



13th International Symposium “Intelligent Systems” (INTELS’18)

Hierarchical Model of Parallel Metaheuristic Optimization Algorithms

E.Y. Seliverstov*, A.P. Karpenko

Bauman Moscow State University, ul. Baumanskaya 2-ya, 5, Moscow 105005, Russia

Abstract

The paper introduces a novel model of parallel metaheuristic optimization algorithms. The hierarchical graph model of a parallel optimization algorithm is proposed. It consists of the model for a parallel optimization algorithm at the top level of the hierarchy and the model for a sequential optimization algorithm at the bottom level. The unified representation of a metaheuristic optimization algorithm, which allows representing a class of metaheuristic algorithms, is used. The extension of the proposed model to the parametric hierarchical model is proposed. Graph model transformations for a parallel algorithm analysis and synthesis are introduced. The representation of several metaheuristic algorithms with the proposed model is discussed.

© 2019 The Authors. Published by Elsevier B.V.

This is an open access article under the CC BY-NC-ND license (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Peer-review under responsibility of the scientific committee of the 13th International Symposium “Intelligent Systems” (INTELS’18).

Keywords:

evolutionary algorithms; metaheuristic; optimization; parametric optimization; parallel algorithms; particle swarm optimization.

1. Introduction

Metaheuristic optimization algorithms are a relatively new and fast developing area of algorithms which combine heuristics for solving a broad class of optimization problems without a priori knowledge about a problem origin. Metaheuristic algorithms are divided mainly into evolutionary algorithms and swarm intelligence algorithms[1].

During the extensive research in evolutionary and swarm intelligence computation a lot of hybrid algorithms were proposed, which combine major features of existing optimization algorithms. A great variety of metaheuristic algorithms on the one hand and their common features (stochastic behaviour, population structure) on the other hand had led to the development of unified models. Such models were researched by De Jong [2], Back [3]. De Jong had developed the unified approach to represent an evolutionary algorithm with basic unified elements such as common evolutionary operators, methods for representing individuals (genotypical or phenotypical), methods for choosing

* Corresponding author. Tel.: +7-916-200-4026.

E-mail address: evgeny.seliverstov@omniverse.ru

population size, methods for fitness function evaluation. Several mathematical models for metaheuristic algorithm had been analyzed: a population dynamics model, a Markov model for fitness function distribution.

Due to a high computational complexity, parallel optimization algorithms became widespread in the last years. The most used models of parallel optimization algorithms [4, 5, 6] are global (known as a master-slave); coarse-grained (island); fine-grained (cell) models. These models represent the parallelism by scheduling parallel tasks on so-called *computational agents*.

The research in parallel task scheduling, parameter control, self-adaptive algorithms motivates the need for an abstract model of a parallel metaheuristic optimization algorithm. Existing models clearly distinguish models of sequential algorithm and models of parallel algorithms so a parallel algorithm cannot be modelled uniformly. Modern models of the parallel metaheuristic optimization algorithm are required to satisfy this requirement. In this paper, we introduce the single hierarchical model, which combines a model of sequential metaheuristic algorithm of optimization and a model of a parallel algorithm. To provide such a model we need to choose a parallel computation model to be based on.

A parallel computation model presents an abstraction of a parallel system for the parallel algorithm. In this paper, we choose the parallel computation model that well exposes essential properties of the island parallel model. This choice is guided mainly by a level of explicit parallelism exposed in a model and an architecture of a parallel system. The classification by the level of explicit parallelism in a parallel computation model was proposed by Skillicorn [7]. This classification introduced the following levels (by decreasing of model abstractness): explicit parallelism, explicit decomposition, explicit mapping, explicit communication, explicit synchronization. The island parallel model corresponds to the level of explicit decomposition. Well-known models of this level are bulk synchronous parallel (BSP) [8], LogP [9], etc. A BSP model is well suited for a modern class of manycore and massive-parallel systems with non-uniform memory access.

The rest of the paper is organized as follows. In section 2 we propose the hierarchical model of the parallel algorithm that efficiently composes a unified model of a sequential metaheuristic algorithm and an island parallel model concerning a BSP parallel computation model. In section 3 we introduce the parametric hierarchical model. In section 4 we apply a proposed model to the certain metaheuristic optimization algorithms — particle swarm optimization [1], evolutionary strategy [10], evolutionary programming [11].

Some theorems stated as assertions to conserve space.

2. Hierarchical model of parallel metaheuristic optimization algorithms

2.1. Hierarchical model of algorithms

The synthesis of the proposed parallel optimization algorithm is based on the hybridization approach. Well-known approaches are a hybridization with a global or island models on top-level and a diffuse model on bottom-level; cooperation hybridization; coevolution hybridization [12]. We use a high-level (sequential and parallel algorithm are clearly separated) embedding hybridization (by classification of Wang [13]). In this approach, the parallel algorithm is based on the island parallel model [14, 15]. The embedded sequential algorithm is performed at the computational agents on each iteration of the parallel algorithm.

Let A_s denote a sequential metaheuristic algorithm (subscript s hereafter means sequential) and A denote a parallel metaheuristic algorithm.

As the mathematical model of the algorithm, we use a flow graph. The formal hierarchical model GA of the metaheuristic algorithm A consists of two levels of hierarchy: *top level* — the graph model GA_p of the parallel algorithm A ; *bottom level* — the graph model GA_s of the sequential algorithm A_s .

For both levels of graph models, we introduce *operation classes* that represent an algorithm as an aggregate of operation classes. For the bottom-level model, these operation classes correspond with basic unified elements from the De Jong's classification. They allow us to abstract the model from a specific metaheuristic algorithm. For the top-level model, operation classes correspond with phases of the BSP computational model. We represent one iteration of the parallel optimization algorithm as one superstep of the BSP model, which includes phases of computation, communication and synchronization. The process of performing the sequential optimization algorithm on distinct computational agents maps to the computation phase of BSP. One iteration in the top-level model consists of one

season for an island PSO or a genetic algorithm. One iteration in the bottom-level model is a step of a metaheuristic algorithm.

2.2. Graph model of metaheuristic optimization algorithms

We introduce an unweighted directed graph $GA_s = (VA_s, EA_s)$ of the sequential metaheuristic optimization algorithm A_s , where $VA_s = V(GA_s)$ is a set of vertices, which represent algorithm operations; $EA_s = E(GA_s)$ is a set of edges, which represent data dependency between operations. Denote a specific graph vertex S_{GA_s} as a *start vertex*. As the graph vertex depicts an algorithmic operation, it is possible to unambiguously use a vertex designation instead of an operation designation.

The algorithm A_s is an iterative algorithm. The graph GA_s models an algorithm with an iteration data dependency, therefore it contains a directed edge $(v, S_{GA_s}), v \in V(GA_s)$ that represent inter-iteration data dependency. Define this edge as the *iteration edge* e_{GA_s} .

Edges $E(GA_s) \setminus e_{GA_s}$ represent intra-iteration data dependencies (solid edges in Fig. 1(a)).

Definition 1. Let the graph \widetilde{G} be the *out-iterative graph* for a graph G , if

$$V(\widetilde{G}) = V(G); e_G \in E(G); E(\widetilde{G}) = E(G) \setminus e_G.$$

It should be noted that the graph \widetilde{G} may be cyclic as well as acyclic.

The set of all possible algorithms A_s forms the algorithms class \mathcal{MA} . The graph GA_s represents one iteration of an algorithm, so the algorithm class \mathcal{MA} is bound by one-step optimization methods.

Denote the set of operations of optimization algorithms from the class \mathcal{MA} as $\mathcal{E}(\mathcal{MA})$. We introduce a partition $O_s(\mathcal{E}(\mathcal{MA}))$ of a set $\mathcal{E}(\mathcal{MA})$ as a tuple of subsets, called *operation classes*:

$$O_s = \langle SS, SA, SB, SE, ST \rangle, \quad (1)$$

where

- SS — operation of the algorithm start;
- SA — operations of the iteration preparation;
- SB — operations of computations during one iteration;
- SE — operations of computation results aggregation from the iteration;
- ST — operation of the algorithm finalization.

Each operation class from the partition O_s corresponds to one or many unified operations of metaheuristic algorithm from the De Jong's classification. Operation class SS consists of one operation represented by the vertex ss of the graph GA_s , which is a source of directed out-iterative graph $\widetilde{GA_s}$. Class ST is represented by the sink vertex st of graph $\widetilde{GA_s}$. Class SB consists of several operations $b_i, i \in [0 \dots m]$, which are mapped to graph vertices b_i . Classes SA, SE represent iteration preparation and iteration aggregation operations accordingly.

Assertion 1. Given any algorithm $A \in \mathcal{MA}$, the graph structure and operation classes of an algorithm remains the same for all algorithms in \mathcal{MA} , but a number of vertices and edges, which correspond to operations, may vary.

2.3. N-iterative graph

Given the graph $G = (V, E)$ has the iteration edge and the corresponding out-iterative graph \widetilde{G} , denote the N -iterative graph $G^{(N)} = (V^{(N)}, E^{(N)})$ be a graph composed from N subgraphs \widetilde{G} and $N - 1$ iteration edges which connect those subgraphs. Iteration edges of the graph $G^{(N)}$ are bridges in this graph.

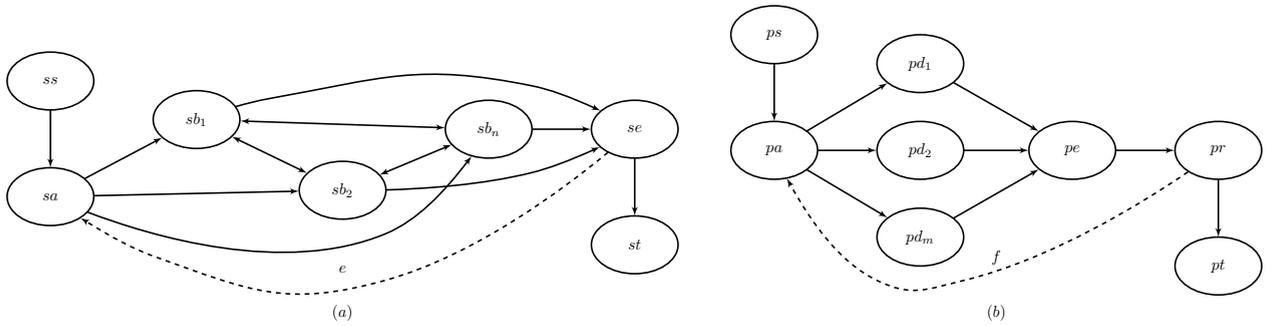


Fig. 1. (a) Graph GA_s of an example algorithm A_s . (b) Graph GA_p of an example algorithm A .

Vertex set of this graph is defined as

$$V^{(N)} = \{v_1^{(1)}, v_2^{(1)}, \dots, v_{|V|}^{(1)}, \dots, v_1^{(N)}, v_2^{(N)}, \dots, v_{|V|}^{(N)}\}.$$

A vertex $v_i^{(k)}$ with a superscript k belongs to the subgraph \tilde{G}_k that corresponds to the algorithm iteration k . Hereafter an expression $|X|$ describes a cardinality of a set X .

Edge set $E^{(N)}$ of the N -iterative graph is defined by the following rules.

Rule 1. If the graph \tilde{G} has the edge (v_i, v_j) and this edge is not an intra-iteration edge, then include to the edge set the subset

$$\left((v_i^{(1)}, v_j^{(1)}), \dots, (v_i^{(N)}, v_j^{(N)}) \right);$$

Rule 2. If the vertex v_i belongs to an operation class SA and the vertex v_j belongs to an operation class SE , then include to the edge set the subset

$$\left((v_j^{(1)}, v_i^{(2)}), (v_j^{(2)}, v_i^{(3)}), \dots, (v_j^{(N-1)}, v_i^{(N)}) \right).$$

Assertion 2. If the graph \tilde{G} is acyclic, then N -iterative graph $G^{(N)}$ is acyclic too.

It should be noted that an out-iterative graph \tilde{G} is the 1-iterative graph too.

An example of 2-iterative graph $GA^{(2)}$ is shown in Fig. 2.

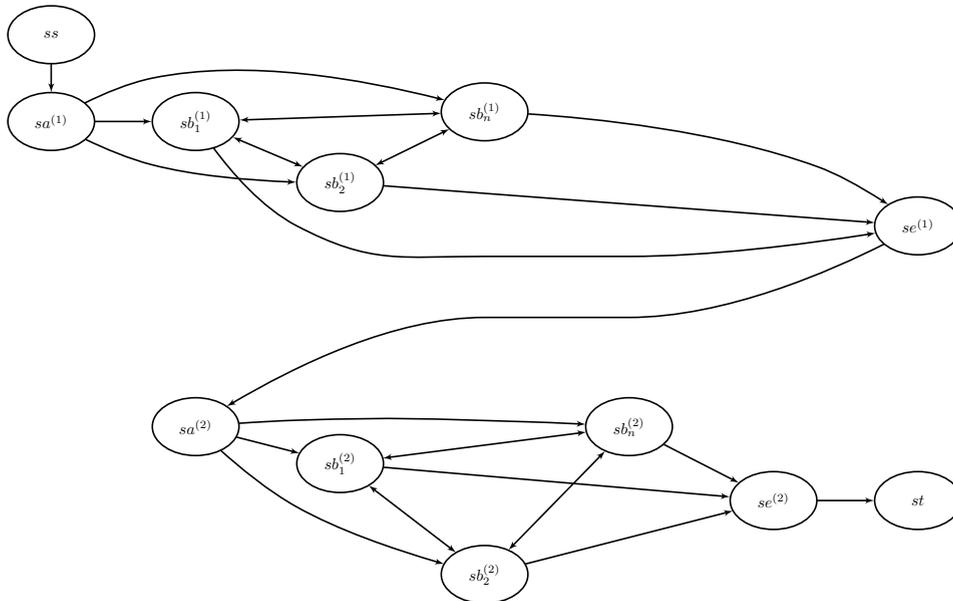
2.4. Graph model of parallel algorithms

We introduce the model GA_p of a parallel algorithm based on a combination of the island parallelism model and the graph model GA_s (see Section 2.2) of the sequential metaheuristic algorithm A_s .

Based on the analogue with an expression (1) we define operation classes in the parallel algorithm;

$$O_p = \langle PS, PA, PD, PE, PR, PT \rangle, \tag{2}$$

where

Fig. 2. Example of 2-iterative graph $GA^{(2)}$.

- PS — operations of agents initialization;
- PA — operations of algorithm state distribution to agents;
- PD — operations of performing the sequential algorithm by agents;
- PE — operations of aggregation of computation results from agents;
- PR — operations of refreshing algorithm's state;
- PT — operations of algorithm finalization.

A parallel graph model GA_p for an example parallel algorithm A is shown in Fig. 1(b).

Operation classes PS , PT correspond to vertices ps , pt of the graph GA_p accordingly. In the out-iterative graph \widetilde{GA}_p they are also sink and source vertices. The operation class PE includes a vertex pe .

As computational agents of the parallel algorithm perform a sequential optimization independently of each other, so each vertex pd_i , $i \in [0 \dots |PD|]$ of the graph GA_p corresponds to subgraphs D_i of the model GA_s of the sequential algorithm, which performed by a computational agent i .

Each path in the graph GA_p from vertices $pa \in PA$ to vertices $pe \in PE$ represents a computational phase of a BSP model performed by each computation agent. The path $pa \dots pr$ represents one BSP superstep.

Definition 2. Given a parallel graph model GA_p and a sequential graph model GA_s , the *graph expansion* is a procedure of replacing vertices $pd_i \in V(GA_p)$ to the subgraphs $D_i \subset GA_s$.

In a general case subgraphs D_i, D_j may correspond to different sequential algorithms.

Definition 3. The *expanded graph* is the graph GA_p^+ , which is obtained by the expansion procedure from a graph GA_p . An original graph GA_p we denote as the *unexpanded graph*.

An example of an expanded graph GA_p^+ for an unexpanded graph GA_p is shown in Fig. 3.

It should be noted that the graph expansion from G to G^+ is not a vertex mapping, because during the graph expansion procedure a vertex set of the graph G^+ is supplemented with a vertex set of each sequential graph model GA_s .

Formally the graph expansion procedure is described as follows:

$$V(G^+) = V(G) \cup \left(\bigcup_i f_V(V(GA_s)_i) \setminus \bigcup_i pd_i \right), \quad (3)$$

$$E(G^+) = f_E(E(G)) \cup \left(\bigcup_i (pa; sa_i) \right) \cup \left(\bigcup_i (se_i; pe) \right) \setminus \left((pa; pd_i) \cup (pd_i; pe) \right), \quad (4)$$

where $i \in [0 \dots |D|]$; f_V, f_E are functions of renaming graph D_i vertices to maintain vertex uniqueness in the graph G^+ . An example of function f_V can be a mapping given by as a set of pairs

$$f_V(V) = \{v_i, v_j\},$$

$$v_j = \begin{cases} v_{ki}, & \text{if } \exists k : v_i \in D_k, \\ v_i & \text{otherwise.} \end{cases}$$

Basing on the analogue with the graph GA_s , the iteration edge f_{GA_p} defines a data dependency between the consecutive iterations of the parallel algorithm. Excluding the iteration edge from the graph extracts one iteration in this graph. That iteration is represented by the out-iterative graph \widetilde{GA}_p .

Assertion 3. Out-iterative graph model \widetilde{GA}_p of a parallel algorithm is acyclic.

It should be noted that the Assertion 3 is inapplicable to the graph model GA_s of a sequential algorithm.

The graph expansion of the algorithm model GA_p , the out-iterative graph GA_{+p} transforms them into the cyclic graphs. From the perspective of a graph model classification, the out-iterative graph GA_p^+ or the N -iterative graph is an acyclic task dependency graph, while the original graph GA_p is a flow graph.

Assertion 4. Any subgraph $D_i, i \in [0 \dots |D|]$ of the graph GA_p is an acyclic task dependency graph of the sequential algorithm A_s .

In other words, we require from the graph model GA_p that any subgraph D_i of the graph model GA_p must be the graph model of the sequential algorithm A .

Thus the graph model of a parallel metaheuristic optimization algorithm A consists of two hierarchy levels:

- top-level model (the graph GA_p , which represents an island parallel model);
- bottom-level model (subgraphs D_i of the graph GA_p , which represent a sequential algorithm $A_s \in \mathcal{MA}$).

Graphs of those two hierarchy levels are correlated with a graph expansion procedure (3).

We should emphasize, that for different metaheuristic optimization algorithms the top-level graph model GA_p remain the same. Algorithms from the class \mathcal{MA} may differ with the control flow because they perform computations differently. So those algorithms have different bottom-level graph models. Therefore given the abstract metaheuristic optimization algorithm model GA_s , we can express only those properties of the algorithm A in the model GA_p that are essential only for a top-level graph model.

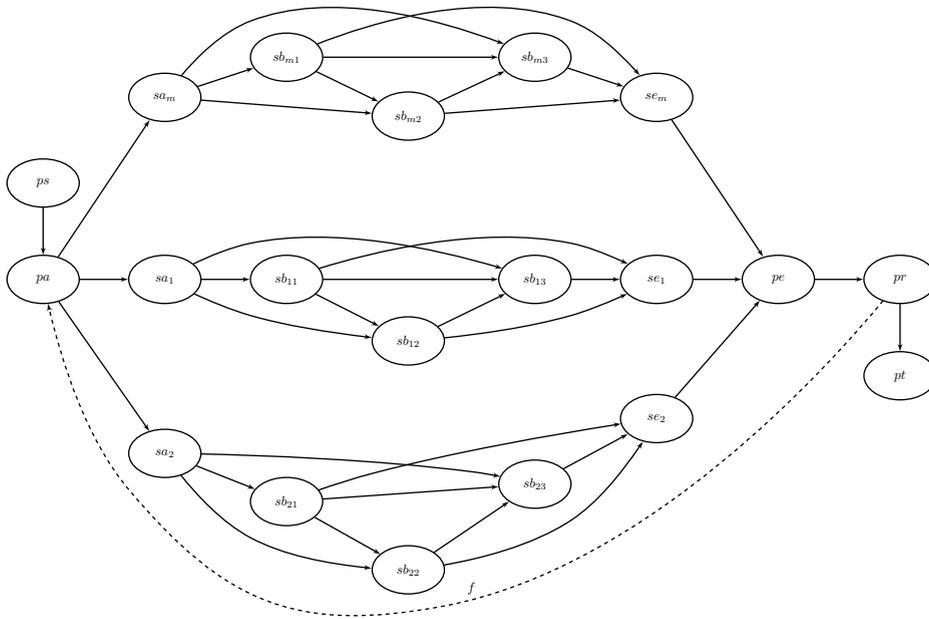


Fig. 3. An expanded graph GA_p^+ of an example algorithm A

3. Parametric hierarchical model

A metaheuristic optimization algorithm usually has free control parameters (also called strategy parameters). Let the BSP-based parallel algorithm have control parameters too, such as computational agents count, superstep duration and so on. Therefore we propose the parametric hierarchical model of a metaheuristic optimization algorithm.

Let the parallel algorithm A has a vector P of control parameters consisting of continuous and discrete parameters. A discrete parameter may describe structural characteristics of the algorithm (such as neighbourhood topology in a PSO algorithm). The vector of parameters P is split to the vector P_s of parameters of the bottom-level model and the vector P_p of parameters of the top-level model. We introduce the parametric metaheuristic algorithm $A_s(P_s)$ which is parametrized with a vector of parameters P_s . This vector should not be confused with a vector of parameters of the metaheuristic algorithm A_s itself.

We add the vector P into the model $GA(A)$ of an algorithm A . The vector P identifies a specific algorithm from a class of algorithms. Let $GA(A, P)$ be a parametric model of an algorithm A . Since the model GA is a hierarchical model, we have

$$GA(A, P) = GA(A, P_s, P_p).$$

Theorem 1. For any values of parameters P'_s , unexpanded graphs $GA(A, P'_s, P_p)$ are isomorphic.

Proof. Changing the values of parameters P_s of the sequential algorithm changes the graph $GA_s(P'_s)$ too. Since the graph GA is an unexpanded graph, by the Assertion 4 subgraphs GA_s are represented with vertices pd_i . Edges, which connects vertices pd_i with other vertices, remain the same for any value of parameter P'_s . Therefore, unexpanded graphs $GA(A, P'_s, P_p)$ are isomorphic. \square

Also, we introduce a parametric algorithm model based on the expanded graph GA^+ :

$$GA^+(A, P) = GA^+(A, P_s, P_p).$$

The graph expansion procedure (3) allows us to use the model $GA^+(A, P)$ in place of the model $GA(A, P)$ in an unambiguous way while the contrary is not true.

4. Hierarchical models of several metaheuristic optimization algorithms

4.1. Particle swarm optimization

Let the parametric graph model $GA(A_{PSO}, P_s, P_p)$ describe the particle swarm optimization algorithm A_{PSO} . Here the vector P_s of the sequential algorithm parameters consists of the following parameters:

- N — a number of particles in a population;
- $alpha, beta, gamma$ — various free parameters of the algorithm;
- $Neighb \in \{Lbest, Gbest, Thor, Cluster\}$ — a neighborhood topology.

The vector P_p of the parallel algorithm parameters consists of:

- NS — a number of sub-swarms (or islands);
- T — a migration season duration;
- $Mig \in \{Copy, Move\}$ — a migration strategy.

Operation class SE from the partition GA_s (1) for the PSO algorithm $A_{PSO} \in \mathcal{MA}$ includes an operation of choosing the best particle in the swarm based on the state of one algorithm step. For the parallel graph partition (2) an operation of class PE describes a particle migration strategy. An operation of class PR describes the migration procedure, and an operation of class PE includes a swarm state update.

Various modifications of the PSO algorithm also belongs to the class \mathcal{MA} . For example, several PSO algorithms with different neighborhood topology (Jordan [16]) have those graph models where graphs of sequential algorithms are subgraphs of the graph GA_s . In $gbest$ topology, a subgraph SB_i is a clique, while in the $lbest$ topology it is a ring graph.

4.2. Evolutionary strategy

Evolution strategy is based on an iterative change of the optimization vector with the mutation and recombination operators. The proposed model has the mutation operator in the operation class SB ; selection and recombination operators in the operation class SE . Operation classes of the parallel model are equivalent to those in an island model.

4.3. Evolutionary programming

The evolutionary programming algorithm differs from the evolutionary strategy by lacking a recombination operator. So the operation class SE includes a selection operator. The operation class SB includes a mutation operator.

5. Conclusion

In this paper, we proposed the hierarchical model of the parallel metaheuristic algorithm, which allows using graph theory methods for analysis and synthesis of sequential and parallel optimization algorithms. Graph models of algorithms are based on a flow graph model that is widely used for developing parallel programs. Procedures for expanding hierarchical graph, building an out-iterative and N -iterative graph has been described. The usage of the proposed parametric hierarchical model allows us to represent a whole class of metaheuristic algorithms with a single parametric model as well as to produce a parallel algorithm with different computational agents count.

The further work includes a research in methods of mapping a parallel algorithm onto an architecture of a parallel system based on the proposed hierarchical model. Annotating of sequential optimization programs to perform an automatic synthesis of the parallel algorithm is also an interesting application.

References

- [1] Kennedy J, Eberhart R. *Swarm intelligence*. Springer; 2006.
- [2] De Jong KA. *Evolutionary computation: a unified approach*. Springer; 2006.
- [3] Bäck T. *Evolutionary algorithms in theory and practice*. Oxford University Press, New York; 1996.
- [4] Belal M, El-Ghazawi T. Parallel models for particle swarm optimizers. *IJICIS* 2004;1:100–11.
- [5] Cantú-Paz E. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis* 1998;10(2):141–71.
- [6] Winter G, Greiner D, Galvan B. Parallel evolutionary computation. Tech. Rep.; CEANI; 2005.
- [7] Skillicorn DB, Talia D. Models and languages for parallel computation. *ACM Comput Surv* 1998;30(2):123–69.
- [8] Valiant LG. A bridging model for parallel computation. *Commun ACM* 1990;33:103–11.
- [9] Culler D, Karp R, Patterson D, Sahay A, Schauer KE, Santos E, et al. Logp: Towards a realistic model of parallel computation. In: *ACM Sigplan Notices*; vol. 28. ACM; 1993, p. 1–12.
- [10] Rechenberg I. Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. frommann-holzbog, stuttgart, 1973. *Step-Size Adaptation Based on Non-Local Use of Selection Information In Parallel Problem Solving from Nature (PPSN3)* 1994;.
- [11] Fogel DB. *Evolving artificial intelligence*. Ph.D. thesis; University of California at San Diego; 1992.
- [12] El-Abd M. *Cooperative models of particle swarm optimizers*. Ph.D. thesis; University of Waterloo; 2008.
- [13] Wang X, et al. *Hybrid nature-inspired computation methods for optimization*. Ph.D. thesis; Helsinki University of Technology; 2009.
- [14] Laguna-Sánchez G, Olguín-Carbajal M, Cruz-Cortés N, Barrón-Fernández R, Álvarez-Cedillo J. Comparative study of parallel variants for a particle swarm optimization algorithm implemented on a multithreading GPU. *Journal of Applied Research Technology* 2009;7(3):292–309.
- [15] Izzo D, Rucinski M, Ampatzis C. Parallel global optimisation meta-heuristics using an asynchronous island-model. In: *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*. IEEE; 2009, p. 2301–8.
- [16] Jordan J, Helwig S, Wanka R. Social interaction in particle swarm optimization, the ranked FIPS, and adaptive multi-swarms. In: *GECCO'08: Proceedings of the 10th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM. ISBN 978-1-60558-130-9; 2008, p. 49–56. URL: <http://doi.acm.org/10.1145/1389095.1389103>.