ITHEA

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 Gaindric	(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 [Hulianytskyi and Sergienko, 2007].

Definition 1. Consider a set $Y = \{1,...,m\}$, an at most countable set Z called *base space*, and a homomorphism $\varphi : Y \to Z$ 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

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

where X is a finite (or possible countably infinite) set of combinatorial objects, $D \subseteq X$ is a subspace of the feasible solutions, and $f : X \to \mathbb{R}^1$ 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 $g : S \to \mathbb{R}^1$ 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 $N : S \to 2^S$ 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* $G_N = \left(S, V_N\right)$, $V_N = \left\{(s,s') \mid s' \in N(s)\right\}$.

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

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

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 [Kirkpatrik 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	*Local Search* [Sergienko, 1964; Papadimitriou and Steiglitz, 1982] …
Stochastic algorithms	*Simulated annealing* [Kirkpatrik 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 that 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)	*Greedy Algorithms* [Khuller et al., 2007], …
Hybrid algorithms (hybrid heuristics)	Sequential execution of heuristics [Zhuravlev, 1977; Bertsekas et al., 1997], …
Metaheuristics	*Ant Colony Optimization* [Dorigo et al., 1991; Dorigo and Stützle, 2004], *Memetic Algorithms* [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 an 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 perturbated 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

$$
\begin{gathered}
S = \bigcup_{i=1,\dots,m} X_i, \\
X_i = \{\varphi^i : Y_i \to Z, Y_i \subset Y, |Y_i| = i\}, i = 1,\dots,m-1, \\
X_m = X,
\end{gathered}
\tag{2}
$$

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

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 [Hulianytskyi, 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 Seach, 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		*Nearest Neighbour Heuristic* [Hoos and Stützle, 2005] …
Iterative (perturbative) algorithms	**Single-point algorithms**	*Simulated Annealing* [Kirkpatrik et al., 1983; Aarts et al., 2007], *G-algorithm* [Hulianytskyi, 1991] …
	Population-based algorithms	*Genetic Algorithm* [Holland, 1975; Fogel, 2006], *Discrete Particle Swarm Optimization* [Clerc, 2006], *H-method* [Hulianytskyi 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 Hulianytskyi, 1981], Cross-Entropy method [Rubinstein, 1999; Rubinstein and Kroese, 2004], and *H*-method [Hulianytskyi 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	*G-algorithm* [Hulianytskyi, 1991] …
Algorithms with memory	*Tabu Search* [Glover, 1986; Glover and Laguna, 1997] …

Table 8. Memory utilization / functions

Type		Examples
Storing		Keeping best-so-far solution
Guiding	**Short term**	*Tabu list in Tabu Search* [Glover, 1986; Glover and Laguna, 1997] …
	Long term	*Penalties in Guided Local Search* [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003] …
Adaptive		*Pheromone values in Ant Colony Optimization* [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 form 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 is these algorithms. To provide a formal basis for districting 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 $STEP : 2^S \to 2^S$ 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

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

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 $\{N_1, ..., N_L\}$ 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 $I \geq 2$

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

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: $\exists N^I \in \{N_1, ..., N_L\} : x_I \in N^I(x_{I-1})$.

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

The trajectory-continuity of a heuristic with respect to the set of neighbourhood structures $\{N_1, ..., N_L\}$ means that for every generated by the algorithm candidate solution $x \in X_I$ there exists a path in an aggregated neighbourhood graph $G_{N_1,...,N_L} = (S, \bigcup_{j \in \{1,...,L\}} V_{N_j})$ 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 neigbourhood 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 Kernigan, 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 Hulianytskyi, 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	*Tabu Search* [Glover, 1986; Glover and Laguna, 1997], *Frontal Optimization Algorithms* [Sergienko and Hulianytskyi, 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	*G-algorithm* [Hulianytskyi, 1991] …
Algorithms that modify search landscape	*Tabu Search* [Glover, 1986; Glover and Laguna, 1997], *Guided Local Search* [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003], …

Table 11. Types of search landscape modification

Type	Examples
Search space modification	*Tabu Search* [Glover, 1986; Glover and Laguna, 1997]
Evaluation function change	*Guided Local Search* [Voudouris and Tsang, 1995; Voudouris and Tsang, 2003], …
Neighborhood change	*Search in Pulsating Neighborhoods* [Hulianytskyi and Khodzinskyi, 1979], *Variable Neighborhood Search* [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

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

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	*Iterated Local Search* [Lourenço et al., 2002] …
Algorithms with adaptation/learning	*Reactive Tabu Search* [Battiti and Tecchiolli,1994], Adaptive Memetic Algorithms [Ong et. al., 2006], …

Table 13. Types of adaptation

Type	Examples
Parameters tuning	*Adjusting tabu tenure in Reactive Tabu Search* [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	*Simulated annealing* [Kirkpatrik et al., 1983; Aarts et al., 2007], *Genetic Algorithm* [Holland, 1975; Fogel, 2006] …
Model-based algorithms	*Ant Colony Optimization* [Dorigo and Stützle, 2004], *Cross Entropy Method* [Rubinstein and Kroese, 2004], *Estimation of Distribution Algorithms* [Larrañaga and Lozano, 2002] …

Introduced in the field of evolutionary computations the Estimation of Distribution Algorithms [Műhlenbein 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

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

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	*ε-approximate algorithms, Local Search with known lower bound* [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	*Simulated Annealing* [Hajek, 1988], *Genetic Algorithm with elitism* [Rudolph, 1994] $ACO_{\tau_{\min}}$ [Dorigo and Blum, 2005], …
Do not convergent algorithms	*Standard Genetic Algorithm* [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.

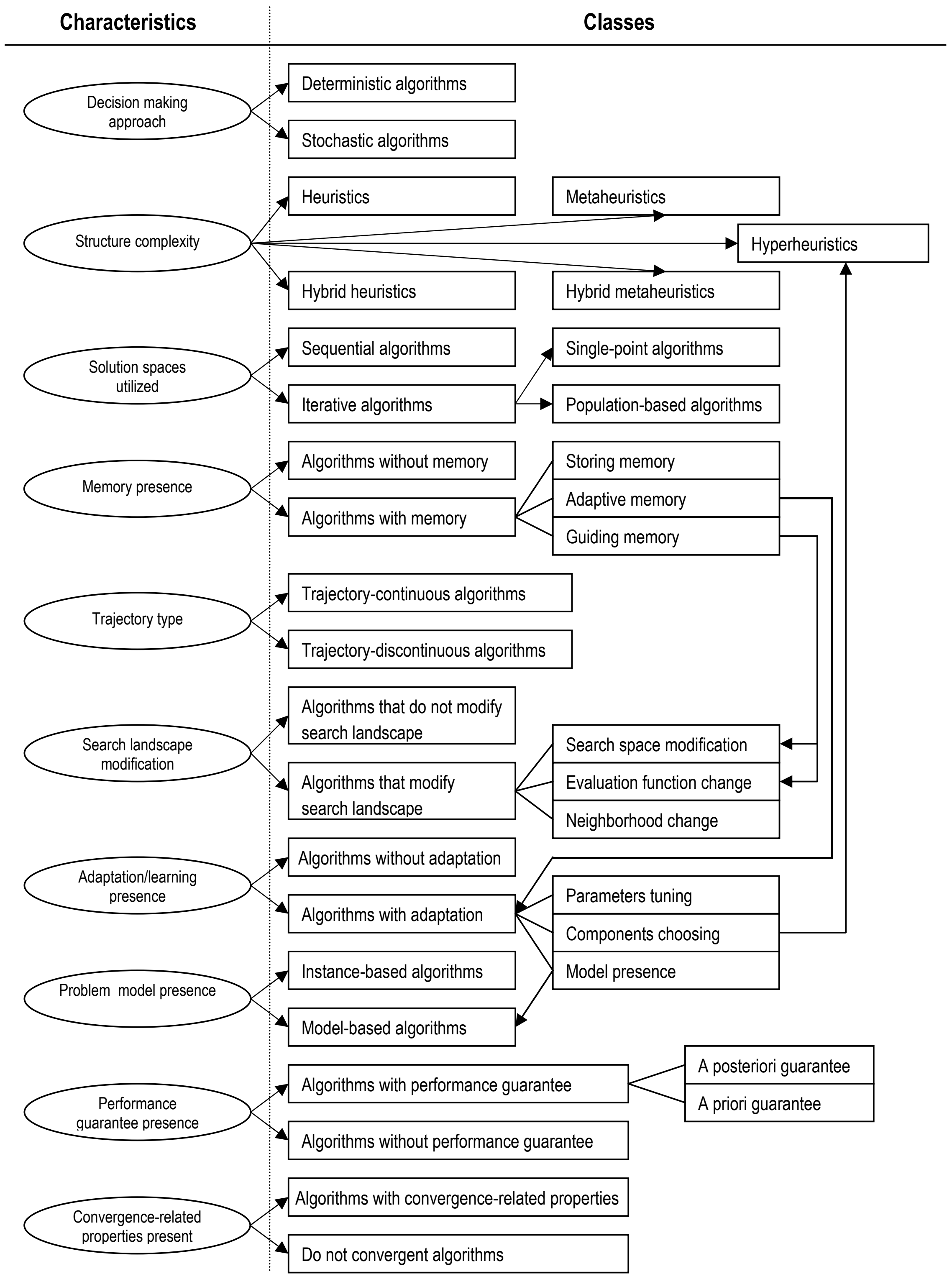
Characteristics
Classes
Decision making approach
Deterministic algorithms
Stochastic algorithms
Structure complexity
Heuristics
Metaheuristics
Hyperheuristics
Hybrid heuristics
Hybrid metaheuristics
Solution spaces utilized
Sequential algorithms
Iterative algorithms
Single-point algorithms
Population-based algorithms
Memory presence
Algorithms without memory
Algorithms with memory
Storing memory
Adaptive memory
Guiding memory
Trajectory type
Trajectory-continuous algorithms
Trajectory-discontinuous algorithms
Search landscape modification
Algorithms that do not modify search landscape
Algorithms that modify search landscape
Search space modification
Evaluation function change
Neighborhood change
Adaptation/learning presence
Algorithms without adaptation
Algorithms with adaptation
Parameters tuning
Components choosing
Model presence
Problem model presence
Instance-based algorithms
Model-based algorithms
Performance guarantee presence
Algorithms with performance guarantee
Algorithms without performance guarantee
A posteriori guarantee
A priori guarantee
Convergence-related properties present
Algorithms with convergence-related properties
Do not convergent algorithms

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 Tecchiolli, 1994] R. Battiti, G. Tecchiolli. 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/~gxk/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. Colorni. Positive feedback as a search strategy. Tech. Rep. 91-016. Milan, Italy: Politecnico di Milano, Dipartimento di Elettronica, 1991. 22 p. ftp://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, Entreprise 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.

[Kirkpatrik et al., 1983] S. Kirkpatrik, 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.

[Műhlenbein and Paaß, 1996] H. Műhlenbein, 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/~eozcan/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/~ierrr01/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 Hulianytskyi, 1981] I.V. Sergienko, L.F. Hulianytskyi. Frontal optimization algorithms for multiprocessor computers. In: Cybernetics, 1981, Vol. 17, No. 6. pp. 719–721.

[Sergienko at al., 1989] I.V. Sergienko, L.F. Hulianytskyi, 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 Universitat 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 p. 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