Time Cohesion Measurement <i>Varun Gupta, Jitender Kumar Chhabra</i>	136
Programming of Agent-Based Systems <i>Dmitry Cheremisinov, Liudmila Cheremisinova</i>	145
Some Approaches for Software Systems Analyses and its Upgrade Prediction <i>Igor Karelin</i>	161
Using Randomized Algorithms for Solving Discrete Ill-posed Problems <i>Elena G. Revunova, Dmitri A. Rachkovskij</i>	176
Kirlian Image Preprocessing Diagnostic System <i>Vitaly Vishnevskiy, Vladimir Kalmykov, Tatyana Romanenko, Aleksandr Tugaenko</i>	193

Number 3

Gene Codification for Novel DNA Computing Procedures <i>Angel Goni Moreno, Paula Cordero, Juan Castellanos</i>	203
Fast Linear Algorithm for Active Rules Application in Transition P Systems <i>Francisco Javier Gil, Jorge Tejedor, Luis Fernández</i>	222
Extended Networks of Evolutionary Processors <i>Luis Fernando de Mingo, Nuria Gómez Blas, Francisco Gisbert, Miguel A. Peña</i>	233
Trained Neural Network Characterizing Variables for Predicting Organic Retention by Nanofiltration Membranes <i>Arcadio Sotto, Ana Martinez, Angel Castellanos</i>	238
The Cascade Neo-Fuzzy Architecture Using Cubic-Spikine Activation Functions <i>Yevgeniy Bodyanskiy, Yevgen Viktorov</i>	245
Distance Matrix Approach to Content Image Retrieval <i>Dmitry Kinoshenko, Vladimir Mashtalir, Elena Yegorova</i>	260
Analogical mapping using similarity of binary distributed representations <i>Serge V. Slipchenko, Dmitri A. Rachkovskij</i>	269
A Genetic and Memetic Algorithm for Solving the University Course Timetable Problem <i>Velin Krlev</i>	291

Number 4

Classification of Heuristic Methods in Combinatorial Optimization <i>Sergii Sirenko</i>	303
Parallelization Methods of Logical Inference for Confluent Rule-based System <i>Irene Artemieva, Michael Tyutyunnik</i>	323
Weblog Clustering in Multilinear Algebra Perspective <i>Andri Mirzal</i>	340
Presentation of Ontologies and Operations on Ontologies in Finite-State Machines Theory <i>Sergii Kryvyi, Oleksandr Khodzinskiy</i>	349
Cognitive Approach in Castings' Quality Control <i>Irina Polyakova, Jürgen Bast, Valeriy Kamaev, Natalia Kudashova, Andrey Tikhonin</i>	356
Extended Algorithm for Translation of MSC-diagrams into Petri Nets <i>Sergiy Kryvyy, Oleksiy Chugayenko</i>	370
Methods of Comparative Analysis of Banks Functioning: Classic and New Approaches <i>Alexander Kuzemin, Vyacheslav Lyashenko</i>	384

TABLE OF CONTENTS OF IJ ITA, VOLUME 16, NUMBER 4

Classification of Heuristic Methods in Combinatorial Optimization	
<i>Sergii Sirenko</i>	303
Parallelization Methods of Logical Inference for Confluent Rule-based System	
<i>Irene Artemieva, Michael Tyutyunnik</i>	323
Weblog Clustering in Multilinear Algebra Perspective	
<i>Andri Mirzal</i>	340
Presentation of Ontologies and Operations on Ontologies in Finite-State Machines Theory	
<i>Sergii Kryvyi, Oleksandr Khodzinskyi</i>	349
Cognitive Approach in Castings' Quality Control	
<i>Irina Polyakova, Jürgen Bast, Valeriy Kamaev, Natalia Kudashova, Andrey Tikhonin</i>	356
Extended Algorithm for Translation of MSC-diagrams into Petri Nets	
<i>Sergiy Kryvyy, Oleksiy Chugayenko</i>	370
Methods of Comparative Analysis of Banks Functioning: Classic and New Approaches	
<i>Alexander Kuzemin, Vyacheslav Lyashenko</i>	384
Index of Authors of IJ ITA, Volume 16	397
Table of Contents of IJ ITA, Volume 16	397
Table of Contents of IJ ITA, Volume 16, Number 4	400