The Architecture of a Software Framework for Biologically-Inspired Optimization Algorithms

Florin Leon
Faculty of Automatic Control and Computer Engineering
"Gheorghe Asachi" Technical University of Iași
Iași, Romania
e-mail: florin.leon@academic.tuiasi.ro

Silvia Curteanu
Faculty of Chemical Engineering and Environmental Protection
"Gheorghe Asachi" Technical University of Iași
Iași, Romania
e-mail: silvia.curteanu@academic.tuiasi.ro

***Abstract*—Biologically-inspired optimization algorithms are robust techniques that can be applied for a wide range of practical problems. A large number of such algorithms have been proposed by researchers, but their implementation is often not available or is done in different programming languages. This paper presents a flexible software architecture of a .NET optimization framework where such methods can be easily incorporated. It includes algorithms that belong to various subfields: from ideas based on human social behavior to ideas based on virus behavior. Using this framework, many modifications and hybridizations for algorithms are also possible.**

Keywords-biologically-inspired optimization algorithms; optimization framework; software architecture; .NET framework.

I. INTRODUCTION

Biologically-inspired optimization algorithms have become popular in the recent years as general, simple, and robust techniques that can be used when other mathematical optimization methods cannot be applied. Among their advantages, which facilitate their application to various fields, such as planning, design, control, classification, clustering, and time series modeling, one can mention the following [1]:

- They do not require the objective function to be continuous and/or differentiable;
- They do not require extensive problem formulation, e.g., in case of traditional methods such as integer programming, geometric programming, as well as branch and bound methods, a special mathematical formulation is required for solving a problem;
- They are not sensitive to the starting point of the search;
- They are more robust in the presence of local optima and are more likely to find the global optimum of a function than gradient-based methods [2][3].

In this work, the architecture of an optimization framework called HAVOC for biologically-inspired optimization will be presented. The abbreviation comes from the title of the project: *From Human to Virus. Optimization Algorithms with Chemical Engineering Applications*. It is one of the few optimization frameworks available for .NET, which incorporates biologically-inspired algorithms that belong to a wide range of subfields: from ideas based on human social behavior to ideas based on virus behavior. While many such algorithms have been proposed and some have their implementation publicly available, they are not integrated into a unified framework.

The motivation of dealing with algorithms inspired by human and virus behavior can be seen from several perspectives:

- A relatively new, little studied field of interest is approached, which creates good scientific perspectives in the sense of identifying directions that can contribute to stimulating creativity and innovative ideas;
- There are also few applications of these algorithms, so future work will include applying these algorithms in a new field, i.e., chemical engineering (more precisely, polymerization engineering), with processes that represent a challenge through the complexity and difficulty of the problems they raise in the actions of modeling and optimization.
- A systematic, detailed study of these algorithms is intended, from basic variants to modified versions, from general to the particular, adapting to specific applications and including comparisons with previous achievements, but also with approaches from the literature.

The rest of this paper is organized as follows. In Section II, some related work in terms of other optimization libraries available today is addressed. Section III describes the architecture of the HAVOC optimization framework. The actual algorithms are briefly presented in Section IV. A discussion of future work and the conclusions are included in Section V.

II. RELATED WORK

Presently, there are a number of evolutionary/genetic algorithm libraries available, such as: *Evolving Objects: an Evolutionary Computation Framework* [4][5], *Genetic Algorithms Framework* [6], *Watchmaker Framework for Evolutionary Computation* [7], *AForge.NET Genetic Algorithms Library* [8], as well as many others.

The *SciPy optimize* [9] package contains strategies such

 ISBN: 978-1-61208-882-2

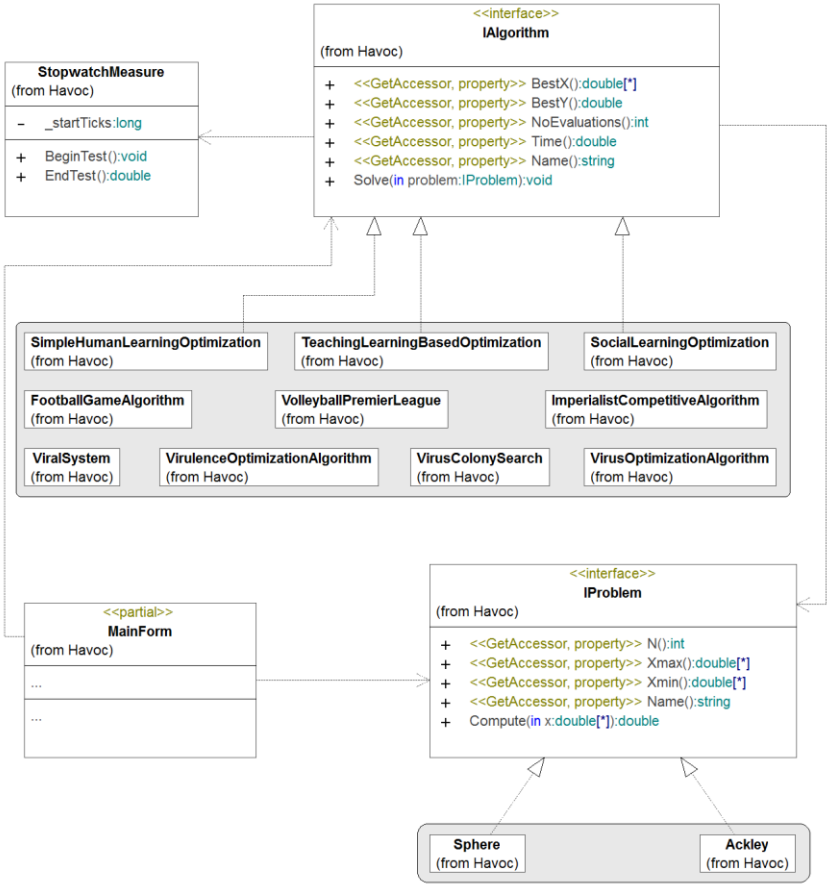

Figure 1. The architecture of the HAVOC framework (UML class diagram).

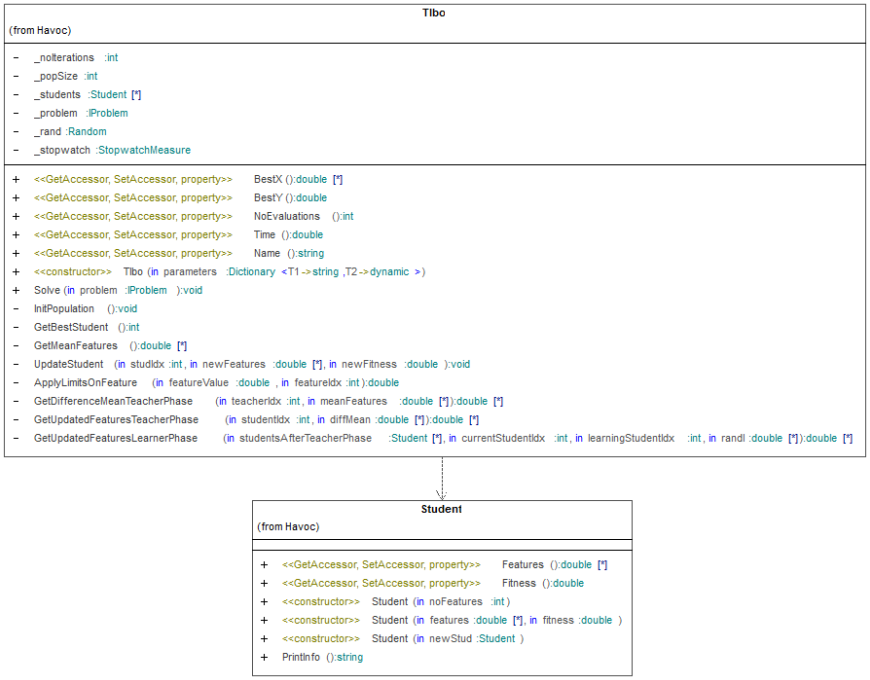

Figure 2. The implementation of the *Teaching-Learning based Optimization* algorithm (UML class diagram).

as Nelder-Mead, Powell, Quasi-Newton strategies, etc., and can also handle linear and nonlinear constraints. *lpsolve* [10] is another powerful mixed integer linear programming solver. *MathWorks* also provides an *Optimization Toolbox* [11] for Matlab, which can solve linear, quadratic, conic, integer, and nonlinear optimization problems. A general evolutionary algorithm framework especially designed for multi-agent systems is presented in [1].

Other optimization systems that can be mentioned are: ECJ [12] and jMetal [13], programmed in Java, DEAP [14], implemented in Python, and HeuristicLab [15]. From these tools, only the last one uses .NET. It also allows the dynamic inclusion of additional algorithms as plug-ins. However, these frameworks do not contain yet the classes of algorithms envisaged to be included in the proposed HAVOC system.

III. SYSTEM ARCHITECTURE

The architecture of the HAVOC framework is presented in Figure 1 as a *Unified Modeling Language* (UML) class diagram. The purpose of the framework is to allow the user to experiment with different algorithms and problems in a very simple way. The user selects a problem and an algorithm, and the framework solves the problem using the specified algorithm. In order to be able to handle this, all ten algorithm classes implement the common *IAlgorithm* interface. Similarly, all the problem classes implement the *IProblem* interface. In this way, the corresponding code is common for all combinations of problems and algorithms:

```
IProblem problem = new XProblem();
IAlgorithm algorithm = new YAlgorithm(parameters);
algorithm.Solve(problem);
```

Afterwards, the solution and additional information can be retrieved from the algorithm object. The *BestX* property represents the "*x*" that minimizes *f*(*x*); it is a vector of real numbers. The *BestY* property is a real number that represents the "*y*", where *y* = *f*(*x*).

Other useful characteristics are the number of function evaluations that the algorithm performs. This is one of the recent ways to compare the performance of heuristic optimization algorithms. The actual execution time is also provided, using the *Stopwatch* class from the .NET framework.

Each algorithm object is instantiated with a set of parameters. More specifically, the parameters are given as a *Dictionary* (a generic hash table collection), where the key is the name of the parameter and the value is the corresponding parameter value:

```
var parameters = new Dictionary<string, dynamic> {
   ["NoIterations"] = 1000, ["PopSize"] = 100 };
IAlgorithm alg = new Tlbo(parameters);
alg.Solve(currentProblem);
// alg.BestX, alg.BestY, alg.NoEvaluations, alg.Time are available
```

These values are not restricted to be real numbers, they can have any desired type. The user will be able to modify the parameter values from the graphical user interface or directly by editing a configuration file.

Each algorithm is implemented in it own class (or classes); e.g., in Figure 2, the UML class diagram for the *Teaching-Learning based Optimization* (TLBO) algorithm is presented. Figure 3 shows the results obtained with TLBO for several well-known benchmark problems: *Sphere*, *Ackley*, *Griewank*, and *Rosenbrock* [16]. They were obtained using a computer with a 4-core 2 GHz Intel processor and 8 GB of RAM.

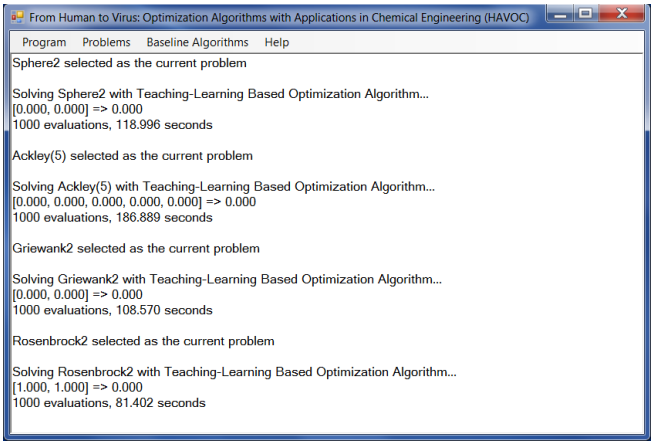

Figure 3. The results obtained with the *Teaching-Learning based Optimization* algorithm for several benchmark problems.

The *IProblem* interface defines the main characteristics of an optimization problem: its dimensionality (the *N* property), the search range defined by the lower bounds and the upper bounds of each dimension (the *XMin* and *XMax* properties), and the actual implementation of the function to be optimized in the *Compute* method. The optimization problems need not be simple, analytical functions. They can be complex, simulation-based procedures, e.g., the minimum distance from a target when launching a space probe with certain initial properties, such as angle, speed and acceleration. The trajectory of this object can be simulated in a discrete, step-by-step fashion, taking into account the resulting force applied by the large space objects on the probe. As long as the simulation returns a value (in this example, the ideal value is 0, i.e., the probe has reached its target), it can be used as an optimization function in the proposed framework.

In the present version, the architecture only supports solutions encoded as an array of real numbers, but solutions represented as an array of integers can also be found. This would require that the user make a genotype-phenotype distinction, where the actual encoding genes are not used as such in the computation of the objective function but they are transformed or interpreted in a certain way. E.g., real numbers can be interpreted as integers, by rounding them or using random key encoding for problems that have permutations as solutions. In future versions, constraints can be included as C# predicates that can be passed as

parameters to the algorithms. Variable length solutions can also be employed using the same array-based representations. However, these changes in the architecture should also be supported by the underlying algorithms.

IV. THE ALGORITHMS

The included algorithms are grouped into three main categories, as shown in Table I. They are discussed in the following subsections.

TABLE I. THE ALGORITHMS SELECTED FOR IMPLEMENTATION

Category	Algorithm	Encoding
Algorithms inspired by the human behaviors of learning and cooperation	*Simplified Human Learning Optimization* (SHLO) [17]	*binary*
	Social Learning Optimization (SLO) [18]	*real*
	Teaching-Learning based Optimization (TLBO) [19]	*real*
Algorithms inspired by human competitive behavior	*Football Game Algorithm* (FGA) [20][21]	*real*
	Volleyball Premier League (VPL) [22]	*real*
	Imperialist Competitive Algorithm (ICA) [23]-[25]	*real*
Algorithms inspired by virus behavior	*Viral System* (VS) [26]	*binary*
	Virulence Optimization Algorithm (VOA1) [27]	*real*
	Virus Colony Search (VCS) [28]-[30]	*real*
	Virus Optimization Algorithm (VOA2) [31]-[33]	*real*

A. Algorithms inspired by the human behaviors of learning and cooperation

Many learning activities are similar to the meta-heuristic search, where the success is achieved through repetition and adjusting. Although the human learning process is very complex, the three main strategies used in human learning include: random learning, individual learning and social learning. SHLO includes all three strategies to search for good solutions. On the other hand, SLO simulates the social learning approach and it is based on the *Culture Algorithm* framework (CA) [34] comprising three elements: population space, belief space and a protocol describing the manner in which knowledge is exchanged between the two spaces. To this CA framework, SLO adds an additional bottom layer that includes individual genetic evolution. In case of the TLBO algorithm, the simulated process is represented by the transfer of knowledge between the teacher and the student, i.e., the teaching phase, and the collaboration between students, i.e., the learning phase. Very few variants and combinations have been developed for the first two algorithms: adaptive SHLO [35][36] and SLO with *Differential Evolution* and improved *Social Cognitive Optimization* [37]. TLBO is the most used algorithm due to its simplicity and efficiency [38], and has been included in various hybrid variants: TLBO with error correction strategy and Cauchy distribution [39], TLBO including genetic crossover and mutation strategies [40], TLBO with *Bird Mating Optimizer* [41], TLBO with *Differential Learning* [42] and more.

B. Algorithms inspired by human competitive behavior

In addition to the concept of learning associated with humans, the concept of competitiveness is studied. FGA mimics the manner in which football is played. The population is represented by the players and the optimization is performed in two steps: random walk and coaching, with two strategies: attacking and substitution. Distinctively from FGA, VPL closely follows the rules of volleyball and the solution is formed from two segments: active (representing the main formulation of each team) and passive (which stores variables and special instruction rules). If FGA and VPL are team-based, in the sense that they include a reduced number of players, ICA simulates the competition behavior at the "national" level, where each individual represents a country and there are two types of countries: imperialist and colonies. For the first two algorithms, very few improvements and applications have been published: modified VPL using sine cosine algorithm [43] and *Multi-Objective Volleyball Premier League* [44]. By comparison, ICA is a more mature algorithm, with many applications, known in different variants, of which one can mention: ICA with different chaotic maps [45], ICA with k-means clustering [46], ICA with neural networks [47][48], etc.

C. Algorithms inspired by virus behavior

VS is based on the viral infection processes and, to reach the optimum, it uses two mechanisms: replication and infection. Depending on the type of virus infection, i.e., selective or massive infection, and on the type of evolution of the virus, different mechanisms are activated during the search. VS stops when the collapse and death of the organism occurs or the virus is isolated. On the other hand, VOA1 simulates several important mechanisms in the virus life-cycle: reproduction and mutation, cloning, and escaping from the infected region. In the case of VCS, the virus diffusion, cell infection and immune response are simulated using Gaussian random walk, *Covariance Matrix Adaptation Evolution Strategy and Evolution Strategy*. VOA2 mainly focuses on converting the concept of a virus attacking a host cell into a continuous domain optimization method. In general, there are few approaches and applications of these algorithms, with a single modified variant [49].

V. CONCLUSIONS AND FUTURE WORK

In this work, the architecture of an optimization framework called HAVOC was presented. It is extensible, so that other algorithms and variants of the base algorithms should be easy to add. In this way, it facilitates comparisons on both benchmark problems and real-life problems from the chemical engineering field. HAVOC uses .NET (C#) and includes classes of algorithms less often implemented in other publicly available frameworks.

Since the open literature presents too few variants and applications of algorithms that mimic the human and virus behaviors, our future research aims to carry out an in-depth, systematic study, accompanied by rigorous tests and various applications, bringing the following as elements of novelty and originality:

 ISBN: 978-1-61208-882-2

- New variants of the optimization algorithms inspired by human, animal and virus behaviors;
- Combinations between different categories of algorithms such as neural networks, support vector machines, evolutionary algorithms, artificial immune systems, etc., leading to new hybrid configurations with improved performance, which can open new perspectives for the advanced modeling and optimization techniques;
- Association of the developed methodologies with new real-world case studies;
- Innovative implementation of the proposed methods by original software.

The new variants of the algorithms, including new concepts, the hybrid configurations of the artificial intelligence tools, and the use of bio-inspired modeling and optimization methodologies for chemical processes can be considered as new approaches, which open new fruitful directions of research and applications in this field. At the same time, the flexibility of the algorithms, the way in which the implementations will be made, their possibilities of adaptation, give them generality and, therefore, possibilities to be applied in various other fields.

Possible applicative research directions derive from the chemical processes considered as case studies for the proposed algorithms. The results of the modeling and optimization procedures provide useful information for experimental and industrial practice, e.g.:

- They can substitute or better schedule the experiments that are materials-, energy- and time-consuming;
- They emphasize the maximum performance of the systems and the conditions necessary to accomplish them;
- These achievements bring important economic benefits, as the use of the developed modeling and optimization techniques represent an important stage in optimal control engineering.

ACKNOWLEDGMENT

This work was supported by Exploratory Research Projects PN-III-P4-ID-PCE-2020-0551, financed by UEFISCDI.

REFERENCES

[1] F. Leon, B. I. Aignătoaiei, and A. D. Leca, “An Evolutionary Optimization Framework for Intelligent Agents”, Proceedings of the 15th International Conference on System Theory, Control and Computing (Joint Conference SINTES 15, SACCS 11, SIMSIS 15), 306-311, 2011.

[2] B. V. Babu and R. Angira, “Modified differential evolution (MDE) for optimization of non-linear chemical processes”, Comp. Chem. Eng., vol. 30, pp. 989–1002, 2006.

[3] S. Curteanu and F. Leon, “Optimization Strategy Based on Genetic Algorithms and Neural Networks Applied to a Polymerization Process”, International Journal of Quantum Chemistry, vol. 108, pp. 617–630, Wiley Periodicals, USA, 2008.

[4] M. Keijzer, J. J. Merelo, G. Romero, and M. Schoenauer, “Evolving objects: A general purpose evolutionary computation library”, Artificial Evolution, vol. 2310, pp. 829–888, 2002.

[5] Evolving Objects (EO): an Evolutionary Computation Framework, http://eodev.sourceforge.net, 2012 [retrieved: June, 2021].

[6] Genetic Algorithms Framework, http://sourceforge.net/projects/ga-fwork, 2013 [retrieved: June, 2021].

[7] Watchmaker Framework for Evolutionary Computation, http://watchmaker.uncommons.org, 2010 [retrieved: June, 2021].

[8] AForge.NET, “Genetic Algorithms Library”, http://www.aforgenet.com/framework/features/genetic_algorithms.html, 2013 [retrieved: June, 2021].

[9] The SciPy community, “Optimization (scipy.optimize)”, https://docs.scipy.org/doc/scipy/reference/tutorial/optimize.html, 2021 [retrieved: June, 2021].

[10] M. Berkelaar, K. Eikland, and P. Notebaert, “lpsolve, Mixed Integer Linear Programming (MILP) solver”, https://sourceforge.net/projects/lpsolve, 2021 [retrieved: June, 2021].

[11] MathWorks, “Optimization Toolbox”, https://www.mathworks.com/products/optimization.html, 2021 [retrieved: June, 2021].

[12] S. Luke et al., “ECJ 27, A Java-based Evolutionary Computation Research System”, https://cs.gmu.edu/~eclab/projects/ecj, 2019 [retrieved: June, 2021].

[13] A. J. Nebro, “jMetal, Metaheuristic Algorithms in Java”, http://jmetal.github.io/jMetal, 2020 [retrieved: June, 2021].

[14] F. A. Fortin, F. M. De Rainville, M. A. Gardner, M. Parizeau, and C. Gagné, “DEAP: Evolutionary Algorithms Made Easy”, Journal of Machine Learning Research, vol. 13, pp. 2171–2175, 2012.

[15] S. Wagner et al., “HeuristicLab, A Paradigm-Independent and Extensible Environment for Heuristic Optimization”, https://dev.heuristiclab.com/trac.fcgi/wiki, 2019 [retrieved: June, 2021].

[16] R. Oldenhuis, “Test functions for global optimization algorithms”, https://github.com/rodyo/FEX-testfunctions/releases/tag/v1.5, 2021 [retrieved: June, 2021].

[17] L. Wang, H. Ni, R. Yang, M. Fei, and W. A. Ye, “Simple Human Learning Optimization Algorithm”, Communications in Computer and Information Science book series CCIS, Springer Berlin, vol. 462, pp. 56–65, 2014.

[18] Z. Z. Liu, D. H. Chu, C. Song, X. Xue, and B. Y. Lu, “Social learning optimization (SLO) algorithm paradigm and its application in QoS-aware cloud service composition”, Information Sciences, vol. 326, pp. 315–333, DOI: 10.1016/j.ins.2015.08.004, 2016.

[19] R. V. Rao, V. J. Savsani, and D. P. Vakharia, “Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems”, Computer-Aided Design, vol. 43, pp. 303–315, 2011.

[20] E. Fadakar and M. Ebrahimi, “A new metaheuristic football game inspired algorithm”, 1st Conference on Swarm Intelligence and Evolutionary Computation, CSIEC 2016 Proceedings, 6-11, 2016.

[21] A. V. Djunaidi and C. P. Juwono, “Football game algorithm implementation on the capacitated vehicle routing problems”,

International Journal of Computing Algorithm, vol. 7.1, pp. 45–53, 2018.

[22] R. Moghdani and K. Salimifard, "Volleyball Premier League Algorithm", Applied Soft Computing, vol. 64, pp. 161–185, 2018.

[23] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition", Evolutionary computation, CEC IEEE Congress, 4661-4667, 2007.

[24] H. Shabani, B. Vahidi, and M. Ebrahimpour, "A robust PID controller based on imperialist competitive algorithm for load-frequency control of power systems", ISA Transactions, vol. 52, no. 1, pp. 88–95, 2013.

[25] S. Hosseni and A. Khaled, "A survey on the imperialist competitive algorithm metaheuristics: implementation in Engineering domain and directions for future research", Applied Soft Computing, vol. 24, pp. 1078–1094, 2014.

[26] P. Cortés, J. M. García, J. Muñuzuri, and L. Onieva, "Viral systems: A new bio-inspired optimisation approach", Computers & Operations Research, vol. 35, pp. 2840–2860, 2008.

[27] M. Jaderyan and H. Khotanlou, "Virulence Optimization Algorithm", Applied Soft Computing, vol. 43, pp. 596–618, 2016.

[28] M. D. Li, H. Zhao, X. W. Weng, and T. Han, "A novel nature-inspired algorithm for optimization: Virus colony search", Advances in Engineering Software, vol. 92, pp. 65–88, 2016.

[29] S. J. D. Hosseini, M. Moradian, H. Shahinzadeh, and S. Ahmadi, "Optimal Placement of Distributed Generators with Regard to Reliability Assessment using Virus Colony Search Algorithm", International Journal of Renewable Energy Research, vol. 8, 2018.

[30] H. Shahinzadeh, G. B. Gharehpetian, M. Moazzami, J. Moradi, and S. H. Hosseinia, "Unit commitment in smart grids with wind farms using virus colony search algorithm and considering adopted bidding strategy", Proceedings of Smart Grid Conference (SGC), 1-9, 2017.

[31] Y. C. Liang and J. R. Cuevas Juarez, "Multilevel image thresholding using relative entropy and virus optimization algorithm", IEEE Congress on Evolutionary Computation, 2012.

[32] Y. C. Liang and J. R. Cuevas Juarez, "A novel metaheuristic for continuous optimization problems: Virus optimization algorithm", Engineering Optimization, vol. 48, pp. 73–93, 2015.

[33] Y. C. Liang and J. R. Cuevas Juarez, "Harmony search and virus optimization algorithm for multi-objective combined economic energy dispatching problems", IEEE Congress on Evolutionary Computation, 3947-3954, 2016.

[34] B. Peng, "Knowledge and population swarms in cultural algorithms for dynamic environments", Wayne State University, 2005.

[35] L. Wang et al., "An adaptive simplified human learning optimization algorithm", Information Sciences, vol. 320, pp. 126–139, 2015.

[36] J. Cao, Z. Yan, X. Xu, G. He, and S. Huang, "Optimal power flow calculation in AC/DC hybrid power system based on adaptive simplified human learning optimization algorithm", Journal of Modern Power Systems and Clean Energy, vol. 4, pp. 690–701, 2016.

[37] A. Naik, A., and S. C. Satapathy, "A comparative study of social group optimization with a few recent optimization algorithms", Complex Intell. Syst. vol. 7, pp. 249–295, DOI: 10.1007/s40747-020-00189-6, 2021.

[38] R. Venkata Rao, "Review of applications of TLBO algorithm and a tutorial for beginners to solve the unconstrained and constrained optimization problems", Decision Science Letter, vol. 5, pp. 1–30, 2016.

[39] Z. Zhai, G. Jia, and K. Wang, "A novel Teaching-Learning-Based Optimization with Error Correction and Cauchy Distribution for Path Planning of Unmanned Air Vehicle", Computational Intelligence and Neuroscience, pp. 1–12, DOI: 10.1155/2018/5671709, 2018.

[40] Y. Kumar, N. Dahiya, S. Malik, and S. Khatri, "A new variant of teaching learning based optimization algorithm for global optimization problems", Informatica, vol. 43, no. 1, 2019.

[41] Q. Zhang, G. Yu, and H. Song, "A hybrid bird mating optimizer algorithm with teaching-learning-based optimization for global numerical optimization", Statistics, Optimization & Information Computing, vol. 3, pp. 54–65, 2015.

[42] F. Zou, L. Wang, D. Chen, and X. Hei, "An Improved Teaching-Learning-Based Optimization with Differential Learning and Its Application", Mathematical Problems in Engineering, pp. 1–19, 2015.

[43] R. Moghdani, M. Abd Elaziz, D. Mohammadi, and N. Neggaz, "An improved volleyball premier league algorithm based on sine cosine algorithm for global optimization problem", Engineering with Computers, pp. 1–30, 2020.

[44] R. Moghdani, K. Salimifard, E. Demir, and A. Benyetton, "Multi-objective volleyball premier league algorithm", Knowledge-Based Systems, vol. 196, no. 21, 2020.

[45] S. Talatahari, B. Farahmand Azar, R. Sheikholeslami, and A. H. Gandomi, "Imperialist competitive algorithm combined with chaos for global optimization", Communications in Nonlinear Science and Numerical Simulation, vol. 17, no. 3, pp. 1312–1319, 2012.

[46] T. Niknam, E. T. Fard, N. Pourjafarian, and A. Rousta, "An efficient hybrid algorithm based on modified imperialist competitive algorithm and K-means for data clustering", Engineering Applications of Artificial Intelligence, 24, 2, pp. 306–317, 2011.

[47] M. A. Ahmadi, M. Ebadi, A. Shokrollahi, and J. M. S. Majidi, "Evolving artificial neural network and imperialist competitive algorithm for prediction oil flow rate of the reservoir", Applied Soft Computing, vol. 13, no. 2, pp. 1085–1098, 2013.

[48] M. Hajihassani, D. J. Armaghani, A. Marto, and E. T. Mohamad, "Ground vibration prediction in quarry blasting through an artificial neural network optimized by imperialist competitive algorithm", Bulletin of Engineering Geology and the Environment, vol. 4, pp. 873–886, 2015.

[49] C. Lu, X. Li, L. Gao, W. Liao, and J. Yi, "An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times", Computers & Industrial Engineering, vol. 104, pp. 156–174, 2017.

 ISBN: 978-1-61208-882-2