Software Impacts 12 (2022) 100274

Contents lists available at ScienceDirect

Software Impacts

journal homepage: www.journals.elsevier.com/software-impacts

Original software publication

Battle Royale Optimizer for solving binary optimization problems

Taymaz Akan [a,b], Saeid Agahian [c], Rahim Dehkharghani [d,*]

[a] Istanbul Ayvansaray University, Istanbul, Turkey
[b] Faculty of Electrical Engineering and Informatics, University of Pardubice, Pardubice, Czech Republic
[c] Erzurum Technical University, Erzurum, Turkey
[d] Department of Computer Engineering, Işik University, Istanbul, Turkey

ARTICLE INFO

Keywords:
Optimization
Discrete optimization
Battle Royale Optimization
Binary Battle Royale Optimization

ABSTRACT

Battle Royale Optimizer (BRO) is a recently proposed metaheuristic optimization algorithm used only in continuous problem spaces. The BinBRO is a binary version of BRO. The BinBRO algorithm employs a differential expression, which utilizes a dissimilarity measure between binary vectors instead of a vector subtraction operator, used in the original BRO algorithm to find the nearest neighbor. To evaluate BinBRO, we applied it to two popular benchmark datasets: the uncapacitated facility location problem (UFLP) and the maximum-cut (Max-Cut) graph problems from OR-Library. An open-source MATLAB implementation of BinBRO is available on CodeOcean and GitHub websites.

Code metadata

Current code version	V1.0
Permanent link to code/repository used for this code version	https://github.com/SoftwareImpacts/SIMPAC-2022-12
Permanent link to Reproducible Capsule	https://codeocean.com/capsule/8988211/tree/v1
Legal Code License	MIT license
Code versioning system used	Created with MATLAB R2020b
Software code languages, tools, and services used	MATLAB
Compilation requirements, operating environments & dependencies	
If available Link to developer documentation/manual	https://github.com/tami64/BinBRO_Max_UFLP/
Support email for questions	taymazakan@ayvansaray.edu.tr saeid.agahian@erzurum.edu.tr rahim.dehkharghani@isikun.edu.tr

1. Introduction

The process of choosing the best potential solution out of a set of candidate solutions for a given problem is known as optimization [1]; Deterministic or stochastic methods can make this choice. Deterministic approaches navigate the whole state space of a problem and find the optimum solution; however, these methods would be prohibitively expensive for problems with non-polynomial (NP) nature. Stochastic methods are an alternative for dealing with this problem. Many real-world problems in both scientific and industrial fields can be solved by optimization techniques. Metaheuristic algorithms have been frequently used to tackle optimization problems for the past two decades [2]. Evolutionary Algorithms (EA), physical phenomena algorithms, and Swarm Intelligence (SI) are the three main stochastic metaheuristics optimization tools.

Darwin's theory of evolution inspired evolutionary optimization algorithms, which share a typical process of mutation, selection, and recombination operations [3–7]. The physical laws such as electromagnetic force, gravitational force, and inertia force are emulated by

The code (and data) in this article has been certified as Reproducible by Code Ocean: (https://codeocean.com/). More information on the Reproducibility Badge Initiative is available at https://www.elsevier.com/physical-sciences-and-engineering/computer-science/journals.

* Correspondence to: Department of Computer Engineering, Işik University, Istanbul, 34980, Turkey.
E-mail addresses: taymazakan@topkapi.edu.tr (T. Akan), saeid.agahian@erzurum.edu.tr (S. Agahian), rahim.dehkharghani@isikun.edu.tr (R. Dehkharghani).

https://doi.org/10.1016/j.simpa.2022.100274
Received 20 February 2022; Received in revised form 12 March 2022; Accepted 24 March 2022

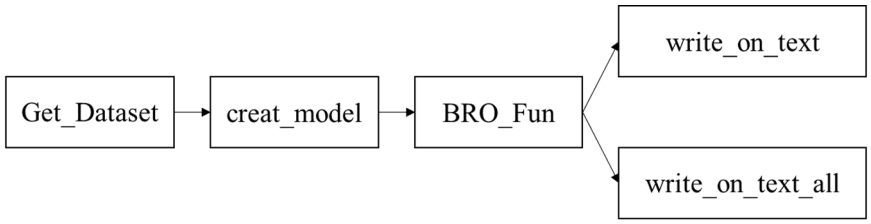

Fig. 1. The flow diagram of the software program for the BinBRO pipeline.

physical phenomena algorithms. These laws perform individuals' movements and interactions [8–16]. Finally, the group activity of diverse living beings, such as animals and insects, has inspired the invention of Swarm Intelligence algorithms. The BRO algorithm is not only a population-based algorithm; its creation in 2020 has added a new category to optimization algorithms named game-based optimization algorithms [17]. In game-based algorithms, players explore their surroundings and compete with their neighboring players as potential solutions. Unlike SI-based algorithms, players do not collaborate to achieve a common goal; instead, they attempt to defeat all opponents and win the game individually. In this work, an open-source MATLAB implementation of the BinBRO applied to the uncapacitated facility location problem (UFLP) and the maximum-cut graph (Max-cut) binary problems have been explained in detail. The complete source code and the dataset, results, and diagrams have been recently published in CodeOcean and GitHub [18,19].

2. The BinBRO algorithm

The BinBRO approach, similar to other optimization algorithms, randomly initializes n Candidate Solutions (CSs) using a uniform distribution, as though players are randomly dispersed over the game field from scratch. In the uniform distribution, if the generated random value is smaller than 0.5, a given facility will be initialized by 0 or 1 otherwise. The number of facilities is equal to the number of solutions. Each solution is a binary vector with d dimensions, $x\_i = (x\_i1, x\_i2, \dots, x\_id)$. This vector can be interpreted differently for different problems; for example, in the UFLP, x_{ij} corresponds to the j_{th} facility (0 as closed and 1 as open) of the i_{th} solution. The procedure is then repeated for several iterations. Each CS is compared to its nearest neighbor in each iteration. The nearest neighbor is determined using Jaccard's similarity coefficient [20]. When a CS is compared to its nearest neighbor, the solution with a greater fitness value -generated from the fitness function- would be named as the winner and the other as the loser. The BinBRO algorithm employs various strategies to handle both the winner and the loser. The algorithm always applies a mutation task to the winner–as in the battle royale game. The position of the current winner player will be changed to help him become the final winner. Only two randomly chosen bits of the winner solution would be flipped in the mutation process for this aim. The BinBRO algorithm [21], on the other hand, performs a crossover task on the loser solution if it has not achieved the maximum number of losses (if it has not died yet). The "maximum number of losses" is a BinBRO parameter that must be defined in advance. This crossover has been utilized in three configurations: single-point, two-points, and uniform. If the loser achieves the maximum number of losses or dies in the game, it will be re-spawned after completing a primary mutation task. The deceased player will be re-spawned in another legal location on the field, as in the Battle Royale game. This basic mutation depends on the iteration, i.e., the intensity of mutation at the beginning is high, but it decreases while the game field is shrinking. The BinBRO algorithm flips the loser (dead) solution's bits from 1 to 0 or vice versa. The intuition behind this is that, like the Battle Royale game, BinBRO constrains the solutions to converge to the global optimum by reducing the space in which the solutions compete with one another. This procedure is repeated until an acceptable solution is found or the "maximum number of iterations" criteria are met. Note that a 2-bit mutation is applied to the best solution after each solution update of the loser or the winner solution found so far. If a superior fitness value can be achieved after this mutation, the optimal solution will be modified; otherwise, it will remain unchanged.

3. Software description

The BinBRO algorithm has been implemented in MATLAB. The approach has been applied to two well-known benchmark datasets: uncapacitated facility location and maximum-cut graph binary optimization problems. The flow diagram of a BinBRO pipeline is illustrated in Fig. 1. The program's key modules are listed as follows.

- The *Get_Dataset* function returns the file address and the desired optimal value for the i_{th} instance.
- The *creat_model* function returns the structure of the i_{th} instance.
- The *BRO_Fun* function applies the BinBRO algorithm to the i_{th} instance and returns the structure and convergence curve. The structure is a group of items of performance evaluation criteria.
- The *write_on_text* function generates a text file for the i_{th} instance and stores all performance criteria in this text file after a predetermined number of algorithms runs.
- The *write_on_text_all* function generates a text file for all instances and stores all performance criteria in it after a predetermined number of algorithm runs.

4. Impact

The BinBRO algorithm has been proposed as one of the most recent optimization algorithms [21]; It is a reliable technique for solving binary optimization problems. This algorithm has been applied to two benchmark datasets, explained in the following subsections. However, to utilize this software for solving other binary optimization problems, the MATLAB function file (of a given objective function) should be added to a folder that includes "main_all.m", and the name of this MATLAB file should be assigned to the Eval variable, e.g.", Eval =@MaxCut;", in the same folder.

4.1. Uncapacitated facility location problem

In the uncapacitated facility location problem (UFLP), the goal is to identify which facility site should be open out of a predetermined set of facility sites and which one should be closed. The facilities are supposed to serve a predetermined set of clients who would receive service from their nearest facility. We are interested in minimizing the overall cost of this service. This cost includes the cost of opening a facility plus the distance of a client from a facility. In the UFLP problem, also known as the "simple" facility location problem, SFLP, clients and the facilities are discrete points on the plane; therefore, the continuous optimization algorithms cannot solve this problem.

Consequently, instead of using the continuous version of the Battle Royale Algorithm, we used its binary version, the BinBRO algorithm, to solve binary optimization probles. The BinBRO algorithm has been

Table 1
Quantitative evaluation of results for three instances in the UFLP.

Instances	Crossover	Mean	Best	Worst	STD	Hit	Gap	NFE	Time
Cap-71	single-point	932615.8	932615.8	932615.8	0	1	0	430.36	0.05
	two-point	932615.8	932615.8	932615.8	0	1	0	851.64	0.11
	Uniform	932615.8	**932615.8**	932615.8	0	1	0	**371.44**	0.6
Cap-101	single-point	796786.08	796648.4	797508.73	321.89	0.84	0.02	28631.96	11.37
	two-point	796751.67	796648.4	797508.73	285.325	0.88	0.01	30040.72	4.95
	Uniform	**796648.4**	796648.4	**796648.4**	0	**1**	**0**	**11010.16**	3.33
Cap-131	single-point	793749.27	793439.6	794299.9	421.453	0.64	0.04	73540.2	22.01
	two-point	793680.44	793439.6	794299.9	394.233	0.72	0.03	64445.68	22.12
	Uniform	**793508.4**	793439.6	794299.9	238.203	**0.92**	**0.01**	**44500.28**	**15.90**

Table 2
Quantitative evaluation of results for three instances in the UFLP.

Instances	Crossover	Mean	Best	Worst	STD	NFE	Time
pw01- 100.0	single-point	1948.1	1988	1901	34.959	57300.9	2181.9
	two-point	**1973.6**	1986	**1940**	**13.243**	57347.9	2163.45
	Uniform	1954.7	**2000**	1918	21.94	**57270.6**	**2099.07**
pw05- 100.0	single-point	8076.3	**8166**	7973	54.805	57271.2	2484.92
	two-point	8074.9	8164	**8013**	**42.459**	57356.20	2193.63
	Uniform	**8085.3**	8160	7986	48.792	**57271.4**	**2159.05**
pw09- 100.0	single-point	13485.7	13567	13367	67.335	57307.7	1971.88
	two-point	**13494.9**	13558	13398	51.898	57304.2	2080.49
	Uniform	13477.1	13551	13411	42.354	57241.9	2936.37

tested on the standard, publicly accessible UFLP dataset from OR-Lib [22]. Table 1 shows the numerical results of BinBRO with different types of crossover (single-point, two-point, and uniform) for various performance measures obtained from some benchmark suites. The obtained results approve that the Bin-BRO algorithm can yield promising results for the UFLP. We provide the obtained results for only a subset of the benchmark datasets. In this table, the best value in each row is bold. The first column includes the name of benchmark suits taken from OR-Lib; the second column indicates the crossover type, and the other columns show the obtained values for each performance criterion.

4.2. The max-cut problem

A maximum cut in a graph is a cut that is at least the size of any other cut. In other words, it is a division of the graph's vertices into two complementary sets, S and T, with as many edges as possible between both sets. The issue can be expressed as follows: the goal is to find a subset S of the vertex set with as many edges as possible between itself and the complementary subset. Alternatively, a bipartite subgraph with as many edges as possible is desired. The BinBRO has also been tested over the standard publicly accessible Max-Cut dataset from OR-Lib [22]. Table 2 shows the numerical results obtained by the BinBRO with different crossover types for various performance measures obtained from some benchmark suites. We provide the obtained results for only a subset of the benchmark datasets. The obtained results demonstrate that the Bin-BRO can produce promising results for Max-Cut. In this table, the best value in each row is bold. The first column includes the name of benchmark suits taken from OR-Lib; the second column indicates the crossover type, and the other columns show the obtained values for each performance criterion.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R.D. Norton, P.B. Hazell, Mathematical Programming for Economic Analysis in Agriculture, Macmillan, 1986.
[2] A. Lazar, Heuristic Knowledge Discovery for Archaeological Data using Genetic Algorithms and Rough Sets Heuristic and Optimization for Knowledge Discovery, IGI Global, 2002, pp. 263–278.
[3] J. Holland, Adaptation in natural and artificial systems: An introductory analysis with application to biology, Control Artif. Intell. (1975).
[4] H.-P. Schwefel, Evolution strategies: A family of non-linear optimization techniques based on imitating some principles of organic evolution, Ann. Oper. Res. 1 (2) (1984) 165–167.
[5] F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (5) (1986) 533–549, http://dx.doi.org/10.1016/0305-0548(86)90048-1.
[6] P.J. Van Laarhoven, E.H. Aarts, Simulated Annealing Simulated Annealing: Theory and Applications, Springer, 1987, pp. 7–15.
[7] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (6) (2008) 702–713, http://dx.doi.org/10.1109/TEVC.2008.919004.
[8] R.A. Formato, Central force optimization, Prog. Electromagn. Res. 77 (2007) 425–491.
[9] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, Gsa: A gravitational search algorithm, Inform. Sci. 179 (13) (2009) 2232–2248, http://dx.doi.org/10.1016/j.ins.2009.03.004.
[10] A. Husseinzadeh Kashan, A new metaheuristic for optimization: Optics inspired optimization (oio), Comput. Oper. Res. 55 (2015) 99–125, http://dx.doi.org/10.1016/j.cor.2014.10.011.
[11] A. Kaveh, T. Bakhshpoori, Water evaporation optimization: A novel physically inspired optimization algorithm, Comput. Struct. 167 (2016) 69–85, http://dx.doi.org/10.1016/j.compstruc.2016.01.008.
[12] A. Hatamlou, Black hole: A new heuristic optimization approach for data clustering, Inform. Sci. 222 (2013) 175–184, http://dx.doi.org/10.1016/j.ins.2012.08.023.
[13] A. Kaveh, S. Talatahari, A novel heuristic optimization method: Charged system search, Acta Mech. 213 (3) (2010) 267–289, http://dx.doi.org/10.1007/s00707-009-0270-4.
[14] V. Punnathanam, P. Kotecha, Yin-yang-pair optimization: A novel lightweight optimization algorithm, Eng. Appl. Artif. Intell. 54 (2016) 62–79, http://dx.doi.org/10.1016/j.engappai.2016.04.004.
[15] A. Kaveh, A. Dadras, A novel meta-heuristic optimization algorithm: Thermal exchange optimization, Adv. Eng. Softw. 110 (2017) 69–84, http://dx.doi.org/10.1016/j.advengsoft.2017.03.014.
[16] H. Abedinpourshotorban, S. Mariyam Shamsuddin, Z. Beheshti, D.N.A. Jawawi, Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm, Swarm Evol. Comput. 26 (2016) 8–22, http://dx.doi.org/10.1016/j.swevo.2015.07.002.
[17] T. Rahkar Farshi, Battle royale optimization algorithm, Neural Comput. Appl. (2020) http://dx.doi.org/10.1007/s00521-020-05004-4.

[18] T. Akan, S. Agahian, R. Dehkharghani, Binbro: Binary battle royale optimizer algorithm [source code], 2022, http://dx.doi.org/10.24433/CO.3120150.v1.
[19] T. Akan, S. Agahian, R. Dehkharghani, Binbro_uflp, 2022, Retrieved from https://github.com/tami64/BinBRO_Max_UFLP.
[20] P.H. Sneath, Some thoughts on bacterial classification, Microbiology 17 (1) (1957) 184–200.
[21] T. Akan, S. Agahian, R. Dehkharghani, Binbro: Binary battle royale optimizer algorithm, Expert Syst. Appl. (2022) http://dx.doi.org/10.1016/j.eswa.2022.116599.
[22] Beasley J. E., Or-library: Distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (11) (1990) 1069–1072.