VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF MECHANICAL ENGINEERING

MM Science Journal | www.mmscience.eu
ISSN 1803-1269 (Print) | ISSN 1805-0476 (On-line)
Special Issue | TEAM 2024
Transdisciplinary and Emerging Approaches to Multidisciplinary Science
11.9.2024 – 13.9.2024, Ostrava, Czech Republic
DOI: 10.17973/MMSJ.2024_12_2024125

MM SCIENCE JOURNAL

TEAM2024-00038

GAUSSIAN TRANSFER FUNCTIONS BASED BINARY PARTICLE SWARM OPTIMIZATION FOR ENHANCED PERFORMANCE IN UN-CAPACITATED FACILITY LOCATION PROBLEM

Kanak Kalita [1,2,*], Lenka Cepova [3], Pradeep Jangir [4,5,6,7]

[1] Department of Mechanical Engineering, Vel Tech Rangarajan Dr, Sagunthala R&D Institute of Science and Technology, Avadi 600062, India. drkanakkalita@veltech.edu.in.

[2] Jadara Research Center, Jadara University, Irbid 21110, Jordan.

[3] Department of Machining, Assembly and Engineering Metrology, Faculty of Mechanical Engineering, VSB-Technical University of Ostrava, 70800 Ostrava, Czech Republic. lenka.cepova@vsb.cz.

[4] Department of Biosciences, Saveetha School of Engineering. Saveetha Institute of Medical and Technical Sciences, Chennai 602105, India. pkjmtech@gmail.com.

[5] University Centre for Research and Development, Chandigarh University, Mohali 140413, India.

[6] Department of CSE, Graphic Era Hill University. Graphic Era Deemed to Be University, Dehradun 248002, India.

[7] Applied Science Research Center, Applied Science Private University, Amman 11931, Jordan.

*Corresponding author; e-mail: drkanakkalita@veltech.edu.in

Abstract

This study introduces Gaussian Binary Particle Swarm Optimization (G-BPSO), designed to address binary optimization challenges effectively. G-BPSO employs new transfer functions of the Gaussian type derived from the power functions to enable mapping of real-valued vectors of individual encodings into binary form. This ensures smooth change between steps and improved convergence. To assess the effectiveness of G-BPSO, a host of complex optimization problems such as the un-capacitated facility location problem are investigated. Enhanced efficiency and improvement over existing methods in binary optimization is observed. The MATLAB code of G-BPSO is made open-access through https://github.com/kanak02/GBPSO.

Keywords:

Evolutionary algorithm, Particle swarm optimization, Gaussian shaped, Transfer function, Optimization, Discrete optimization

1 INTRODUCTION

Binary integer optimization challenges are a type of combinatorial optimization problems namely binary optimization problems (BOPs) which includes problems like 0-1 knapsack problem (0–1KP) [Abdel-Basset 2024], un-capacitated facility location problem (UFLP) [Ozsoydan 2024], maximum coverage problem (MCP) [Baldomero-Naranjo 2024], feature selection problem (FSP) [Premalatha 2024], software and hardware partitioning problem (HW/SW) [Deng 2024]. These problems have wide applications ranging from information technology, economic management, industrial engineering to telecommunications [Blekos 2024]. It is important to note that for BOPs, the solution is bounded to binary value only i.e., 0 or 1, giving a function solution of $\{0,1\}^n$. This limitation reduces the applicability of traditional deterministic algorithms for large-scale BOPs and thus, has motivated researchers to turn to stochastic algorithms. Among such stochastic algorithms, evolutionary algorithms (EAs) are perhaps the most favored ones [Abdel-Basset 2024].

Particle Swarm Optimization (PSO) is an extremely popular EA that mimics the characteristics of bird swarms [Eberhart 1995]. Its simple design and easy computation are some of the reasons why people have embraced it so readily. Most discrete optimization problems have binary search spaces and hence requires the use of binary-based algorithms for solving them. For instance, to solve Binary HS (BHS) problem, HS basic principles and pitch adjustment rules were used [Ling 2010a]; Ling et al. [Ling 2010b] employed a probability estimation operator in doing modification on the DE for discrete problems. Similarly, new algorithms such as Binary MOA [Mirjalili 2012] and Binary GSA [Rashedi 2009] have been created, utilizing transfer functions and updated positioning rules to maintain the characteristics of continuous algorithms.

Kennedy and Eberhart proposed the binary version of PSO known as BPSO in 1997 [Kennedy 1997], which had different features from that of continuous PSO including a new transfer function and a modification in the method of updating the positions of different particles. This adaptation assists in mapping the continuous search space to binary states and shifts the position of particles in binary arenas. It was also reported in [Luh 2011] that the initial BPSO has some limitations such as local optima entrapment and they further modified it to optimize its performance. To apply these algorithms for BOPs, there are techniques developed to discretize EAs by use of transfer functions [Mirjalili 2013]. Transfer functions can be divided into two main groups—S-shaped and V-shaped [Mafarja 2017]. These functions help in converting a real vector to a binary vector that can be used for discretizing EAs to work on BOPs.

In this paper, novel Gaussian-shaped transfer functions are proposed and analyzed to improve the convergence rates of the algorithm and to prevent it from being trapped in local minima. Using the proposed transfer functions, a novel binary PSO variant called G-BPSO is introduced. Further, a comprehensive exposition of the open-source MATLAB version of G-BPSO for UFLP binary problems is given. The original and in-house developed source code of the study along with the datasets used in the analysis, the possible results and diagrams are made available through GitHub.

2 PROPOSED GAUSSIAN-SHAPED TRANSFER FUNCTIONS

A transfer function is instrumental in evaluating the probability of transitioning between the values of the position vector elements, from 0 to 1 and vice versa in a binary environment [Rashedi 2009]. This mechanism forces the particles to move within these restrictions. The characteristics of the transfer functions are—

The output of a transfer function must be within the interval [0,1] meaning the probability of a particle changing positions.

The transfer functions should therefore give a higher probability of a position change for particles having large absolute velocities as this indicates that the particles are far from the optimum solution in the current iteration and thus, change of position is needed in the next iteration.

On the other hand, a low value of absolute velocity should be associated with low probability of position change.

The probability indicated by a transfer function should increase with the velocity, suggesting that particles moving away from the optimal solution are more likely to adjust their position vectors to revert to favorable states.

Similarly, the probability should decrease as the velocity lessens, aligning movement closer to the optimal trajectory.

The Gaussian transfer function employed in the traditional BPSO [Mafarja 2017] is exemplified by Equation (1) and shown in Fig. 1. Analysis of Fig. 1 reveals the introduction of four novel transfer functions.

$$G1(x) = e^{-(x)^2};\; G2(x) = e^{-(x/2)^2};\; G3(x) = e^{-(x/3)^2};\; G4(x) = e^{-(x/4)^2} \quad (1)$$

Due to their characteristic curves, these are termed Gaussian-shaped transfer functions and collectively, they are referred to as the G-shaped family of transfer functions. Position updating rules for this family prompts particles to adopt either a value of 0 or 1, reinforcing the binary nature of their operation.

3 GAUSSIAN-SHAPED BINARY PARTICLE SWARM (G-BPSO) ALGORITHM

PSO is an evolutionary computation method originally developed by Kennedy and Eberhart Eberhart 1995]. This technique draws inspiration from the social behavior observed in bird flocking. It utilizes a collection of particles (candidate solutions) that navigate the search space to identify the optimal solution. As they move, these particles track the most promising location (best solution) encountered along their paths. Essentially, each particle evaluates its own best-found solution and also considers the best solution achieved by the entire swarm. In PSO, every particle must account for its present position, its current velocity, its proximity to its personal best solution ($pbest$) and its distance to the swarm best solution ($gbest$) to adjust its trajectory. The mathematical formulation of PSO is outlined as follows:

$$v_i^{t+1} = wv_i^t + c_1 \times rand \times (pbest_i - x_i^t) + c_2 \times rand \times (gbest - x_i^t) \quad (2)$$

$$x_i^{t+1} = x_i^t + v_i^{t+1} \quad (3)$$

In equations (2) and (3), v_i^t represents the velocity of particle i at iteration t, where w is a weight function, c_1 and c_2 are acceleration coefficient and $rand$ is a randomly generated number between 0 and 1. The term x_i^t denotes the current position of particle i at iteration t, $pbest_i$ refers to the best solution that particle i has achieved to date and $gbest$ signifies the optimal solution discovered by the swarm up to that point.

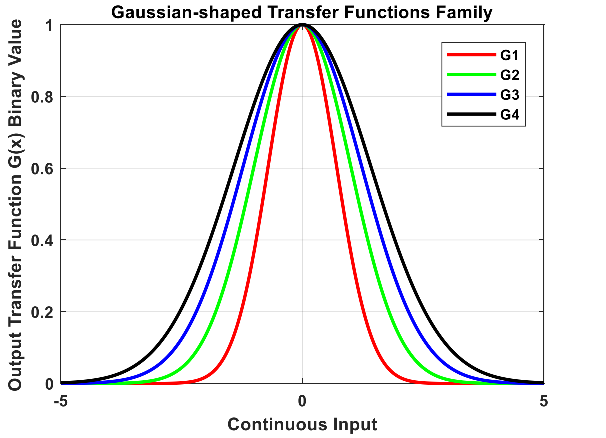


Fig. 1: G-shaped family of transfer functions.

In the continuous variant of PSO, particles navigate the search space using position vectors within the real-valued domain. This allows for straightforward position updating, where velocities are simply added to positions as shown in Equation (3). Within this binary context, where only the values 0 and 1 are possible, traditional position updating using Equation (3) is not applicable. Consequently, an alternative method must be developed to employ velocities in toggling agents' positions between 0 and 1. Essentially, a connection between velocity and position must be established, necessitating a revision of the position updating process defined in Equation (3). In binary spaces, updating a position fundamentally involves switching between the 0 and 1 states, guided by the velocities of the agents.

The idea is somewhat entrenched in changing the position of an agent according to the probability deduced by its velocity. To achieve this, a transfer function is used which transforms velocity measures into probabilities determining position changes. Originally, the binary version of PSO (BPSO) was presented by Kennedy and Eberhart [Kennedy 1997] as an extension of the basic PSO for binary-valued problem spaces. However, the original BPSO also has the problem of easily falling into the local optimum and there are many improved models to get over this drawback.

In this adaptation, particles are restricted to a pure search space with position vectors that must be set to 0s and 1s only. The use of the velocities is to abstract the probability of a bit taking 0 or 1 states. Equation (4) is a Gaussian function that is employed as the transfer function in order to normalize all the real-valued velocities between 0 and 1 as probabilities.

$$G(v_{i,j}(t)) = e^{-(v_{i,j})^2}(t) \quad (4)$$

This indicates the velocity of particle i at iteration t in the j-th dimension. Once velocities are transformed into probability values, the position vectors are then updated based on these probabilities, as per Equation (5).

$$x_{i,j}(t+1) = \begin{cases} 0 & \text{If } rand < G(v_{i,j}(t+1)) \\ 1 & \text{If } rand \geq G(v_{i,j}(t+1)) \end{cases} \quad (5)$$

4 G-BPSO FOR SOLVING UFLP

4.1 Definition and Mathematical Model of UFLP

The UFLP, originally formulated by Kuehn and Hamburger [Kuehn 1963], is identified as one of the significant NP-hard problems in location theory complexities [Alultan 1999]. UFLP is crucial in many aspects which include warehouse design and operations, designing a network for a supply chain, logistics and transportation and planning the location of public facilities [Ghaderi 2013]. The problem is defined as follows—

A set of customers $K = \{k_1, k_2, \ldots, k_m\}$ exists, where m is the number of customers and k_i represents the i-th customer. There is also a set of potential facilities $S = \{s_1, s_2, \ldots, s_n\}$, where n denotes the number of facilities and s_j the j-th facility. An $m \times n$ service matrix $D = [d_{ij}]_{m\times n}$ details the service cost for servicing the i-th customer from the j-th facility. $G = \{g_1, g_2, \ldots, g_n\}$ outlines the fixed costs to open each facility. The objective of UFLP is to determine which facilities to open and how to allocate them to customers so as to minimize the combined costs of service and facility opening. The mathematical model in Equations (6-9), for UFLP can be expressed as a minimizing problem, i.e.,

Minimize

$$f(X,W) = \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij}w_{ij} + \sum_{j=1}^{n} g_j x_j \quad (6)$$

$$s.t. \sum_{j=1}^{n} w_{ij} = 1, i = 1,2,\ldots,m \quad (7)$$

$$w_{ij} \leq x_j, i = 1,2,\ldots,m, j = 1,2,\ldots,n \quad (8)$$

$$w_{ij}, x_j \in \{0,1\}, i = 1,2,\ldots,m, j = 1,2,\ldots,n \quad (9)$$

where $W = [w_{ij}]_{m\times n}$, $w_{ij} = 1$ if customer k_i is serviced by facility s_j and $w_{ij} = 0. x_j = 1$ if facility s_j is open. The UFLP instances can be divided into 4 classes by their size $n \times m$: the first is 16 instances with size 16×50, named Cap41~Cap44, Cap61~Cap64, Cap71~Cap74 and Cap81~Cap84.

4.2 Computational Experiments

All computations were performed on standard PCs within Microsoft Windows 10 and equipped with AMD Ryzen3 2200G processors running at 3.50 GHz and 8 GB RAM. All algorithms were developed in the MATLAB programming language within the Code environment. The performance of G-i-BPSO ($i = 1,2,3,4$) for solving UFLP is measured. This comparison aims to demonstrate the high competitiveness of G-BPSO in solving UFLP challenges. Each algorithm was run independently 10 times per UFLP instance with search agent 40 and maximum iterations 100.

4.3 Comparison and Analysis of Computational Results for UFLP

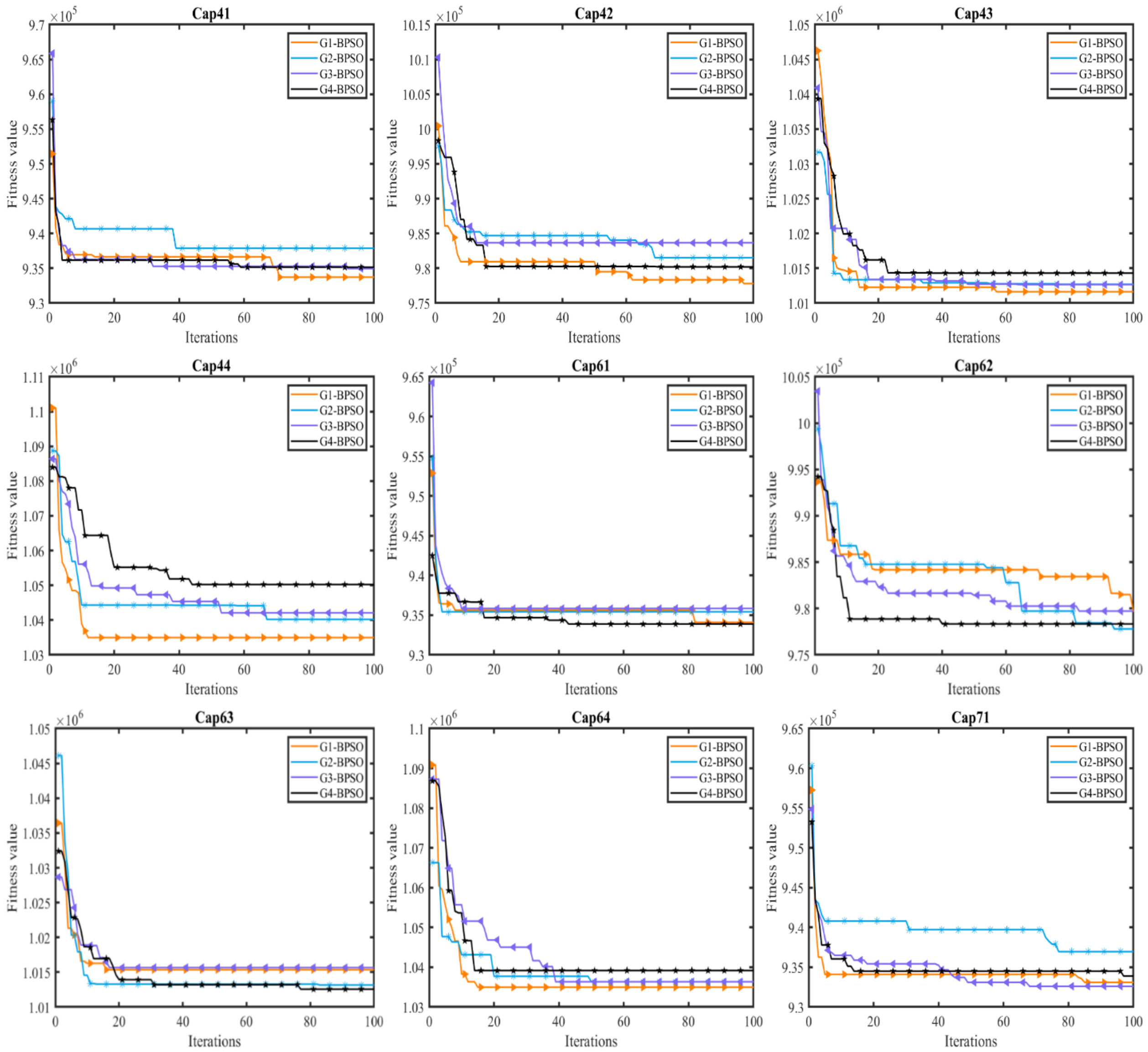
As observed from Tab. 1, the minimum average for the algorithm G1-BPSO is the smallest among other algorithms whenever this algorithm gives the best solution. Notably, G1-BPSO provides better results for Cap43, Cap44, Cap63, Cap64, Cap74, Cap81, Cap83 and Cap84. In fact, G1-BPSO obtains the minimum value of the objective functions in 8 of the 16 problems of the UFLP. These are Cap43, Cap44, Cap63, Cap64, Cap74, Cap81, Cap83 and Cap84. This proves that G1-BPSO has a good characteristic to converge to the minimum solution throughout all the iterations. The performance of the G4-BPSO is better in the other problems in some ways. Notably, G4-BPSO performs the best on Cap41, Cap62, Cap71 and Cap72. G4-BPSO reaches the minimum value in the lowest level in 4 of the 16 problems which confirms the efficiency of the algorithm in finding the optimal solutions. For problems Cap42 and Cap61 the minimum values are the lowest and both G1-BPSO and G4-BPSO algorithms give similar results. In contrast, G2-BPSO demonstrates its efficiency in Cap82 and reaches the minimum value. On the other hand, the G3-BPSO is the best for Cap74 because it can solve the best solution as seen in this problem. As can be seen, G1-BPSO has the smallest minimum in 50% of the problems (8 out of 16) proving that it is the most successful method in finding the exact solution. G4-BPSO accomplishes this 25% of the time (4 out of 16) and demonstrates its capacity but not its reliability in contrast to G1-BPSO. This again shows that G2-BPSO and G3-BPSO have lower reliability than G1-BPSO because each of them finds the lowest minimum value in one of the problems. This consistent performance of G1-BPSO indicates that the algorithm is quite stable in terms of its performance demonstrated by the convergence curve in Fig. 2.

The maximum values of G1-BPSO are the lowest among all the tested problems in most of the cases when it shows the highest performance. The G1-BPSO performs better for the Cap41, Cap42, Cap43, Cap44, Cap61, Cap64, Cap73, Cap81, Cap83 and Cap84. Namely, G1-BPSO yields the best result in terms of the maximum value in 10 out of the 16 considered UFLP problems. Some of these are Cap41, Cap42, Cap43, Cap44, Cap61, Cap64, Cap73, Cap81, Cap83 and Cap84. This consistent performance in attaining lower maximum values speaks volume on the reliability of G1-BPSO to hold stable solutions across iterations. For problems Cap62 and Cap82, G2-BPSO is better as it yields the lowest maxima. G2-BPSO does this in 2 out of the 16 problems, which shows that the proposed approach can provide optimal solutions to these problems. In cases of problems Cap63 and Cap74, G4-BPSO is better, as it yields the lowest maximum values. This is done in 2 of the 16 problems which proves that G4-BPSO has the capacity to arrive at the best solutions in these kinds of problems. In problem Cap71, G3-BPSO is the best as it minimizes the maximum value. In problem Cap72 G2-BPSO, G3-BPSO and G4-BPSO are at par with the other as they have the least maximum values. From the results displayed above, it is seen that the G1-BPSO has the lowest maximum value in 63% of the problems which signify that the algorithm is excellent in finding the optimal solutions. G2-BPSO succeeds in doing this in 13% of the problems (2 out of 16) proving its proficiency but at the same time its unpredictability concerning to G1-BPSO. G4-BPSO also succeeds in attaining this in 13% of the problems (2 out of 16). G3-BPSO accomplish this in 6% of the problems (1 out of 16). G1-BPSO has shown a consistent performance in all the runs thus indicating the ability of the algorithm to perform optimally in all the runs.

Tab. 1: Performance metrics of G-BPSO family algorithms for UFLP problems

Problems	Algorithm	Minimum	Maximum	Average	Std.	Time
	G1-BPSO	933568.9	933876.3	933722.6	217.3646245	117.1013296
Cap41	G2-BPSO	937268.8875	938514.65	937891.7688	880.8871115	77.6718501
	G3-BPSO	933568.9	936363.2	934966.05	1975.868479	84.4352743
	G4-BPSO	932615.75	937692.325	935154.0375	3589.680608	105.942596
	G1-BPSO	977799.4	977799.4	977799.4	0	110.5861844
Cap42	G2-BPSO	981538.85	981538.85	981538.85	0	110.2323077
	G3-BPSO	983549.675	983752.55	983651.1125	143.4542882	110.8551864
	G4-BPSO	977799.4	982615.75	980207.575	3405.673746	110.2475591
	G1-BPSO	1010641.45	1012643.688	1011642.569	1415.795714	110.7569178
Cap43	G2-BPSO	1011067.65	1014341.238	1012704.444	2314.77592	110.1741575
	G3-BPSO	1010808.163	1014491.4	1012649.781	2604.442213	111.4355116
	G4-BPSO	1013856.45	1014767.438	1014311.944	644.1654388	110.0220761
	G1-BPSO	1034976.975	1034976.975	1034976.975	0	127.4650759
Cap44	G2-BPSO	1037717.075	1042713.15	1040215.113	3532.758512	128.8655808
	G3-BPSO	1037717.075	1046508.088	1042112.581	6216.184552	124.2724103
	G4-BPSO	1044253.438	1056220.588	1050237.013	8462.052916	122.4181605
	G1-BPSO	933568.9	934622.575	934095.7375	745.0607377	120.906408
Cap61	G2-BPSO	932615.75	938222.0375	935418.8938	3964.243909	124.2333037
	G3-BPSO	933568.9	938110.625	935839.7625	3211.484546	119.2123679
	G4-BPSO	932615.75	935152.2875	933884.0188	1793.602867	125.7635494
	G1-BPSO	979099.6125	980176.5125	979638.0625	761.4832927	107.3647425
Cap62	G2-BPSO	977799.4	977799.4	977799.4	0	124.7331869
	G3-BPSO	977799.4	981649.35	979724.375	2722.325752	116.7499259
	G4-BPSO	977799.4	978876.3	978337.85	761.4832927	113.5375082
	G1-BPSO	1010641.45	1020091.513	1015366.481	6682.203276	137.4832544
Cap63	G2-BPSO	1010808.163	1015508.938	1013158.55	3323.949879	140.3122942
	G3-BPSO	1014097.288	1017249.375	1015673.331	2228.862446	118.5596177
	G4-BPSO	1012476.975	1012643.688	1012560.331	117.8835393	63.2874312
	G1-BPSO	1034976.975	1034976.975	1034976.975	0	131.2889561
Cap64	G2-BPSO	1034976.975	1037717.075	1036347.025	1937.543291	117.8053658
	G3-BPSO	1034976.975	1037717.075	1036347.025	1937.543291	121.7511571
	G4-BPSO	1037717.075	1040641.45	1039179.263	2067.845393	112.283952
	G1-BPSO	932615.75	933568.9	933092.325	673.9788285	130.918797
Cap71	G2-BPSO	936638.65	937268.8875	936953.7688	445.64521	126.979329
	G3-BPSO	932615.75	932615.75	932615.75	0	127.7031067
	G4-BPSO	933568.9	934199.1375	933884.0188	445.64521	127.125402
	G1-BPSO	977799.4	981538.85	979669.125	2644.190453	122.9883171
Cap72	G2-BPSO	977799.4	978876.3	978337.85	761.4832927	121.6306033
	G3-BPSO	977799.4	978876.3	978337.85	761.4832927	129.8552684
	G4-BPSO	977799.4	978876.3	978337.85	761.4832927	123.7230272
	G1-BPSO	1010641.45	1010641.45	1010641.45	0	109.4152445
Cap73	G2-BPSO	1010641.45	1012476.975	1011559.213	1297.912175	109.5062754
	G3-BPSO	1010808.163	1013932.9	1012370.531	2209.523076	109.7488977
	G4-BPSO	1012476.975	1013506.7	1012991.838	728.1255303	109.1736951
	G1-BPSO	1037717.075	1045383.788	1041550.431	5421.184398	109.3015241
Cap74	G2-BPSO	1037717.075	1048082.325	1042899.7	7329.338564	109.143237

	G3-BPSO	1034976.975	1052179.638	1043578.306	12164.11931	109.4911478
	G4-BPSO	1037717.075	1040641.45	1039179.263	2067.845393	108.6813492
	G1-BPSO	799695.025	800652.0375	800173.5313	676.7100284	146.7019072
Cap81	G2-BPSO	803698.625	805409.2625	804553.9438	1209.603376	148.8237855
	G3-BPSO	805859.1125	808847.0125	807353.0625	2112.764352	146.4893845
	G4-BPSO	805818.825	805880.05	805849.4375	43.29261268	146.3145998
	G1-BPSO	862562.8	864993.75	863778.275	1718.94123	147.6058885
Cap82	G2-BPSO	860960.2625	862260.475	861610.3688	919.3890757	146.9550613
	G3-BPSO	865602.325	866731.7	866167.0125	798.588721	147.7737961
	G4-BPSO	862663.075	869523.775	866093.425	4851.247494	146.6959447
	G1-BPSO	899460.975	904231.1625	901846.0688	3373.031929	147.3973123
Cap83	G2-BPSO	905642.5125	909550.9875	907596.75	2763.709177	148.702049
	G3-BPSO	904914.575	919410.6125	912162.5938	10250.24642	146.8517095
	G4-BPSO	906678.625	910813.1375	908745.8813	2923.541826	147.1013641
	G1-BPSO	944008.525	958857.3875	951432.9563	10499.73137	79.9797149
Cap84	G2-BPSO	953022.6375	975432.7875	964227.7125	15846.36903	79.1895906
	G3-BPSO	956402.3875	967562.1375	961982.2625	7891.134901	79.4995506
	G4-BPSO	967892.975	974525.75	971209.3625	4690.080181	79.3127924



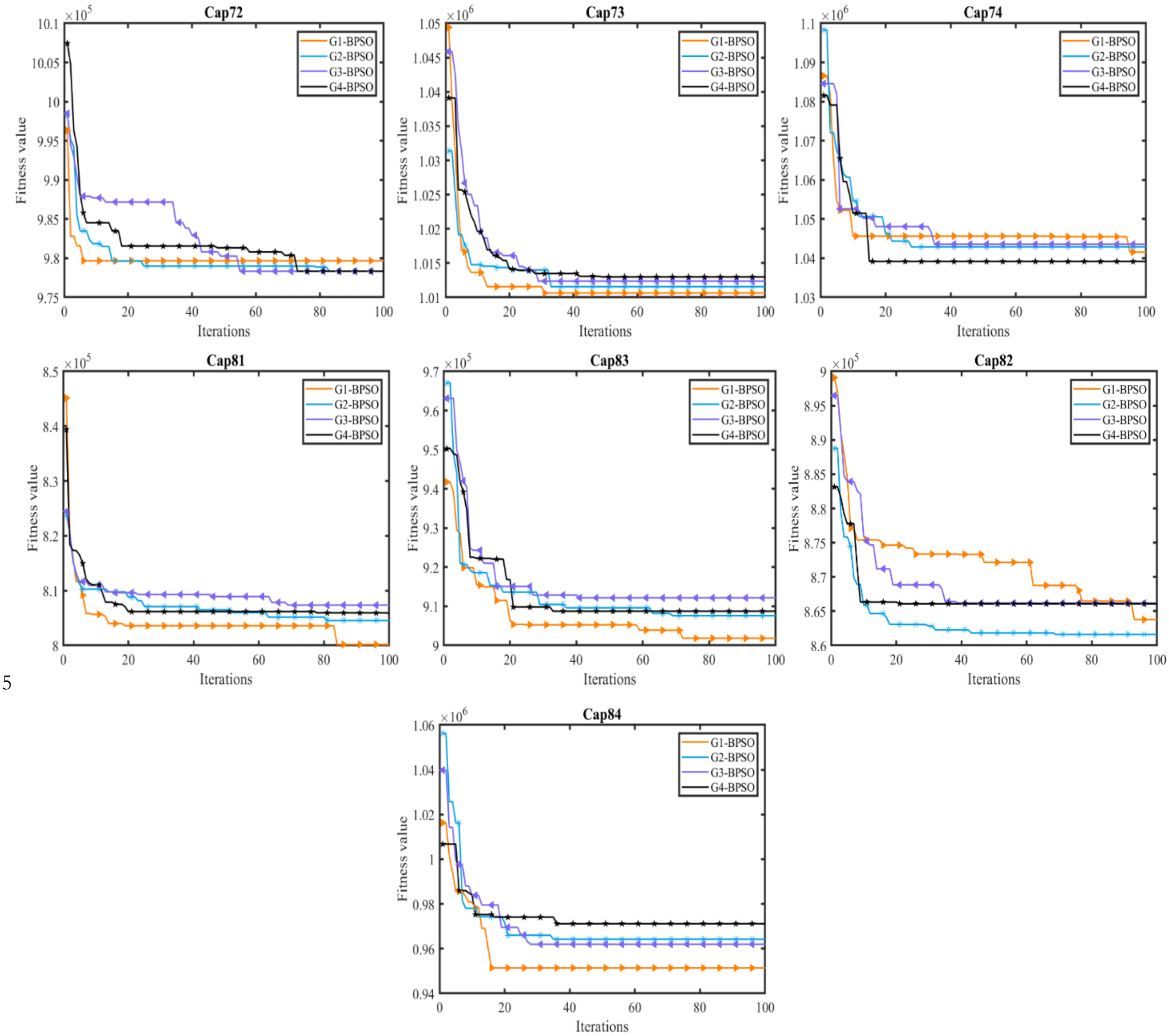
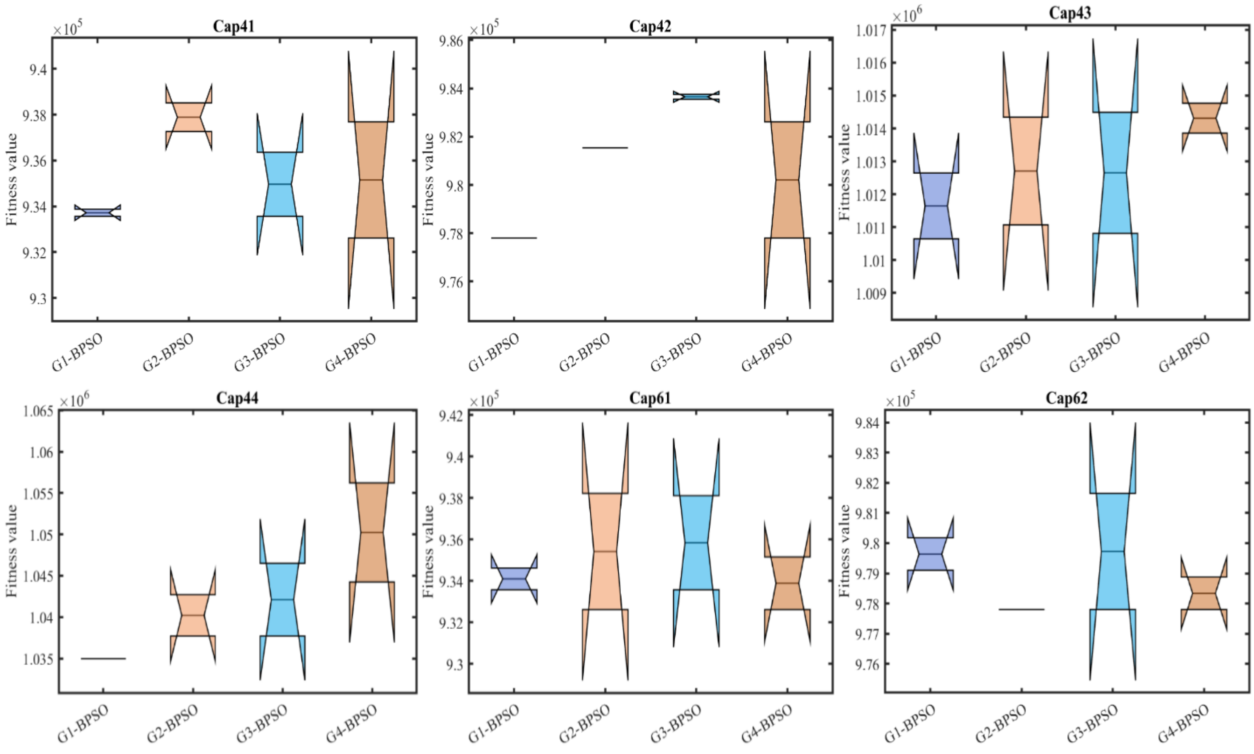


Fig. 2: Convergence curve of G-BPSO family algorithms on UFLP problems



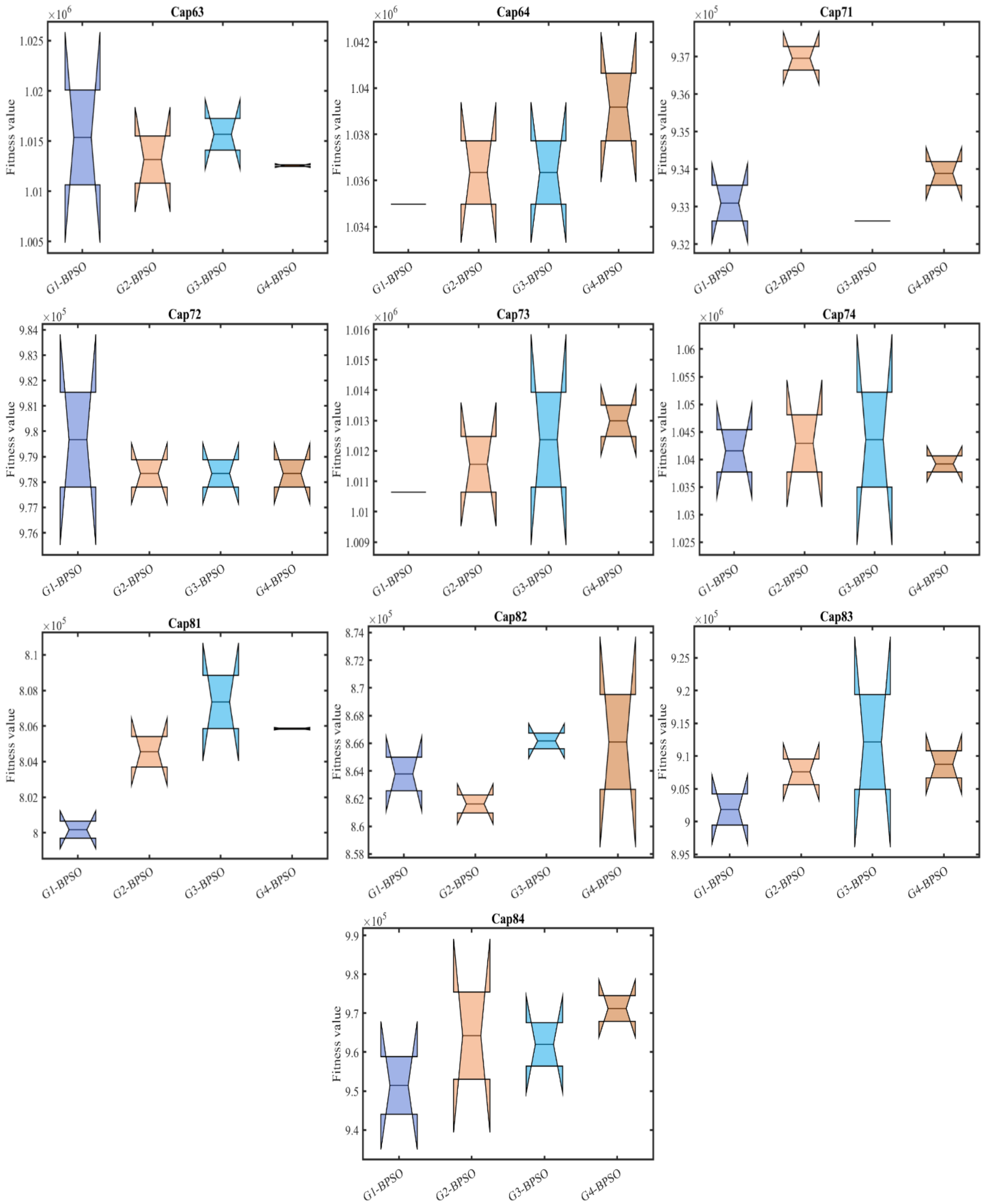


Fig. 3. Box Plot of G-BPSO family algorithms on UFLP problems.

G1-BPSO obtains the lowest average value in 9 out of the 16 tested UFLP instances. Some of these are Cap41, Cap42, Cap43, Cap44, Cap64, Cap73, Cap81, Cap83 and Cap84. These consistent performances to achieve lower average values make G1-BPSO more reliable to maintain stable solutions over iterations. As for problems Cap 61 and Cap 74, it can be stated that G4-BPSO is the most effective one, reaching the lowest average values. G4-BPSO does this in 2 of the 16 problems meaning that it is able to find the optimal solutions in the case of these two problems. Concerning problems Cap62 and Cap82, G2-BPSO is more effective attaining the lowest average. G2-BPSO does this effectively in 2 of the 16 problems, which shows its ability to solve the problems optimally. Specifically, for problem Cap63, G4-BPSO is the best as it has the minimum average value. Thus, for problem Cap71, G3-BPSO is better as it has given the least average value. In problem Cap72, all the groups, G2-BPSO, G3-BPSO, G4-BPSO are equally efficient and they have the lowest average values. As seen, G1-BPSO has the lowest mean value in 56 % of the problems (9 out of 16), which clearly indicates that it is efficient in finding the global solutions. G4-BPSO accomplishes this in 19% of the problems (3 out of 16) proving its efficiency, yet randomness in comparison to G1-BPSO. G2-BPSO achieves this in only 13% of the problems (2 out of 16). G3-BPSO manages to do this in 6% of the problems (1 out of 16). These consistencies of G1-BPSO indicate that the algorithm can perform well consistently throughout the different runs depicted in the box plot (Fig. 3).

Comparing the variability of the results using standard deviation, the proposed G1-BPSO has less variability in 9 of the 16 problems. The problems are Cap41, Cap42, Cap44, Cap61, Cap64, Cap73, Cap74, Cap81 and Cap83. Thus, the effectiveness of the G1-BPSO in offering consistent and accurate solutions with low fluctuations between the various runs is evident. G4-BPSO is the best in solving all the four problems, Cap43, Cap63, Cap74 and Cap81 as it has the lowest standard deviation. G4-BPSO achieves this in 4 out of the 16 problems that were tested, which shows that it can produce reliable results in these problems. In problems Cap62 and Cap82, G2-BPSO is more favorable since it yields the lowest standard deviation. G2-BPSO achieves this in only 2 out of the 16 problems. In problem Cap71 and Cap84, G3-BPSO is best as it results in the lowest standard deviations. G3-BPSO is able to do this in 2 of the 16 problems. In problem Cap42, both G1-BPSO and G2-BPSO have the lowest standard deviation values which indicate that both algorithms are equally efficient. Likewise, G2-BPSO, G3-BPSO and G4-BPSO have the least standard deviation in problem Cap72 and hence, the performance is equal. From the analysis of the results, it can be noted that G1-BPSO yields the lowest standard deviation in 56% of the problems, thus showing that the proposed approach can yield more consistent solutions. This is done in 25% of the problems, again highlighting the effectiveness, but also the variability of G4-BPSO compared to G1-BPSO. G2-BPSO does this in 13% of the problems (2 out of 16) and G3-BPSO in 13% of the problems (2 out of 16). This constant behavior of G1-BPSO implies that the algorithm is reliable in attaining high performance in different runs depicted by the box plot in Fig. 3.

the computational time of the algorithms, it is evident that G2-BPSO was quicker in solving 6 of the 16 problems which are Cap41, Cap42, Cap61, Cap71, Cap72 and Cap84. G4-BPSO is the most efficient in seven problems out of sixteen problems which are Cap43, Cap44, Cap63, Cap64, Cap73, Cap74 and Cap81. Out of the four algorithms, G3-BPSO is the most efficient in solving Cap83. G1-BPSO is the fastest when it comes to solving Cap62. Even though the approach of G1-BPSO does not show the best time in most of the problems, it is relatively efficient in the most cases. With reference to computational time, G2-BPSO and G4-BPSO algorithms are more convergent in a higher number of cases. We can see that G2-BPSO is the fastest in 6 out of 16 problems (37. 5%). In the overall comparison, G4-BPSO outperforms all the other algorithms in solving 7 of 16 problems which is 43.75%. Thus, G3-BPSO is the fastest in one of the sixteen problems, which is equal to 6. 25%. G1-BPSO takes the least time in one problem out of 16 (6. 25%). This distribution also demonstrates that G2-BPSO and G4-BPSO are more efficient in terms of time than the other algorithms, but it likewise shows that, while G1-BPSO is not the fastest algorithm overall, it can be successful in certain conditions.

- Minimum Value: G1-BPSO > G4-BPSO > G2-BPSO ~ G3-BPSO
- Maximum Value: G1-BPSO > G2-BPSO > G3-BPSO ~ G4-BPSO
- Average Value: G1-BPSO > G2-BPSO > G3-BPSO ~ G4-BPSO
- Standard Deviation: G1-BPSO > G2-BPSO > G3-BPSO ~ G4-BPSO
- Time: G1-BPSO > G2-BPSO ~ G4-BPSO > G3-BPSO

The comparative performance measure for the minimum value, the maximum value, the average value and variance with respect to time suggests that G1-BPSO outperforms all other BPSO variants for nearly all the UFLP problems. For this reason, the degree of accuracy in determining the best solution values with low standard deviations and the CPU time make G1-BPSO the most suitable algorithm to solve the set of UFLP problems. It has been noted that sometimes G2-BPSO, G3-BPSO and G4-BPSO could perform well on some UFLPs while G1-BPSO outperforms and is more stable over a wide range of cases. The fact that it has the unique ability to find the best solutions within small standard deviations and reasonable CPU time makes it the most reliable tool for solving the given UFLP problem set.

5 CONCLUSION

This study introduced four novel Gaussian-shaped transfer functions designed through power functions, leading to the development of the binary particle swarm optimizer, G-BPSO for solving binary optimization problems. Comparative analysis of discrete particle swarm optimizer demonstrates that Gaussian-shaped transfer functions are optimal for PSO discretization. Furthermore, G-BPSO not only proves to be an exceptional algorithm for solving UFLP but also serves as a valuable reference for designing discrete evolutionary algorithms.

6 ACKNOWLEDGMENTS

This article was co-funded by the European Union under the REFRESH – Research Excellence For REgion Sustainability and High-tech Industries project number CZ.10.03.01/00/22_003/0000048 via the Operational Programme Just Transition and has been done in connection with project Students Grant Competition SP2024/087 "Specific Research of Sustainable Manufacturing Technologies“ financed by the Ministry of Education, Youth and Sports and Faculty of Mechanical Engineering VŠB-TUO.

7 REFERENCES

[Abdel-Basset 2024] Abdel-Basset, M., Mohamed, R., Saber, S., Hezam, I. M., Sallam, K. M., & Hameed, I. A. (2024). Binary metaheuristic algorithms for 0-1 knapsack problems: Performance analysis, hybrid variants, and real-world application. Journal of King Saud University-Computer and Information Sciences, p. 102093. https://doi.org/10.1016/j.jksuci.2024.102093

[Alultan 1999] Alultan, K. S., & Al'Fawzan, M. A. (1999). A tabu search approach to the uncapacitated facility location problem. Annals of Operations Research, vol. 86(1), pp. 91–103.

[Baldomero-Naranjo 2024] Baldomero-Naranjo, M., Kalcsics, J., & Rodríguez-Chía, A. M. (2024). On the complexity of the upgrading version of the Maximal Covering Location Problem. Networks. https://doi.org/10.1002/net.22207

[Blekos 2024] Blekos, K., Brand, D., Ceschini, A., Chou, C. H., Li, R. H., Pandya, K., & Summer, A. (2024). A review on quantum approximate optimization algorithm and its variants. Physics Reports, vol. 1068, pp. 1–66. https://doi.org/10.1016/j.physrep.2024.03.002

[Choppakatla 2024] Choppakatla, N. D., Sivalenka, M. K. C., & Boda, R. (2024). Task ordering in multiprocessor embedded system using a novel hybrid optimization model. Multimedia Tools and Applications, pp. 1–25. https://doi.org/10.1007/s11042-024-19083-1

[Deng 2024] Deng, S., Xiao, S., Deng, Q., & Lu, H. (2024). A hovering swarm particle swarm optimization algorithm based on node resource attributes for hardware/software partitioning. The Journal of Supercomputing, vol. 80(4), pp. 4625–4647. https://doi.org/10.1007/s11227-023-05603-7

[Eberhart 1995] Eberhart, R., & Kennedy, J. (1995). A new optimizer using particles swarm theory. In Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan.

[Ghaderi 2013] Ghaderi, A., & Jabalameli, M. S. (2013). Modeling the budget-constrained dynamic uncapacitated facility location-network design problem and solving it via two efficient heuristics: a case study of health care. Mathematical and Computer Modelling, vol. 57(3-4), pp. 382–400. https://doi.org/10.1016/j.mcm.2012.06.017

[Kennedy 1997] Kennedy, J., & Eberhart, R. (1997). A discrete binary version of the particle swarm algorithm. In Proceedings of the IEEE International Conference on Computational Cybernetics and Simulation.

[Kuehn 1963] Kuehn, A. A., & Hamburger, M. J. (1963). A heuristic program for locating warehouses. Management Science, vol. 9, pp. 643–666. https://doi.org/10.1287/mnsc.9.4.643

[Ling 2010] Ling, W., Fu, X., Menhas, M., & Fei, M. (2010). A Modified Binary Differential Evolution Algorithm. In Li, K., Fei, M., Jia, L., & Irwin, G. (Eds.), Life System Modeling and Intelligent Computing (vol. 6329), Springer Berlin / Heidelberg, pp. 49–57. https://doi.org/10.1007/978-3-642-15597-0_6

[Ling 2010] Ling, W., Xu, Y., Mao, Y., & Fei, M. (2010). A Discrete Harmony Search Algorithm. In Li, K., Li, X., Ma, S., & Irwin, G. W. (Eds.), Life System Modeling and Intelligent Computing (vol. 98), Springer Berlin Heidelberg, pp. 37–43. https://doi.org/10.1007/978-3-642-15859-9_6

[Luh 2011] Luh, G., & Lin, C. (2011). A binary particle swarm optimization for continuum structural topology optimization. Applied Soft Computing, vol. 11, pp. 2833–2844. https://doi.org/10.1016/j.asoc.2010.11.013

[Mafarja 2017] Mafarja, M., Eleyan, D., Abdullah, S., & Mirjalili, S. (2017). S-shaped vs. V-shaped transfer functions for ant lion optimization algorithm in feature selection problem. In Proceedings of the International Conference on Future Networks and Distributed Systems (ICFNDS '17). https://doi.org/10.1145/3102304.3102325

[Mafarja 2018] Mafarja, M., Aljarah, I., Heidari, A. A., Faris, H., Fournier-Viger, P., Li, X., & Mirjalili, S. (2018). Binary dragonfly optimization for feature selection using time-varying transfer functions. Knowledge-Based Systems, vol. 161, pp. 185–204. https://doi.org/10.1016/j.knosys.2018.08.003

[Mirjalili 2012] Mirjalili, S., & Hashim, S. Z. M. (2012). BMOA: binary magnetic optimization algorithm. International Journal of Machine Learning and Computing, vol. 2(3), pp. 204–208. https://doi.org/10.7763/IJMLC.2012.V2.114

[Mirjalili 2013] Mirjalili, S., & Lewis, A. (2013). S-shaped versus V-shaped transfer functions for binary particle swarm optimization. Swarm and Evolutionary Computation, vol. 9(4), pp. 1–14. https://doi.org/10.1016/j.swevo.2012.09.002

[Ozsoydan 2024] Ozsoydan, F. B., & Kasırga, A. E. (2024). Evolution inspired binary flower pollination for the uncapacitated facility location problem. Neural Computing and Applications, pp. 1–14. https://doi.org/10.1007/s00521-024-09684-0

[Premalatha 2024] Premalatha, M., Jayasudha, M., Čep, R., Priyadarshini, J., Kalita, K., & Chatterjee, P. (2024). A comparative evaluation of nature-inspired algorithms for feature selection problems. Heliyon, vol. 10(1). https://doi.org/10.1016/j.heliyon.2023.e23571

[Rashedi 2009] Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). BGSA: binary gravitational search algorithm. Natural Computing, vol. 9(3), pp. 727–745. https://doi.org/10.1007/s11047-009-9175-3