Dynamic Search Tree Growth Algorithm for Global Optimization

Ivana Strumberger, Eva Tuba, Miodrag Zivkovic, Nebojsa Bacanin, Marko Beko, Milan Tuba

▶ **To cite this version:**

Ivana Strumberger, Eva Tuba, Miodrag Zivkovic, Nebojsa Bacanin, Marko Beko, et al.. Dynamic Search Tree Growth Algorithm for Global Optimization. 10th Doctoral Conference on Computing, Electrical and Industrial Systems (DoCEIS), May 2019, Costa de Caparica, Portugal. pp.143-153, 10.1007/978-3-030-17771-3_12 . hal-02295249

HAL Id: hal-02295249

https://inria.hal.science/hal-02295249v1

Submitted on 24 Sep 2019

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Dynamic Search Tree Growth Algorithm for Global Optimization

Ivana Strumberger[1], Eva Tuba[1], Miodrag Zivkovic[1],
Nebojsa Bacanin[1], Marko Beko[2] and Milan Tuba[1]

[1] Singidunum University, Faculty of Informatics and Computing
Danijelova 32, 11000 Belgrade, Serbia
[2] COPELABS/ULHT, Lisbon, Portugal and UNINOVA, Caparica, Portugal
istrumberger@singidunum.ac.rs, etuba@singidunum.ac.rs, mzivkovic@singidunum.ac.rs,
nbacanin@singidunum.ac.rs, mbeko@uninova.pt, tuba@ieee.org

Abstract. This paper presents dynamic version of the tree growth algorithm. Tree growth algorithm is a novel optimization approach that belongs to the group of swarm intelligence metaheuristics. Only few papers addressed this method so far. This algorithm simulates the competition between the trees for resources such as food and light. The dynamic version of the tree growth algorithm introduces dynamical adjustment of exploitation and exploration search parameters. The efficiency and robustness of the proposed method were tested on a well-known set of standard global unconstrained benchmarks. Besides numerical results obtained by dynamic tree growth algorithm, in the experimental part of this paper, we have also shown comparative analysis with the original tree growth algorithm, as well as comparison with other methods, which were tested on the same benchmark set. Since many problems from the domains of industrial and service systems can be modeled as global optimization tasks, dynamic tree growth algorithm shows great potential in this area and can be further adapted for tackling many real-world unconstrained and constrained optimization challenges.

Keywords: tree growth algorithm, swarm intelligence, global, unconstrained, metaheuristics, dynamically adjusted parameters

1 Introduction

1.1 Motivation

In the todays era of digitalization, Internet of Things (IoT) and Cloud Computing, the industry and service sectors are transforming by introducing many innovations, where the most emphasized features of these innovations are so-called “exponential technologies”. This transformation is usually being expressed with the terms Industry 4.0, Smart Manufacturing and Economy 4.0.

In order to implement and to establish innovations in industry and service sectors, many problems need to be addressed. Most of such problems can be mathematically modelled and can be categorized as NP hard problems. For example, in the domains of Industry 4.0, cloud computing and IoT, many scheduling problems exist, that are characterized with NP hardness [1]. Also, the implementation of the wireless sensor networks (WSN) is very important in conducting innovations of industry and service sectors. Many problems, such is the problem of localization, from this domain belong to the NP hard group [2].

Another area which belongs to the group of NP hard problems is data clustering, along with data mining. Modern industry often relies on large amount of data. Large datasets are stored on the clusters, and focus of the algorithms is partitioning any dataset into an optimal number of groups through one run of optimization [3]. Modern industry also heavily depends on telecommunication networks. In order to keep the cost of the network reasonable, it is not possible to directly connect all communication nodes. Routing process is responsible for selection of the path between source and destination, by optimizing the objectives together with constraints. Any routing problem has the main objective to maximize the network performance [4].

For solving NP hard problems, it is not enough to use classic deterministic algorithms, since the results cannot be generated in a reasonable amount of time. For such purposes, it is better to employ stochastic methods like metaheuristics. Metaheuristic algorithms can be roughly divided into those who are inspired by the nature, and those who are not nature inspired. Nature-inspired metaheuristics can be further divided into evolutionary algorithms (EA) and swarm intelligence (SI).

Swarm intelligence is population-based, stochastic and iterative search methods that try to improve the population of candidate solutions in a predetermined number of iterations. These algorithms simulate the groups of natural organisms such as flock of birds and fish, groups of fireflies and bats, etc. One of the first swarm intelligence approaches was particle swarm optimization (PSO) that was developed in 1990s [5]. After the PSO, many other swarm intelligence algorithms emerged that were adapted for solving benchmark, as well as real-world NP hard optimization tasks. Some of the examples include artificial bee colony (ABC) [6], firefly algorithm (FA) [7], fireworks algorithm (FWA) [8], monarch butterfly optimization (MBO) [9], and many others [10].

According to the literature review, many swarm intelligence algorithms were successfully applied to the domains that are crucial for innovations of industry and service sectors. The PSO metaheuristics obtained satisfying results for solving job-shop scheduling problem in the era of industry 4.0 [1], as well as in some problems from the domain of IoT [11] and cloud computing [12], MBO algorithm can be applied to localization of unknown sensor nodes in some wireless sensor network topologies [13], etc.

Besides mentioned, swarm intelligence algorithms were also successfully applied to the domains of neural networks training [14], and image processing [15]. In recent years, the adaptations of swarm intelligence methods for image classification with convolutional neural networks (CNN) emerged [16]. These implementations may be crucial for digitalization.

Since the role of metaheuristics will be important in solving NP hard problems that will have to be tackled in the process of transformation of industry and service sectors,

and due to the fact that many of such problems can be formulated as bound-constrained and constrained problems global optimization problems, our motivation behind this work is to improve recently proposed tree growth algorithm (TGA) [17] for global optimization assignments.

In this paper, we propose dynamic version of the TGA swarm intelligence metaheuristics for solving global optimization problems. The approach was tested on a set of standard global benchmarks, and with only few modifications, enhanced TGA can be successfully applied to NP hard problems from the domain of innovations of industry and service sectors.

1.2 Research Question

According to the literature survey [18], [17], we concluded that the TGA can be successfully applied to solving range types of NP hard problems. The research question, which is the stepping stone of this paper, can be formulated as:

How to design and implement an efficient and robust TGA algorithm that will be capable of solving bound constrained and constrained global optimization NP hard problems and that will outperform other similar approaches tested for the same problem formulations?

In order to address the research question, we have formulated the following hypothesis:

TGA metaheuristics, as novel and promising method, can be further improved for solving global optimization NP hard problems. Once improved, this method can be easily adapted for solving different kinds of problems that innovations in industry and service systems face.

1.3 Related Work

TGA algorithm is novel approach that was developed in 2017 by Armin Cheraghalipour and Mostafa Hajiaghaei-Keshteli [17]. The algorithm was tested on standard global optimization benchmarks and on some engineering (constrained) benchmarks [17]. According to the literature review, only two papers that deal with this specific method exist in the literature.

On the other hand, since many NP hard problems that are crucial for innovations in industry and service systems were successfully tackled by employing swarm intelligence methods [3], [11], [12], and according to our analysis of the TGA metaheuristics, we concluded that the TGA can be further improved, and in the future research can be adapted for solving various kinds of problems from the domain innovations of industry and service systems.

1.4 Contributions

In this paper, we show an implementation of the improved tree growth algorithm (TGA) adopted for solving global optimization problems. Due to the fact that many problems

from the domains of industry and service sectors innovation can be formulated as global NP hard problems, the improved TGA in the unmodified, or slightly adjusted implementations, can be easily applied to these problems as well, and this represents the clear contribution of this paper for this research domain. Also, we should emphasize that the TGA was not explored well, since only two papers that deal with this approach exist in the literature.

2 Relationship to Technological Innovation for Industrial and Service Systems

Currently, the 4th industrial revolution is taking place. The idea of this new revolution is represented by the terms Industry 4.0, Smart Manufacturing and Economy 4.0. During this revolution, many innovations of industrial and service systems will take place. Cloud computing, Internet of things, WSNs, and the integration of "exponential technologies" play an important role in these innovations. Also, during these innovations, increasing digitalization and interconnection of systems, products, value chains, and business models is needed.

However, in order for innovations to be successfully implemented, many problems should be solved. As already, stated in the Section 1, most of such problems can be mathematically modelled and such they belong to the category of NP hard optimization. Since the swarm intelligence methods have proved to be efficient in solving NP hard problems, these methods can be applied for tackling NP hard problems when innovating industrial and service systems.

For example, cloud computing may be the production platform for technological innovations for industrial and service systems. According to the literature review, load balancing [19], as well as scheduling problems [20] in cloud computing environments were successfully tackled by using swarm intelligence metaheuristics. Swarm intelligence has also been successfully applied in numerous optimizations in telecommunications and routing systems [4].

3 Basic and Enhanced Tree Growth Algorithm

TGA is novel swarm intelligence metaheuristics developed in 2017 by Armin Cheraghalipour and Mostafa Hajiaghaei-Keshteli [17]. TGA models the competition between the trees in the nature for acquiring light and food sources.

Inspired by the nature, one iteration of the TGA is divided into four phases. In the first phase, N_1 better solutions (individuals) in the population perform the local search process. In the second phase, N_2 solutions are moved to the distance between the close best solutions under different α angles. Third phase is conducted by discarding N_3 worst solutions from the population, which are replaced by randomly generated solution from the feasible domain of the search space. Finally, in the fourth phase, N_4 new solutions are generated, when each newly created solution is modified by the mask operator respect to the best solution in the set N_1 that consists of best solutions in the population.

One execution (run) of the TGA can be summarized in the eight following steps:

Step 1: Generate random initial population of N candidate solutions within the values of lower and upper parameters' bound and calculate fitness of each solution. In the case of minimization problems, the fitness is reverse proportional to the value of objective function.

Step 2: Sort population according to the value of fitness and find the current best solution T^j_{GB}, where j-th denotes the current iteration.

Step 3: For N_1 better solutions in the population perform the local search for all solutions' parameters according to the Eq. (1) [17].

$$\boldsymbol{T_i^{j+1} = \frac{T_i^j}{\theta} + rT_i^j} \tag{1}$$

where T_i^j and T_i^{j+1} denote the i-th solution in the population in the iterations j (old solution i) and $j+1$ (new solution i), respectively, θ is the reproduction rate parameter, and r is uniformly distributed pseudo-random number between 0 and 1. When the new solution is generated, the selection between old and new solution is performed using greedy mechanism (the solution with the higher fitness value is retained).

Step 4: N_2 solutions from the population are moved towards the two closest better solutions from the subpopulation N_1 under different α angles. To find the closest best solutions, the following equation is applied for all solutions' parameters [17]:

$$d_i = \sqrt{\sum_{i=1}^{N_1}\left(T_{N_2}^j - T_i^j\right)^2}, \text{where } d_i = \begin{cases} d_i & if\ T_{N_2}^j \neq T_i^j \\ \infty & if\ T_{N_2}^j \neq T_i^j \end{cases} \tag{2}$$

When the distance is calculated, then two solutions x_1 and x_2 with the minimal distance (d_i) from each solution $T_{N_2}^j$ are selected for producing linear combination by using the equation [17]:

$$y = \lambda x_1 + (1-\lambda)x_2 \tag{3}$$

where λ represents the TGA control parameter and its value is between 0 and 1. Finally, all solutions $T_{N_2}^j$ from the subpopulation N_2 are moved between two adjacent solutions with α different angles by employing Eq. (4) [17].

$$T_{N_2}^j = T_{N_2}^j + \alpha_i y \tag{4}$$

Step 5: The N_3 worst solutions are discarded from the population and they are replaced with the randomly generated solutions from the feasible region of the search space.

Step 6: New population N is created, where $N = N_1 + N_2 + N_3$

Step 7: In this step, new set of N_4 randomly distributed solutions are created and each solution is modified by using the mask operator respect to the best solution from the subpopulation N_1. Solutions from the N_4 subpopulation are then added to the population N.

Step 8: Population $N + N_4$ is sorted according to the fitness and the best N solutions are chosen as the initial population for the next iteration of the algorithm. In this step, tournament or roulette wheel selection process is employed.

In the presented steps, N, N_1, N_2, N_4, θ and λ are the control parameters of the TGA.

3.1 Dynamic Tree Growth Algorithm

Two basic processes of any swarm intelligence algorithm, that guide the search, are the exploitation (intensification) and exploration (diversification). To obtain the satisfactory results, the balance between these two processes should be adjusted well.

If the balance between exploration and exploitation in not properly balanced in favor to exploitation, the algorithm may converge to suboptimal solutions in early iterations, and may be trapped in the local optimum, which is called the premature convergence. At the other hand, if this balance is in favor of exploration, it can happen that the algorithm may find the right part of the search space, but may not be able to fine tune around the current best solutions. As a consequence, the algorithm cannot find the optimum solution.

The exploration and exploitation in the original TGA are controlled by two control parameters θ and λ, and the number of solutions N_3, which are discarded from the population in each iteration. The θ parameter directs the local search process, while the λ parameter directs solutions towards the better solutions in the population. Both parameters conduct the process of exploitation. At the other hand, by adjusting N_3, the process of exploration is conducted. All three parameters are static during the whole course of algorithm's execution.

By conducting empirical tests, we concluded that the TGA performance can be improved by dynamically adjusting the values of θ and N_3. In early iterations of algorithm's execution, the value of the θ parameter should be lower, and the newly generated solution should be further from the current solution (Eq. (1)). Basic assumption is that the algorithm in early iterations has not found the right part of the search space. However, in later iterations, the value of the θ parameter should be higher, with the assumption that the algorithm has found the proper part of the search space. The value of the parameter λ should be static, since the nature of Eq. (3).

Also, the number of solutions that are discarded from the population (parameter N_3) should be higher in early iterations, since in this phase more exploration power is needed. The change of N_3 parameter is directly influenced by the values of N_1 and N_2.

By incorporating dynamic behavior of θ and N_3, we devised dynamic search TGA (dynsTGA) metaheuristics. Pseudo-code of dynsTGA is shown in Algorithm 1.

Algorithm 1 Pseudo-code of the dynsTGA metaheuristics

Initialization. Generate pseudo-random population, set the iteration counter $t=1$, maximum iteration number *MaxIter*, and initial values for θ and N_3 control parameters
while $t <$ *MaxIter* **do**
  Evaluate population and sort all solutions according to their fitness
  for all solutions in N_1
    Perform local search by using Eq. (1)
    Apply greedy selection between old and new solution
  end for
  for all solutions in N_2

Move solutions towards the closest best solutions in N_1 by using Eqs. (2) – (4)
Apply greedy selection between old and new solution
end for
Discard N_3 worst solutions from the population and replace them with pseudo-random solutions
Generate N_4 randomly distributed solutions and modify each solution in respect to the best solutions in N_1 by using the mask operator
Evaluate population and sort all solutions according to their fitness
Choose N solutions for the new iteration
Adjust the values of θ and N_3 along with N_2 and N_1 control parameters
end while
return the best solution in the population

4 Empirical Results and Discussion

For the purpose of the research presented in this paper we devised our own software framework by using .NET 4.5 technology and C# in Visual Studio 2017 Integrated Development Environment. In the framework we incorporated both, original TGA and dynsTGA, along with the benchmark functions.

To evaluate the performance of the dynsTGA, we used similar parameter values as in [18] and [17]. The size of initial population (N) was set to 100, while N1 and N2 were set to 20. In the beginning of the algorithm's execution, the value of N3 was set to 60 (N3 = N-(N1 + N2)). The value of N4 was set to 30 and it was fixed during the whole course of the algorithm's execution. The maximum iteration number (MaxIter) was set to 250, and the algorithm was executed in 30 independent runs.

In the first 210 iterations, the values of N_1, N_2 and N_3 were fixed. Then, in each of the following iterations (from iteration 211 to iteration 250), the value of N_3 was decremented by one, and the values of N_1 and N_2 were incremented by 1 in even and odd iterations, respectively. The value of the λ parameter was set to 0.5, while the initial value of θ was set to 0.2. In each iteration, the value of θ was adjusted according to the following expression: $\theta^{j+1} = \theta^{j} \cdot 1.002$, until the threshold value of 1.5 is reached.

We tested dynsTGA on a standard set of global optimization benchmarks and compared results with the original TGA, as well as with other state-of-the-art approaches that were tested on the same benchmark test. Details of benchmark functions are given in Table 1.

Table 1. Benchmark function details

ID	Name of the problem	Dim.	Type	Parameter range
F1	Ackley's Problem (ACK)	10	Multimodal	(-30,30)
F2	Aluffi–Pentini's Problem (AP)	2	Multimodal	(-10,10)
F3	Becker and Lago Problem (BL)	2	Multimodal	(-10,10)
F4	Easom Problem (EP)	2	Unimodal	(-10,10)
F5	Rastrigin Problem (RG)	10	Multimodal	(-5.12,5.12)
F6	Rosenbrock Problem (RB)	10	Multimodal	(-30,30)
F7	Goldstein and Price Problem (GP)	2	Multimodal	(-2,2)
F8	Gulf Research Problem (GRP)	2	Unimodal	(0,100)

Algorithm was tested in 30 independent runs. We performed comparative analysis with outstanding stochastic algorithms from the literature that were tested on the same benchmark set, and compared obtained values for best, mean and standard deviation performance indicators. Comparative analysis is presented in Table 2, where the best results for each performance indicator and each benchmark test are bolded.

First thing that we want to emphasize is that from the Table 2, it is clear that the dynsTGA outperforms original TGA metaheuristics. For example, in F3 (BL Problem) benchmark, dynsTGA showed better performance for best, mean, as well as for standard deviation indicator. Similar, in the case of F8 (GRP) test, dynsTGA showed significantly better robustness than the basic TGA. The improvements over unmodified TGA are also significant in the F6 test, where dynsTGA obtained better values.

As can be seen from Table 2, comparative analysis with IBPA [17], LADA [17], TS [17] and WSA [17] was performed. In average, dynsTGA approach performs better than all other metaheuristics that are included in comparative analysis. The superiority of dynsTGA can be best pictured in the F3 (BL Problem) benchmark, where dynsTGA completely outperformed all other approaches for best, mean and standard deviation indicators.

5 Conclusion

In this paper, a modified and improved version of relatively new tree growth algorithm (TGA) metaheuristics was presented. Only few papers from the literature deal with the TGA approach. Original TGA was enhanced by introducing dynamical adjustment of exploitation and exploration search parameters. The performance of enhanced TGA metaheuristics was measured on standard set of global optimization benchmarks. In the experiments, similar parameter setup as in [17] was applied.

To prove the robustness and solutions' quality of the dynamic TGA, comparative analysis with the basic TGA, as well as with other state-of-the-art algorithms for global optimization was conducted. From the presented side-by-side comparison, it is obvious that the dynamic TGA significantly improves original TGA and performs better than other algorithms that were included in analysis.

Since many problems from the domains of industrial and service systems can be modeled as global optimization tasks, dynamic TGA metaheuristics shows great potential in this area and can be adapted for tackling many real-world unconstrained and constrained optimization challenges.

Table 2. Simulation results for performance indicators and comparative analysis.

ID	Indicator	IBPA	LADA	TS	WSA	TGA	dynsTGA
	Best	0.00815	0.00088	0.14185	0.888E−15	0.00	0.00
F1	Mean	0.02260	0.00473	0.38528	0.888E−15	0.00	0.00
	StdDev	0.01021	0.00157	0.07488	1.0029E−31	0.00	0.00
	Best	−0.35238	−0.35238	−0.35238	−0.35238	−0.35239	−0.35239
F2	Mean	−0.35238	−0.35238	−0.35238	−0.35236	−0.35239	−0.35239
	StdDev	1.067E−6	5.576E−7	2.183E−5	8.761E−6	6.09E−07	**3.53E-09**
	Best	3.217E−9	1.259E−9	3.955E−7	5.589E−8	1.07E−08	**0.00**
F3	Mean	2.826E−7	2.486E−7	7.637E−6	1.267E−7	3.70E−07	**2.04E-09**
	StdDev	2.838E−7	2.704E−7	6.302E−6	3.877E−8	5.84E−07	**5.11E-10**

	Best	−0.99999	−0.99999	−0.99999	−0.99999	−0.99999	−0.99999
F4	Mean	−0.83334	−0.99999	−0.46667	−0.99957	−0.99999	−0.99999
	StdDev	0.379010	**2.885E−6**	0.507330	2.025E−4	1.59E−06	8.45E-05
	Best	0.08790	0.00606	4.58753	0.00	0.00	0.00
F5	Mean	0.29275	0.01584	6.35541	0.00	0.00	0.00
	StdDev	0.12481	0.00554	0.89405	0.00	0.00	0.00
	Best	1.6578	13.1161	24.7395	8.9167	0.5653	**0.5361**
F6	Mean	12.1420	26.4740	66.1024	8.9449	**0.8231**	0.8502
	StdDev	14.9202	14.9521	19.1763	0.0160	0.3419	**0.3137**
	Best	3.00000	3.00000	3.00000	3.00000	3.00207	3.00000
F7	Mean	5.70001	10.00710	3.00053	**3.00032**	3.11253	3.00068
	StdDev	8.23847	16.46670	**5.751E−4**	1.622E−4	1.34E−01	7.305E−4
	Best	5.399E−6	8.124E−5	3.120E−5	32.83	1.52E−05	**8.45E-06**
F8	Mean	**0.00157**	5.362E−4	2.047E−4	32.83	7.05E−02	3.50E-02
	StdDev	**0.00162**	3.456E−4	1.382E−4	1.445E−15	2.60E−01	1.72E-01

Acknowledgements: This research is supported by the Ministry of Education, Science and Technological Development of Republic of Serbia, Grant No. III-44006. The work of M. Beko was supported in part by Fundao para a Cincia e a Tecnologia under Projects PEst-OE/EEI/UI0066/2014 (UNINOVA) and Program Investigador FCT (IF/00325/2015).

References

1. Leusin, M.E., Frazzon, E.M., Maldonado, M.U., Kück, M., Freitag, M.: Solving the Job-Shop Scheduling Problem in the Industry 4.0 Era, Technologies, Vol. 6, Issue 4, MDPI (2018), doi: https://doi.org/10.3390/technologies6040107.
2. Strumberger, I., Beko, M., Tuba, M., Minovic, M., Bacanin, N: Elephant Herding Optimization Algorithm for Wireless Sensor Network Localization Problem. In: Camarinha-Matos L., Adu-Kankam K., Julashokri M. (eds) Technological Innovation for Resilient Systems. DoCEIS 2018. IFIP Advances in Information and Communication Technology, Vol 521, pp 175-184, Springer, Cham (2018).
3. Abraham, A., Das S., Roy, S.: Swarm Intelligence Algorithms for Data Clustering. In: Maimon O., Rokach L. (eds) Soft Computing for Knowledge Discovery and Data Mining. Springer, Boston, MA, pp. 279-313 (2008).
4. Ducatelle, F., Gianni A. D., Luca, M.G.: Principles and applications of swarm intelligence for adaptive routing in telecommunications networks, Swarm Intelligence Vol. 4, Issue 3, pp. 173-198 (2010).
5. Kennedy, J., Eberhart, R.: Particle swarm optimization, Proceedings of ICNN'95 - International Conference on Neural Networks, Perth, WA, Australia, pp. 1942-1948 (1995), doi: 10.1109/ICNN.1995.488968.
6. Bacanin, N., Tuba, M.: Artificial bee colony (ABC) algorithm for constrained optimization improved with genetic operators, Studies in Informatics and Control, Vol. 21, Issue 2, pp. 137-146 (2012).
7. Yang,X.-S., He, X.: Firefly Algorithm: Recent Advances and Applications, Int. J. Swarm Intelligence, Vol. 1, No. 1, pp. 36-50 (2013), doi: 10.1504/IJSI.2013.05580
8. Strumberger, I., Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Bare Bones Fireworks Algorithm for the RFID Network Planning Problem, 2018 IEEE Congress on Evolutionary Computation (CEC), Rio de Janeiro, pp. 1-8 (2018), doi: 10.1109/CEC.2018.8477990.

9. Wang, G.-G., Deb, S., Cui, Z.: Monarch butterfly optimization, Neural Computing and Applications, pp. 1–20 (2015).
10. Tuba, M., Bacanin, N.: Hybridized bat algorithm for multi-objective radio frequency identification (RFID) network planning, 2015 IEEE Congress on Evolutionary Computation (CEC), Sendai, pp. 499-506 (2015), doi: 10.1109/CEC.2015.725693.
11. Nouiri, M., Jemai, A., Ammari, A.C., Bekrar, A., Trentesaux D., Niar, S.: Using IoT in breakdown tolerance: PSO solving FJSP, 2016 11th International Design & Test Symposium (IDT), Hammamet, pp. 19-24 (2016), doi: 10.1109/IDT.2016.7843008.
12. Masdari, M., Salehi, F., Jalali, M. et al.: A Survey of PSO-Based Scheduling Algorithms in Cloud Computing, Journal of Network and Systems Management, Vol. 25, Issue 1, pp. 122-158 (2017), doi: https://doi.org/10.1007/s10922-016-9385-9.
13. Strumberger, I, Tuba, E., Bacanin, N., Beko, M., Tuba, M.: Monarch butterfly optimization algorithm for localization in wireless sensor networks, 2018 28th International Conference Radioelektronika (RADIOELEKTRONIKA), Prague, pp. 1-6 (2018), doi: 10.1109/RADIOELEK.2018.8376387.
14. Tuba M., Alihodzic A., Bacanin N.: Cuckoo Search and Bat Algorithm Applied to Training Feed-Forward Neural Networks. In: Yang XS. (eds) Recent Advances in Swarm Intelligence and Evolutionary Computation. Studies in Computational Intelligence, Vol. 585, pp. 139-162, Springer, Cham (2018).
15. Tuba, E., Alihodzic, A., Tuba, M.: Multilevel image thresholding using elephant herding optimization algorithm, 2017 14th International Conference on Engineering of Modern Electric Systems (EMES), Oradea, pp. 240-243 (2017), doi: 10.1109/EMES.2017.7980424.
16. França da Silva, G.C., Valente, T.L.A., Silva, A.C., Cardoso de Paiva, A., Gattass, A.: Convolutional neural network-based PSO for lung nodule false positive reduction on CT images, Computer Methods and Programs in Biomedicine, Vol. 162, pp. 109-118 (2018), doi: https://doi.org/10.1016/j.cmpb.2018.05.006.
17. Cheraghalipour, A., Hajiaghaei-Keshteli, M: Tree Growth Algorithm (TGA): An Effective Metaheuristic Algorithm Inspired by trees' behavior, 13th Int. Conf. on Industrial Engineering, Vol. 13 (2017).
18. Cheraghalipour, A., Hajiaghaei-Keshteli, M., Paydar, M.M.: Tree Growth Algorithm (TGA): A novel approach for solving optimization problems, Engineering Applications of Artificial Intelligence, Vol. 72, pp. 393-414 (2018), doi: https://doi.org/10.1016/j.engappai.2018.04.021.
19. Li, D., Li, K., Liang, J., Ouyang, A.: A hybrid particle swarm optimization algorithm for load balancing of MDS on heterogeneous computing systems, article in press, Neurocomputing (2018), doi: https://doi.org/10.1016/j.neucom.2018.11.034.
20. Kalra, M., Singh, S.: A review of metaheuristic scheduling techniques in cloud computing, Egyptian Informatics Journal, Vol. 16, Issue 3, pp. 275-295 (2015), https://doi.org/10.1016/j.eij.2015.07.001.