JOURNAL OF AERONAUTICS AND SPACE TECHNOLOGIES
(ISSN : 1304-0448)
July 2021 Volume 14 Number 2
jast.hezarfen.msu.edu.tr

Research Article

Modified Grey Wolf Optimizer with Parasitism Phase

Ali Nadi ÜNAL[1*], Funda SEÇKİN[2]

[1]*National Defense University, Department of Industrial Engineering , 34000 Yeşilyurt , Istanbul, Turkey*, anunal@hho.msu.edu.tr, http://orcid.org/0000-0002-6956-1514
[2]*National Defense University, Department of Industrial Engineering , 34000 Yeşilyurt , Istanbul, Turkey*, ftan@hho.msu.edu.tr, http://orcid.org/0000-0002-5691-7366
* *Corresponding Author*

Article Info

Received: *January 13, 2021*
Accepted: *April 26, 2021*
Online: *July 26, 2021*

Keywords: Grey Wolf Optimizer (GWO), Metaheuristics, Symbiotic Organisms Search (SOS), Swarm Intelligence

Abstract

Grey Wolf Optimizer (GWO) is a popular and effective nature-inspired, swarm intelligence based metaheuristic optimization algorithm which imitates the hunting process of a wolf pack in the nature. It has been widely used to solve both theoretical and real life engineering optimization problems. Even though metaheuristics have many advantages such as requiring no derivative information of the problem at hand, simplicity, and flexibility, they also have some drawbacks like trapping local optima, and premature convergence. They should have a proper balance between diversification (exploration) and intensification (exploitation). In this study, inclusion of a simple operator, namely "parasitism", is proposed to improve the performance of GWO by means of exploration. Parasitism is a phase within Symbiotic Organisms Search (SOS) algorithm. The performances of 2 modified versions are compared with the original GWO, using 13 test beds. Results indicate that inclusion of parasitism phase into the original version has produced better results on the parameters of the algorithm.

To Cite This Article: Ali Nadi ÜNAL, Funda SEÇKİN, "Modified Grey Wolf Optimizer with Parasitism Phase", Journal of Aeronautics and Space Technologies, Vol. 14, No. 2, pp. 133-144, July 2021.

Parazitizm Safhası ile Değiştirilmiş Bozkurt Eniyileme Algoritması

Makale Bilgisi

Geliş: *13 Ocak 2021*
Kabul: *26 Nisan 2021*
Yayın: *26 Temmuz 2021*

Anahtar Kelimeler: Bozkurt Eniyileme Algoritması, Metasezgiseller, Simbiyotik Organizmalar Arama Algoritması, Sürü zekâsı

Öz

Bozkurt Eniyileme Algoritması,vahşi doğadaki bir bozkurt sürüsünün avlanma sürecini taklit eden, doğadan esinlenmiş, sürü zekâsı tabanlı, popüler ve etkili bir metasezgisel eniyileme algoritmasıdır. Algoritma hem teorik hem de gerçek hayat problemlerini çözmek için geniş bir alanda kullanılmaktadır. Metasezgiseller eldeki problem için herhangi bir türev bilgisine ihtiyaç duyulmaması, basitlik ve esneklik gibi avantajları olmasına rağmen yerel en iyi çözümde takılıp kalma, erken yakınsama gibi istenilmeyen özelliklere de sahiptir. Metasezgiseller keşif ve yoğunlaşma özelliklerini dengeli bir şekilde kullanmalıdır. Bu çalışmada Bozkurt Eniyileme Algoritmasının keşif yeteneğini geliştirmek amacıyla "parazitizm" olarak adlandırılan basit bir operatörün eklenmesi önerilmiştir. Parazitizm Simbiyotik Organizmalar Araması algoritmasının bir safhasını teşkil eder. Değiştirilmiş algoritmanın iki varyantının performansı 13 test problemi kullanılarak orijinal Bozkurt Eniyileme Algoritması ile karşılaştırılmıştır. Elde edilen sonuçlara göre, orijinal sürüme parazitim safhasının eklenmesi algoritma parametrelerinde daha iyi sonuçların üretilmesini sağlamıştır.

1. INTRODUCTION

Every research area has its own challenging optimization problems which represent a highly complex structure in their nature. Researchers and practitioners are trying to model and solve these complex, multi-modal, and huge problems to facilitate modern human life. Although classical techniques like gradient based deterministic methods, mixed integer linear programming, dynamic programming, branch and bound approaches have some advantages, they may fail to solve these kind of problems [1–3]. For that reason, metaheuristic optimization methods are accepted as one of the most important topic for real world optimization. [4, 5]. They have been widely used not only in science but also in industry [6, 7]. They are specified with their nature inspired, stochastic, derivative free, adaptive, simple and flexible characteristics [8–10].

They can be divided into three main categories named, (i) evolutionary algorithms (EAs) such as Genetic Algorithms (GA) and Differential Evolution (DE), (ii) physics based techniques such as gravitational search algorithm (GSA) and artificial chemical reaction optimization algorithm, (iii) swarm intelligence (SI)

based techniques such as particle swarm optimization (PSO) and ant colony optimization (ACO) [5, 8, 9].

Due to their quite simple implementation and capability on achieving near optimal solutions within a reasonable time frame, nature inspired computational techniques have become popular, and they also have space for possible improvements [11]. In this regard, as a subset of metaheuristics, SI is one of the emerging research fields of the past decade, and its numerous types of algorithms are accepted as the most successful and competitive methods for solving numerical optimization problems especially handling non-linear, non-convex, discontinuous, or combinatorial optimization problems [1, 7, 12, 13].

GWO is one of the most recent swarm intelligence-based algorithms. Since the first publication by Mirjalili et al. [14] in 2014 the GWO has recently received significant interest within a very short time.

GWO has been proved its superior performance among other conventional population based algorithms such as PSO, DE, and GSA [15, 16]. It has many desired abilities and advantages such as fast convergence, avoidance of trapping local optima, simplicity, flexibility, adaptability, stability, easy implementation, fewer parameters to adjust, and powerful search ability [1, 15–19].

Although its above-mentioned advantages and abilities, several studies have noted the drawbacks or disadvantages. According to Jitkongchuen et al. [20] "traditional GWO has opportunity to trap in local minimum since it uses only one pack of wolves." Jaiswal et al. [19] stated that, especially for complex multi-modal functions, GWO tends to possess premature convergence at later iterations, poor quality solution and local optima stagnation at few problem instances. In addition to local optima stagnation issue, Long et al. [16] stated that convergence rate of the algorithm will decrease considerably in the later period of evolution. They also draw attention to performance reduction for complicated problems due to the algorithm's memory-less population-based nature. Kohli and Arora [21] stated that there exist some problems on implementation of global search, because of the search strategy depends on random walks. Yan et al. [1] pointed out two problems named premature convergence, and imbalance between exploration and exploitation. Khanum et al. [22] also stated that GWO shows superiority in exploitation while it is inefficient in exploration. Similar to other SI algorithms GWO may have a deficiency in preserving diversity since it depends on the best three solutions found through the iterative search [18]. A detailed literature review on the GWO [23] stated that there are still several areas that need new or further works, and implied that the exploration and/or exploitation of GWO can be improved with employing operators from other algorithms. This is the main motivation for the current study.

The main contribution of this study is proposal of a simple but effective method to improve classical GWO in terms of exploration capability. A phase called "parasitism" is adopted from SOS algorithm. In parasitism phase of SOS algorithm [24], a member of population, supposedly X_j (j^{th} member of the population), is randomly selected and duplicated. It is called "parasite vector". Some selected dimensions of this parasite vector are modified. If parasite vector has a better fitness value, F, compared with X_j, it replaces X_j. This procedure is similar to "mutation" operator, which is originally designed as a genetic operator. Employing genetic operators such as mutation and perturbation may promote diversity, and enhances the exploration capability of population-based algorithms [25, 26].

To the best of authors' knowledge, this is the first study that proposes inclusion of SOS parasitism phase into the GWO. Performance of modified version of the algorithm is compared with the original version using 13 test beds.

The remainder of this paper is organized as follows. Section 2 focuses on previous studies on modified versions of GWO. It is followed by a summary of original GWO procedure and the proposed modified version. Experimentation, computational results, and discussions are provided in Section 4. Finally, Section 5 concludes this paper.

2. RELATED WORK

According to the No Free Lunch (NFL) theorem there is no algorithm that appropriately suits for all optimization problems [5]. In other words, when an algorithm A outperforms algorithm B for a specific problem, then algorithm B may outperforms algorithm A for other types of problems [27]. Based on this philosophy, there is a vast optimization literature on both improving previously proposed algorithms, and introducing new algorithms as well. Regarding the GWO literature, it has been observed that the studies on GWO can be included in four categories named, (i) change in parameters, (ii) hybridization, (iii) introducing new operators, and (iv) modification of position update equation [22]. A brief review on recent studies which enhances or modifies the GWO is presented in this section.

Saremi et al. [28] proposed the use of evolutionary population dynamics (EPD) operator in GWO. In their proposal the poorest half of the population is re-positioned around the best three (i.e., alpha, beta, delta) or totally randomized fashion within an equal probability. They concluded that inclusion of the EPD operator improves the performance of the GWO on both unimodal and multi-modal test functions, which are composed of 13 instances.

Belkacem and Srairi [29] hybridized the GWO with Pattern Search (GW-PS) in order to develop a

generalized optimal security power system planning strategy. The proposed method searches equilibrium between exploration and exploitation during evaluation process.

Zhang and Zhou [30] proposed a novel GWO, based on Powell local optimization method (PGWO). Powell's method served as a critical element to perform the local search exploiting the limited space intensively to get better solutions. Their method also tried to avoid search agents trapping local optimal regions, and to balance exploration and exploitation. They reported the effectiveness of proposed method both for solution quality and convergence speed.

Zhou et al. [31] designed a novel variant of the GWO called "GWO with chaotic local search (CLSGWO)" making an improvement. They incorporated a chaotic mutation operator into the search process in order to enhance local search ability of the algorithm. They noted that the proposed modification showed a good performance after conducting two real experiments.

Kishor and Singh [32] proposed modified grey wolf optimizer (MGWO), which improves the information sharing mechanism of the GWO. Firstly, they investigated the performance of the original GWO, and demonstrated that it lacks exploration. Secondly, they proposed the modified version. They incorporated a crossover operator into the GWO. They concluded that the modified version solutions are better than the original one by means of the solution quality and the convergence speed.

Rodriguez et al. [33] introduced a hierarchical operator into the original GWO. The proposed operator is a hierarchical transformation that is inspired in the hierarchical social pyramid of the grey wolf. They proposed five variants that are based on fuzzy logic. It was noted that the proposed variants have great impact in the GWO.

Miao et al. [34] proposed a GWO with enhanced hierarchy (GWO-EH) to mitigate premature convergence and local optima stagnation deficiencies. They introduced fitness-based self-adaptive weight coefficients, and they used an improved position updating equation. In addition to these two modifications, they redesigned the strategy of repositioning the wolves to keep the balance between exploration and exploitation. They reported competitive overall performance against some other promising GWO variants.

Nadimi-Shahraki et al. [35] proposed an Improved Grey Wolf Optimizer (I-GWO) to mitigate the lack of population diversity and premature convergence, in addition to balance exploration and exploitation of the GWO. Their dimension learning based hunting (DLH) strategy enables wolves to share neighboring information with each other. They concluded that the new improved variant is very competitive and generally better than the other algorithms compared.

3. GREY WOLF OPTIMIZER

Nature inspired metaheuristic optimization methods mimic the behavior of the animals in nature in order to optimize a function within allowed search space [36]. GWO optimization mimics the leadership hierarchy and hunting behavior of grey wolves. It was first introduced by Mirjalili et al. [14], and it has been very popular among the researchers in metaheuristics field.

3.1. Original Version

As introduced firstly in [14], four types of wolves namely alpha, beta, delta, and omega create a wolf pack in a hierarchical fashion. The GWO is based on the three phases of the hunting mechanism of a wolf pack. These phases are, (i) searching for prey, (ii) encircling prey, and (iii) attacking pray. The GWO is inspired of these phases, and uses several equations in order to model the hunting mechanism mathematically. Like the other SI techniques, modeling equations are updated iteratively until a predefined condition (e.g., certain number of iterations).

Equations and coefficients of the GWO are as follows [14]:

$$\vec{D} = \left|\vec{C} \cdot \vec{X}_p(t) - \vec{X}(t)\right| \quad (1)$$

$$\vec{X}(t+1) = \vec{X}(t) - \vec{A} \cdot \vec{D} \quad (2)$$

$$\vec{A} = 2\vec{a} \cdot \vec{r}_1 - \vec{a} \quad (3)$$

$$\vec{C} = 2 \cdot \vec{r}_2 \quad (4)$$

where t is the iteration counter, $\vec{A}$ and $\vec{C}$ are coefficient vectors, $\vec{D}$ is a vector which is used for position updating, $\vec{X}_p$ is the position vector of the prey, and $\vec{X}$ is the position vector of the search agent (i.e., grey wolf). Components of $\vec{a}$ are linearly decreased from 2 to 0 throughout iterations ($\vec{a}_t = 2 - t * \left(\frac{2}{numIter}\right)$, where t is the current number of iteration, and *numIter* is the maximum number of iterations. $\vec{r}_1, \vec{r}_2$ are random vectors in [0,1].

$$\vec{D}_\alpha = \left|\vec{C}_1 \cdot \vec{X}_\alpha - \vec{X}\right|, \vec{D}_\beta = \left|\vec{C}_2 \cdot \vec{X}_\beta - \vec{X}\right|, \quad (5)$$
$$\vec{D}_\delta = \left|\vec{C}_3 \cdot \vec{X}_\delta - \vec{X}\right|$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 \cdot (\vec{D}_\alpha), \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 \cdot (\vec{D}_\beta), \quad (6)$$
$$\vec{X}_3 = \vec{X}_\delta - \vec{A}_3 \cdot (\vec{D}_\delta)$$

$$\vec{X}(t+1) = (\vec{X}_1 + \vec{X}_2 + \vec{X}_3)/3 \quad (7)$$

where $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ represent the first three best solutions in the wolf pack. Positions of other search agents are updated according to the position of these three, as shown in Eq.(7).

The pseudo-code for the GWO is given in Algorithm 1.

Algorithm 1. Pseudo-code for original GWO algorithm.

1. Generate the wolf pack
2. Define a, A, C
3. Calculate the fitness values of each search agent (i.e., wolves)
4. Define the first best three agent with respect to the fitness values (i.e., alfa, beta, and delta)
5. while stopping condition not met
6. **for** each search agent
7. Update the position of the current agent using equation (3.7)
8. **end for**
9. Update a, A, and C
10. Calculate the fitness of each agent
11. Update alfa, beta, and delta
12. end while
13. return alfa

3.2. Proposed Modified Versions

We applied two different modification schemes. Each modification is applied after the 8th step of Algorithm 1. In other words, after updating the positions of all members of the wolf pack, parasitism phase [37] is included.

In the first one, a random search agent $\vec{X}_j$ is selected and copied. This is called parasite vector. Throughout all dimensions of this parasite vector, respected dimension is randomly changed within the boundaries if random number $r_1$is less than another random number r_2, where r_1 and $r_2$in [0,1]. After that, if the fitness value of the parasite vector, $F_{parasite}$, is less than the fitness value of randomly selected $\vec{X}_j$ (assuming minimization), F_j, parasite vector replaces $\vec{X}_j$.

For the second modification scheme, in a similar manner after the 8th step of original version in Algorithm 1, a random member $\vec{X}_j$ of the wolf pack is selected and omitted. Instead of that omitted member, a randomly generated member is included regardless of the fitness value, at each iteration. This inclusion can be considered as a limited execution of "random immigrants" approach. Random immigrants approach is employed to EAs in order to increase the diversity within the population [3], [38]. Since it has a potential to increase the diversity of the population, inclusion of random immigrants approach within a limited fashion might be efficient for multi-modal problem landscapes, even a leader wolf is likely to be omitted.

4. RESULTS AND DISCUSSION

This section debates the effectiveness of the proposed modification, comparing the original one.

We use benchmark problems for optimization, similar to other studies in literature. 13 problems are employed as given in Ref. [14] . Used test benchmarks are given in Table 1. The first 7 equations are unimodal and the others are multi-modal. Roughly speaking, a unimodal function has a single global optima, while a multi-modal function has more than one "mode" or optima. Multi-modal equations are useful for testing exploration ability.

Original GWO and two proposed variants are run 30 times. The number of search agents is 50, and the number of maximum iterations is 500. For each test problem, minimum, maximum, mean, and standard deviation of the solutions obtained in last iteration and execution times are recorded. Experimentation was implemented in Python, using EvoloPy [27], on a Packard Bell© PC, Intel(R) Core (TM) i5-4200U CPU @ 1.60Ghz, and 8 GB RAM.

The results obtained for unimodal, and multi-modal functions are given in Table 2, and Table 3 respectively. The results presented in Table 2 show that the first modified variant shows competitive performance compared with the original version, by means of solution quality (obtained minimum value) and CPU times. The first variant outperforms the original one through 5 problem instant in 7. For the 4th test problem, the first modified variant shows better performance significantly. This is likely because of replacing strategy of the first modified variant. As explained in Section 3.2 a search agent is replaced by the mutated one if and only if the mutated one has a better (less) fitness value.

Regarding the second modified variant, for unimodal test functions, it doesn't have a good performance compared with the original version and the first variant, except for the 7th problem. We believe that the reason of poor performance may also be related to the replacing strategy of parasitism phase. As explained previously, randomly selected search agent is replaced with a randomly generated one, regardless of the fitness value. This process may result changing the first three agents (i.e. alfa, beta, and delta) even they have a better fitness value compared with the randomly generated one. For some iterations, this process may affect the search performance negatively. The second variant was proposed to test whether inclusion of a randomly generated member improves the search capability regardless of the fitness function values, but it seems that this scheme doesn't have an improving effect for unimodal test functions.

Regarding the multi-modal test beds, according to the results given in Table 3, the first modified version of the GWO shows competitive performance compared with the original version, by means of solution quality and CPU times. Specifically, for the 9th and the 13th functions, the first modified version has a better performance compared with the original version. On the other hand, for the 12th and 8th functions, the

original version outperforms the first modified variant significantly. For the 10th and 11th functions both original version and the first variants have the same minimum value. The second variant also yields competitive performance compared with the original GWO. The original version significantly outperforms the second variant just for the 8th problem. The second variant significantly has a better performance compared with the original version, for the 12th and 13th functions.

Convergence characteristics of the original GWO and two proposed variants are given in Fig.1. For each test problem, and for three algorithms, recorded fitness values at each iteration were averaged over 30 runs, and these results were used to plot convergence characteristics. As shown in Figure 1, each of three algorithms has a fast convergence capability.

To sum up, the first variant has a competitive or superior performance compared with the original GWO for both unimodal and multi-modal test beds. Yet, the second variant is not competitive at all compared with the original one for unimodal test beds. This is likely because of the nature of the search space for unimodal test functions. For relatively simple structure of the unimodal search space, GWO favors exploitation capacity instead of exploration. Additionally, replacement strategy for the second variant, which is based on totally random generation of a new member regardless of the fitness value, may be another reason. On the other hand, for challenging multi-modal search spaces, the algorithm needs to enhance the exploration capacity. In this regard, the second variant has a competitive or superior performance compared with the original version, since it improves the exploration capacity employing a random inclusion scheme.

5. CONCLUSION

This paper proposed the use of SOS algorithm's parasitism operator in the GWO algorithm. Two variants of parasitism were studied. In the first one, a random search agent from the wolf pack was selected and copied at each iteration. This is called parasite vector. Positions of this parasite vector were manipulated randomly. After that, this parasite vector replaced the original one if and only if it had a better fitness value. For the second one, a totally randomly generated parasite vector replaced a randomly selected search agent regardless of the fitness value. 13 benchmark functions were used to test the performance. Compared statistical results showed that inclusion of parasitism phase may improve the performance of the original GWO. On the other hand, the second variant, may not be effective since the randomly selected and replaced agent can be one of the leader wolves.

As a future recommendation, inclusion of further variants for parasitism operator can be studied throughout the newly proposed metaheuristic algorithms. Additionally, it would be interesting to test modified GWO on constrained real life optimization problems within different fields such as energy systems planning, unmanned air vehicle route planning, portfolio optimization. Furthermore, it would also be interesting to employ modified GWO on multi-objective optimization problems.

6. REFERENCES

[1] F. Yan, X. Xu, J. Xu, "Grey wolf optimizer with a novel weighted distance for global optimization", *IEEE Access*, vol.8, June 2020.

[2] L. K. Panwar, S. Reddy K, A. Verma, B. K. Panigrahi, R. Kumar, "Binary grey wolf optimizer for large scale unit commitment problem", *Swarm Evol. Comput.*, vol. 38, pp. 251–266, February, 2018.

[3] A.N.Ünal, G. Kayakutlu, "Multi-objective particle swarm optimization with random immigrants", *Complex Intell. Syst.*, vol. 6, pp. 635–650, June 2020.

[4] P.Niu, S.Niu, N.Liu, L. Chang, "The defect of the grey wolf optimization algorithm and its verification method", *Knowledge-Based Syst.*, vol. 171, pp. 37–43, May 2019.

[5] L. Rodríguez, O. Castillo, J. Soria, "Grey wolf optimizer with dynamic adaptation of parameters using fuzzy logic", in *2016 IEEE Congress on Evolutionary Computation, CEC 2016*, Vancouver, BC, Canada, July 24-29, 2016, pp. 3116–3123.

[6] S. Dhargupta, M. Ghosh, S. Mirjalili, R. Sarkar, "Selective Opposition based Grey Wolf Optimization", *Expert Syst. Appl.*, vol. 151, August. 2020.

[7] D. Jitkongchuen, P. Phaidang, P. Pongtawevirat, "Grey wolf optimization algorithm with invasion-based migration operation", in *2016 IEEE/ACIS 15th International Conference on Computer and Information Science, ICIS* 2016, Okayama, Japan, June 26-29, 2016.

[8] A. Saxena, R. Kumar, S. Das, "β-Chaotic map enabled grey wolf optimizer", *Appl. Soft Comput. J.*, vol. 75, pp. 84–105, February 2019.

[9]P. Gupta, K. P. S. Rana, V. Kumar, P. Mishra, J. Kumar, S. S. Nair, "Development of a grey wolf optimizer toolkit in LabVIEW™", in *2015 International Conference on Futuristic Trends on Computational Analysis and Knowledge Management ABLAZE* 2015, Greater Noida, India, February 25-27, 2015, pp. 107–113.

[10] S. Kohli, M. Kaushik, K. Chugh, A. C. Pandey, "Levy inspired enhanced grey wolf optimizer", in *2019 Fifth International Conference on Image Information Processing, ICIIP* 2019, Shimla, Indiai November 15-17, 2019, pp. 338–342.

[11] F. B. Ozsoydan, "Effects of dominant wolves in grey wolf optimization algorithm", *Appl. Soft Comput. J.*, vol. 83, October 2019.

[12] İ. Gölcük, F. B. Ozsoydan, "Evolutionary and adaptive inheritance enhanced grey wolf optimization algorithm for binary domains", *Knowledge-Based Syst.*, vol. 194, April 2020.

[13] K. Luo, "Enhanced grey wolf optimizer with a model for dynamically estimating the location of the prey", *Appl. Soft Comput. J.*, vol. 77, pp. 225–235, April 2019.

[14] S. Mirjalili, S. M. Mirjalili, A. Lewis, "Grey Wolf Optimizer", *Adv. Eng. Softw.*, vol. 69, pp. 46–61, March 2014.

[15] P. Hu, J. S. Pan, S. C. Chu, "Improved binary grey wolf optimizer and its application for feature selection", *Knowledge-Based Syst.*, vol. 195, May 2020.

[16] W. Long, J. Jiao, X. Liang, M. Tang, "Inspired grey wolf optimizer for solving large-scale function optimization problems", *Appl. Math. Model.*, vol. 60, pp. 112–126, August 2018.

[17] C. Lu, L. Gao, J. Yi, "Grey wolf optimizer with cellular topological structure", *Expert Syst. Appl.*, vol. 107, pp. 89–114, October 2018.

[18] M. A. Al-Betar, M. A. Awadallah, H. Faris, I. Aljarah, A. I. Hammouri, "Natural selection methods for grey wolf optimizer", *Expert Syst. Appl.*, vol. 113, pp. 481–498, December 2018.

[19] K. Jaiswal, H. Mittal, S. Kukreja, "Randomized grey wolf optimizer (RGWO) with randomly weighted coefficients", in *2017 Tenth International Conference on Contemporary Computing, IC3 2017*, Noida, India, August 10-12, 2017, pp. 1–3.

[20] D. Jitkongchuen, W. Sukpongthai, A. Thammano, "Weighted distance grey wolf optimization with immigration operation for global optimization problems", in *2017 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2017*, Kanazawa, Japan, June 26-28 2017, pp. 5–9.

[21]M. Kohli, S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems", *J. Comput. Des. Eng.*, vol. 5, pp. 458–472, October 2018.

[22] R. A. Khanum, M. A. Jan, A. Aldegheishem, A. Mehmood, N. Alrajeh, A. Khanan, "Two new ımproved variants of grey wolf optimizer for unconstrained optimization", *IEEE Access*, vol. 8, pp. 30805–30825, December 2020.

[23] H. Faris, I. Aljarah, M. A. Al-Betar, S. Mirjalili, "Grey wolf optimizer: a review of recent variants and applications", *Neural Comput. Appl.*, vol. 30, pp. 413–435, July 2018.

[24] P. Jangir, N. Jangir, "A new non-dominated sorting grey wolf optimizer (NS-GWO) algorithm: Development and application to solve engineering designs and economic constrained emission dispatch problem with integration of wind power", *Eng. Appl. Artif. Intell.*, vol. 72, pp. 449–467, June 2018.

[25] W. Leong, G. G. Yen, S. Member, "PSO-Based multiobjective optimization adaptive local archives", *IEEE Trans. Evol. Comput.*, vol. 38, pp. 1270–1293, October 2008.

[26] S.Mirjalili, T.Rawlins, A. Lewis, J. Hettenhausen, "A comparison of multi-objective optimisation metaheuristics on the 2D airfoil design problem", *Anziam J.*, vol. 54, pp. 345–360, July 2013.

[27] H. Faris., I. Aljarah., S. Mirjalili., P. A. Castillo., J. J. M., "EvoloPy: An Open-source Nature-inspired Optimization Framework in Python", in *Proceedings of the 8th International Joint Conference on Computational Intelligence - Volume 3: ECTA, IJCCI 2016*, Porto, Portugal, November 9-11, 2016, pp. 171–177.

[28]S.Saremi, S.Z.Mirjalili, S.M.Mirjalili, "Evolutionary population dynamics and grey wolf optimizer", Neural Comput. Appl., vol. 26, pp. 1257 1263, July 2015.

[29] B. Mahdad, K. Srairi, "Blackout risk prevention in a smart grid based flexible optimal strategy using grey wolf-pattern search algorithms", *Energy Convers. Manag.*, vol. 98, pp. 411–429, July 2015.

[30] S. Zhang, Y. Zhou, "Grey wolf optimizer based on powell local optimization method for clustering analysis", *Discret. Dyn. Nat. Soc.*, vol. 2015, November 2015.

[31] J. Zhou, W. Zhu, Y. Zheng, C. Li, "Precise equivalent model of small hydro generator cluster and its parameter identification using improved grey wolf optimiser", *IET Gener. Transm. Distrib.*, vol. 10, pp. 2108–2117, June 2016.

[32] A. Kishor, P. K. Singh, "Empirical Study of Grey Wolf Optimizer", in *Proceedings of Fifth International Conference on Soft Computing for Problem Solving*, 2016, pp. 1037–1049.

[33]L. Rodríguez *vd.*, "A fuzzy hierarchical operator in the grey wolf optimizer algorithm", *Appl. Soft Comput. J.*, vol. 57, pp. 315–328, August. 2017.

[34]Z. Miao, X. Yuan, F. Zhou, X. Qiu, Y. Song, K. Chen, "Grey wolf optimizer with an enhanced hierarchy and its application to the wireless sensor

network coverage optimization problem", *Appl. Soft Comput. J.*, vol. 96, November. 2020.

[35] M. H. Nadimi-Shahraki, S. Taghian, ve S. Mirjalili, "An improved grey wolf optimizer for solving engineering problems", *Expert Syst. Appl.*, vol. 166, March 2021.

[36] B. Martin, J. Marot, S. Bourennane, "Improved discrete grey wolf optimizer", in *2018 26th European Signal Processing Conference, EUSIPCO* 2018, Rome, Italy, September 3-7, 2018, pp. 494–498.

[37] M.-Y. Cheng, D. Prayogo, "Symbiotic Organisms Search: A new metaheuristic optimization algorithm", *Comput. Struct.*, vol. 139, pp. 98–112, July 2014.

[38] M. Mavrovouniotis, S. Yang, "Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors", *Appl. Soft Comput. J.*, vol. 13, pp. 4023–4037, October 2013.

VITAE

Ali Nadi ÜNAL received his B.Sc. degree in Industrial Engineering from Department of Industrial Engineering, Turkish Air Force Academy, Turkey in 2002. He received his M.Sc. degree in Defense Resources Management from Atatürk Institute of Strategic Studies, Turkey in 2010. He received his Ph.D. in Industrial Engineering from Hezarfen Aeronautics and Space Technologies Institute, Turkey in 2016. He is currently with The Turkish Air Force.

Funda SEÇKİN received her B.Sc. degree in Industrial Engineering from Department of Industrial Engineering, Turkish Air Force Academy, Turkey in 2001. She received her M.Sc. degree in Industrial Engineering from Balikesir University, Turkey in 2007. She received her Ph.D. in Industrial Engineering from Hezarfen Aeronautics and Space Technologies Institute, Turkey in 2016. She is currently working in the Department of Industrial Engineering at National Defense University.

Table 1. Benchmark Functions.

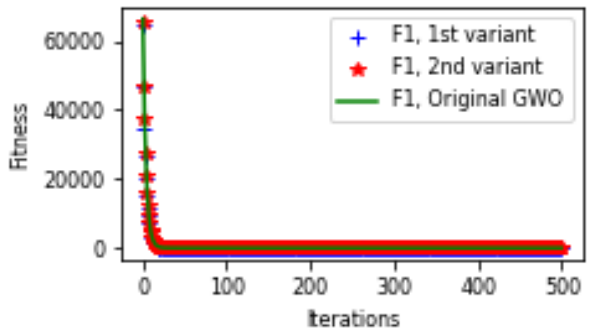
Function	**Range (decision variables)**
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]$
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]$
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j-1}^{i} x_j \right)^2$	$[-100, 100]$
$f_4(x) = max_i\{\lvert x_i \rvert, 1 \le i \le n\}$	$[-100, 100]$
$f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-30, 30]$
$f_6(x) = \sum_{i=1}^{n} ([x_i + 0{,}5])^2$	$[-100, 100]$
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	$[-1.28, 1.28]$
$F_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]$
$F_9(x) = \sum_{i=1}^{n} [x_i^2 - 10cos(2\pi x_i) + 10]$	$[-5.12, 5.12]$
$F_{10}(x) = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right) + 20 + e + 10$	$[-32, 32]$
$F_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]$
$F_{12}(x) = \frac{\pi}{n}\left\{10sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10sin^2(\pi y_{i+1})] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i + 1}{4}$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m & x_i > a \\ 0 & -a < x_i < a \\ k(-x_i - a)^m & x_i < -a \end{cases}$	$[-50, 50]$
$F_{13}(x) = 0.1\left\{sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2[1 + sin^2(3\pi x_i + 1)] + (x_n - 1)^2[1 + sin^2(2\pi x_n)]\right\}$ $+ \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]$
All of the functions have 30 dimensions. All but function $F_8(x)$ have a global minimum value of 0. The global minimum of $F_8(x)$ is equal to -418.9829 x 5.	

Table 2. Comparison of obtained values throughout 30 runs for unimodal functions.

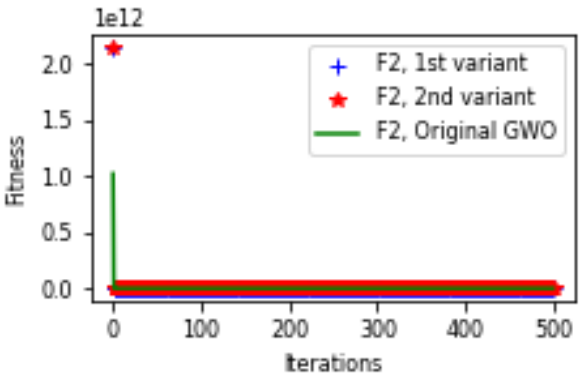
		GWO		Modified Variant 1		Modified Variant 2	
Function	**Criteria**	**Solution**	**CPU Time (seconds)**	**Solution**	**CPU Time (seconds)**	**Solution**	**CPU Time (seconds)**
1	**Min**	3.03E-38	13.59	**2.21E-38**	**10.873**	3.89E-34	17.446
	Max	9.78E-36	21.38	5.10E-36	21.731	6.17E-32	20.552
	Mean	1.11E-36	18.97	1.12E-36	17.479	1.94E-32	18.616
	Std.D.	1.98703E-36	1.81	1.56425E-36	3.071	1.87E-32	0.632
2	**Min**	1.42E-22	17.63	**1.05E-22**	16.969	5.68E-20	**14.376**
	Max	2.42E-21	22.27	4.36E-21	19.227	8.91E-19	21.028
	Mean	7.88E-22	20.26	8.48E-22	18.458	2.71E-19	17.440
	Std.D.	5.81E-22	1.35	8.3E-22	0.545	1.81E-19	1.485
3	**Min**	5.73E-10	28.24	**3.63E-11**	26.102	1.71E-09	**22.220**
	Max	1.71E-05	32.33	2.13E-06	29.720	4.55E-04	32.387
	Mean	1.05E-06	30.23	3.17E-07	28.078	1.73E-05	29.400
	Std.D.	3.6E-06	0.82	6.67E-07	0.858	8.27E-05	1.983
4	**Min**	**2.02E-09**	18.82	7.26E-09	**16.143**	2.16E-08	17.775
	Max	2.11E-07	19.56	1.95E-07	20.147	2.62E-06	20.517
	Mean	4.55E-08	19.13	5.66E-08	17.977	4.75E-07	19.024
	Std.D.	5.93E-08	0.17	5.52E-08	0.761	6.18E-07	0.652
5	**Min**	**2.53E+01**	**16.89**	2.56E+01	17.306	2.55E+01	19.346
	Max	2.85E+01	20.18	2.79E+01	19.328	2.88E+01	23.221
	Mean	2.64E+01	19.52	2.66E+01	18.802	2.63E+01	20.590
	Std.D.	0.749195	0.54	0.613892	0.532	0.585917	0.804
6	**Min**	2.20E-05	17.81	**1.77E-05**	17.039	2.72E-05	**16.124**
	Max	9.99E-01	19.53	9.40E-01	19.502	6.13E-05	20.968
	Mean	3.86E-01	19.20	3.90E-01	18.418	4.19E-05	18.791
	Std.D.	0.265205	0.29	0.271808	0.538	9.39E-06	0.986
7	**Min**	4.95E-04	20.21	3.23E-04	14.721	**3.04E-04**	**11.328**
	Max	4.09E-03	20.69	4.90E-03	22.536	3.51E-03	20.698
	Mean	1.45E-03	20.32	1.67E-03	19.743	1.61E-03	18.591
	Std.D.	0.000834	0.12	0.000984	1.399	0.000756	2.197

Table 3. Comparison of obtained values throughout 30 runs, for multi-modal functions.

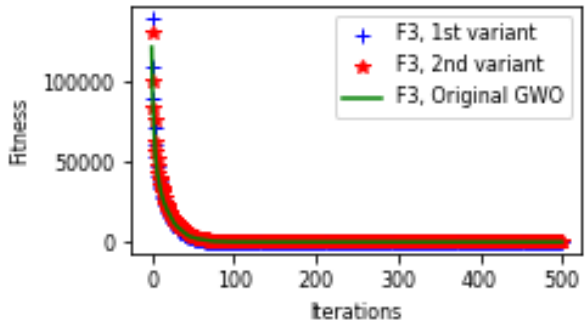
		GWO		**Modified Variant 1**		**Modified Variant 2**	
Function	**Criteria**	**Solution**	**CPU Time (seconds)**	**Solution**	**CPU Time (seconds)**	**Solution**	**CPU Time (seconds)**
8	**Min**	**-8745.72**	**15.93**	-7.63E+03	14.200	-7.62E+03	14.026
	Max	-3412.72	21.39	-3.23E+03	19.201	-2.93E+03	20.248
	Mean	-6501.88	19.11	-5.80E+03	17.681	-5.79E+03	18.545
	Std.D.	1047.851	1.24	1371.098	1.302	1657.309	1.330
9	**Min**	1.14E-13	**8.64**	**5.68E-14**	15.221	1.14E-13	18.048
	Max	68.41015	19.75	3.12E+01	18.992	2.99E+01	20.377
	Mean	9.990329	16.76	1.05E+01	18.146	6.54E+00	18.950
	Std.D.	16.67717	3.63	7.752831	0.759	7.582242	0.611
10	**Min**	**3.24E-14**	**6.47**	**3.24E-14**	16.219	3.95E-14	16.281
	Max	4.31E-14	23.96	5.02E-14	20.203	6.79E-14	20.826
	Mean	3.96E-14	19.02	3.91E-14	19.112	5.52E-14	18.626
	Std.D.	3.02E-15	4.56	3.82E-15	0.941	7.95E-15	1.323
11	**Min**	**0**	20.16	**0.00E+00**	**15.673**	**0.00E+00**	17.768
	Max	0.037617	25.02	1.74E-02	21.455	3.01E-02	21.586
	Mean	0.006641	22.16	2.79E-03	19.135	5.05E-03	19.864
	Std.D.	0.010885	0.95	0.005829	1.340	0.00916	0.944
12	**Min**	3.44E-06	23.14	5.37E-03	17.362	**2.32E-06**	**11.922**
	Max	0.203627	25.86	1.99E-01	23.131	8.00E-03	22.152
	Mean	0.073773	24.21	8.96E-02	21.309	1.56E-03	18.658
	Std.D.	0.059404	0.56	0.047675	1.366	0.002759	2.786
13	**Min**	0.100125	**16.31**	4.31E-05	18.036	**3.68E-05**	18.661
	Max	0.618225	25.99	8.09E-01	23.303	5.22E-01	23.574
	Mean	0.348725	23.42	3.73E-01	20.925	2.02E-01	22.009
	Std.D.	0.173681	1.76	0.219248	1.033	0.153377	0.965


F1, 1st variant
F1, 2nd variant
F1, Original GWO
Fitness
Iterations

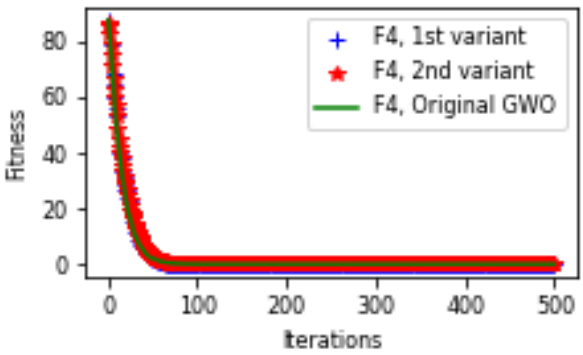
(a)


1e12
F2, 1st variant
F2, 2nd variant
F2, Original GWO
Fitness
Iterations

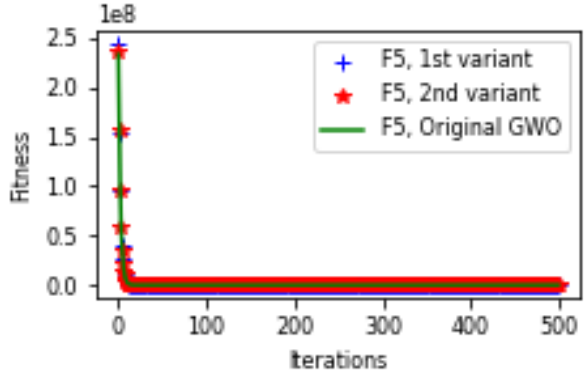
(b)


F3, 1st variant
F3, 2nd variant
F3, Original GWO
Fitness
Iterations

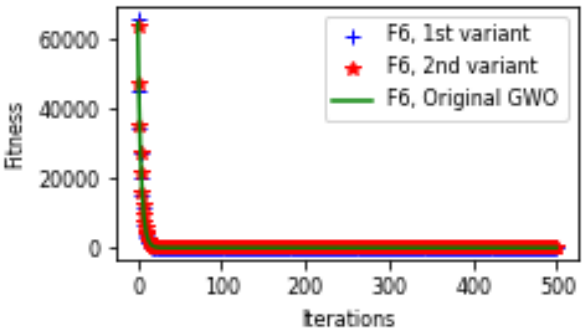
(c)


F4, 1st variant
F4, 2nd variant
F4, Original GWO
Fitness
Iterations

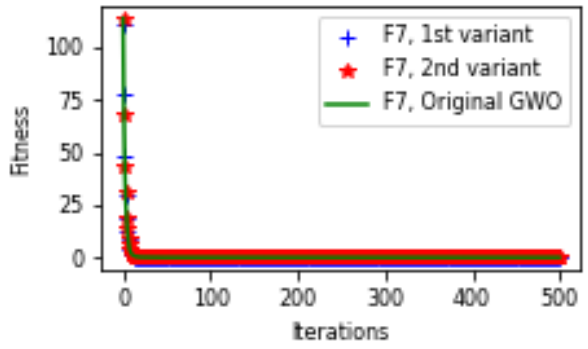
(d)


1e8
F5, 1st variant
F5, 2nd variant
F5, Original GWO
Fitness
Iterations

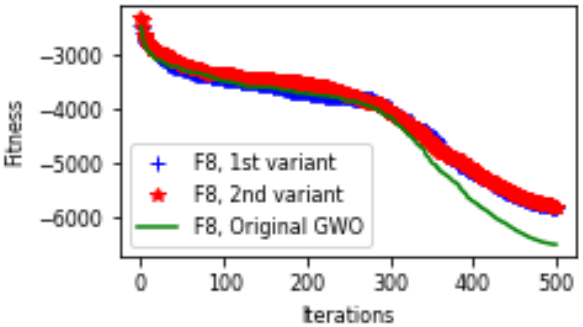
(e)


F6, 1st variant
F6, 2nd variant
F6, Original GWO
Fitness
Iterations

(f)


F7, 1st variant
F7, 2nd variant
F7, Original GWO
Fitness
Iterations

(g)


F8, 1st variant
F8, 2nd variant
F8, Original GWO
Fitness
Iterations

(h)

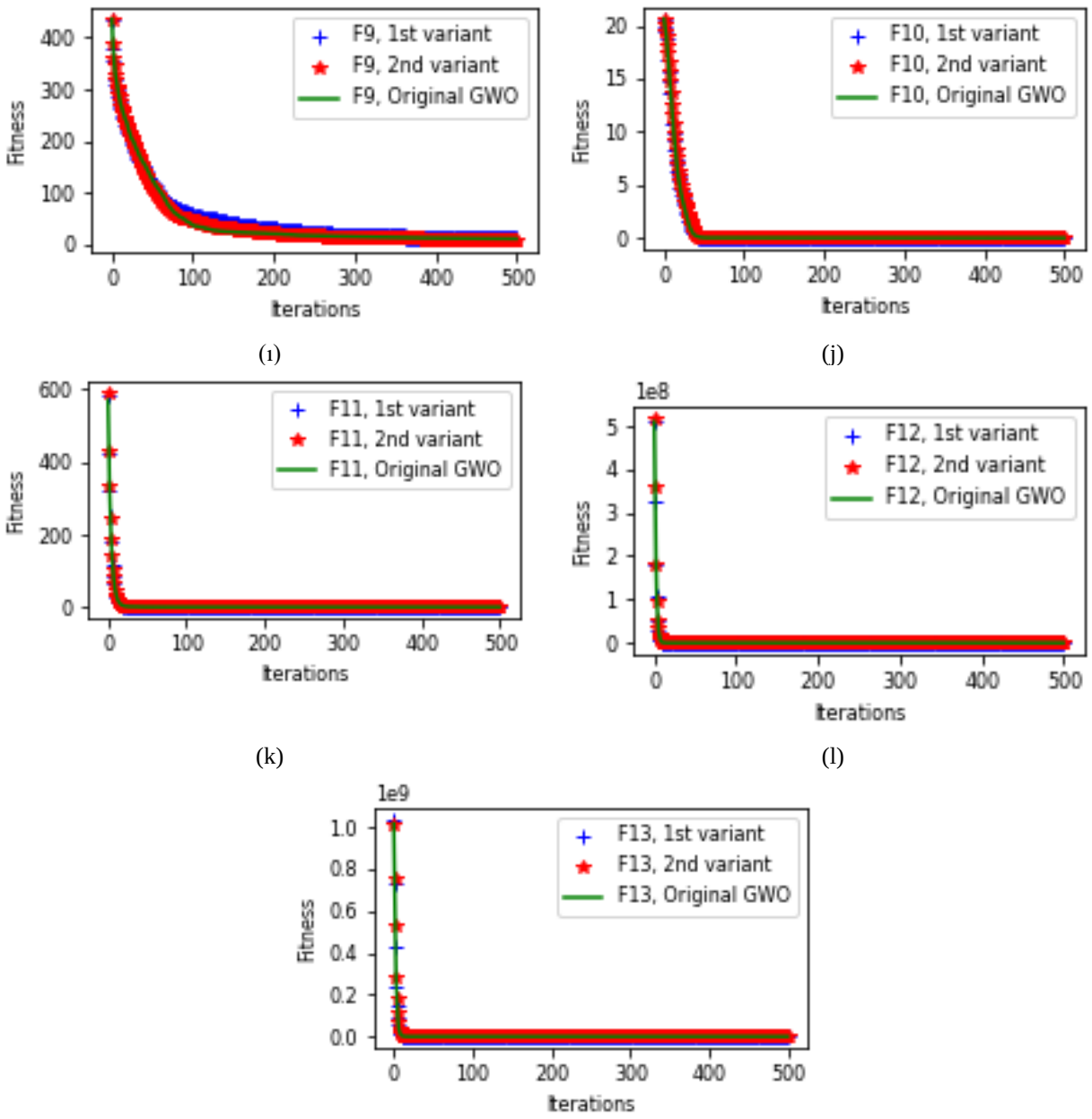


Figure 1: Convergence characteristics of the original GWO and proposed two variants through test functions.