15

Grasshopper Optimization Algorithm - Modifications and Applications

Szymon Łukasik

Faculty of Physics and Applied Computer Science
AGH University of Science and Technology, Kraków, Poland

CONTENTS

15.1 Introduction

The optimization technique known as Grasshopper Optimization Algorithm (GOA) was introduced by Saremi, Mirjalili and Lewis in 2017 [24]. It uses the well known concept of intelligent swarms [10]. From the methodological side it includes both pairwise interactions between swarm members, as well as the influence of the best individual of the swarm. The first is reported to stem from the interaction of grasshoppers which demonstrates itself through slow movements (while in larvae stage) and dynamic motion (while in insect form). The second corresponds to the tendency to move towards the source of food, with deceleration observed on the approach path. The algorithm was positively evaluated both by the authors of the concept (and its original scheme) as well

as researchers introducing the adaptation of the algorithm for a variety of the domains.

The rest of this paper is organized as follows: in Section 15.2 we provide a more detailed description of the basic scheme of the algorithm under-consideration, enclosing also the pseudo-code of its implementation. Section 15.3 reviews the modification proposed in the literature. Finally, in Section 15.4 we provide the results of our experiments with the clustering technique using GOA as the optimization engine.

15.2 Description of the original Grasshopper Optimization Algorithm

The original GOA, as presented in [24] constitutes a novel optimization algorithm dealing with continuous optimization, i.e. the task of finding a value of x – within the feasible search space $S \subset R^D$ – denoted as x^* such as $x^* = \text{argmin}_{x \in S} f(x)$, assuming that the goal is to minimize cost function f. GOA belongs to a class of population based metaheuristics [27]. Thus it uses the population comprising P agents to tackle the aforementioned problem. Each of them is represented as a solution vector $x_p,\ p = 1, ..., P$ and corresponds to exactly one solution in the domain of considered function f [15].

The movement of individual p in iteration k is described with the following equation:

$$x_{pd} = c \left(\sum_{q=1, q \neq p}^{P} c \frac{UB_d - LB_d}{2} s(|x_{qd} - x_{pd}|) \quad \frac{x_{qd} - x_{pd}}{dist(x_q, x_p)} \right) + x_d^* \tag{15.1}$$

where $d = 1, 2, ..., D$ represents dimensionality of the search space, and $dist(a, b)$ Euclidean distance between a and b. The equation (15.1) contains two components: the first line corresponds to the pairwise social interactions between grasshoppers, the second – to the movement attributed to the wind. GOA implement it by a shift towards the best individual. In contrast to the behavior of real grasshopper swarms [6] the algorithm does not contain a gravitation factor.

Function s represents the strength of social forces, and was originally defined as:

$$s(r) = f e^{\frac{-r}{l}} - e^{-r} \tag{15.2}$$

with default values of $l = 1.5$ and $f = 0.5$.

It divides the space between two considered grasshoppers into three separate zones. Individuals being very close are found in the so called repulsion

zone. On the other hand distant grasshoppers are located in the zone of attraction. The zone (or equilibrium state) between them – called the comfort zone – is characterized by the lack of social interactions. The first factor is additionally normalized by the upper (UB_d) and lower bounds (LB_d) of the feasible search space.

Parameter c – occurring twice in formula (15.1) – is decreased according to the following equation:

$$c = c_{max} - k\frac{c_{max} - c_{min}}{K_{max}} \tag{15.3}$$

with maximum and minimum values given by c_{max}, c_{min} respectively, and maximum number of iterations denoted as K_{max}. The first occurrence of c in (15.1) reduces the movements of grasshoppers around the target – balancing between exploration and exploitation of the swarm around it. The same concept is used in the Particle Swarm Optimization Algorithm and it is known as inertia weight [5]. The component $c\frac{UB_d - LB_d}{2}$ on the other hand linearly decreases the space that the grasshoppers should explore and exploit.

The pseudo-code for the generic version of GOA technique is presented in Algorithm 17.

Algorithm 17 Grasshopper Optimization Algorithm.

$k \leftarrow 1$, $f(x^*(0)) \leftarrow \infty$ ▷ initialization
for $p = 1$ to P **do**
  $x_p(k) \leftarrow$ Generate_Solution(LB, UB)
end for
▷ find best
for $p = 1$ to P **do**
  $f(x_p(k)) \leftarrow$ Evaluate_quality$(x_p(k))$
  if $f(x_p(k)) < f(x^*(k-1))$ **then**
    $x^*(k) \leftarrow x_p(k)$
  else
    $x^*(k) \leftarrow x^*(k-1)$
  end if
end for
repeat
  $c \leftarrow$ Update_c$(c_{max}, c_{min}, k, K_{max})$
  for $p = 1$ to P **do**
    ▷ move according to formula (15.1)
    $x_p(k) \leftarrow$ Move_Grasshopper$(c, UB, LB, x^*(k))$
    ▷ correct if out of bounds
    $x_p(k) \leftarrow$ Correct_Solution$(x_p(k), UB, LB)$
    $f(x_p(k)) \leftarrow$ Evaluate_quality$(x_p(k))$
    if $f(x_p(k)) < f(x^*k)$ **then**
      $x^*(k) \leftarrow x_p(k)$, $f(x^*k) \leftarrow f(x_p(k))$
    end if

end for
for $p = 1$ to P **do**
    $f(x_p(k+1)) \leftarrow f(x_p k)$, $x_p(k+1) \leftarrow x_p(k)$
end for
$f(x^*(k+1)) \leftarrow f(x^* k)$, $x^*(k+1) \leftarrow x^*(k)$
$k \leftarrow k+1$
until $k < K_{max}$ **return** $f(x^*(k))$, $x^*(k)$

For more details regarding the scheme of the original algorithm, as well as its practical implementation one could refer to [18].

15.3 Modifications of the GOA technique

Since its introduction in 2017 the Grasshopper Optimization Algorithm has attracted significant attention indicated by 221 citations in Scopus of the original paper introducing the algorithm up to this day (Sept. 4th 2019). Besides GOA applications – in its standard variant as described in the previous section – in a variety of fields numerous improvements and modifications have been suggested. It will be briefly presented in the subsequent parts of this section.

15.3.1 Adaptation to other optimization domains

As was observed in the enclosed description of GOA – it was originally designed for continuous optimization tasks. However the algorithm can be easily modified to be used for binary optimization. As an example two variants of GOA – presented in [19] – could be named. They involve using either transfer functions or stochastic mutation as an attraction operator. Both approaches proved to be highly competitive for feature selection problems considered in the aforementioned publication.

In addition to that the GOA scheme was also modified to allow the algorithm to tackle multi-objective optimization tasks [20]. In [21] a concept of archive – storing Pareto optimal solutions with additional elimination of solutions lying within a crowded neighbourhood – were introduced to form Multi-objective Grasshopper Optimization Algorithm (MOGOA). Its performance was tested for well known ZDT and CEC 2009 test suites. During those tests the algorithm was also compared with Multi-objective PSO, Multi-objective Dragonfly Algorithm, Multi-objective Ant Lion Algorithm and NSGA-II technique. It was concluded therein that MOGOA offers relatively high distribution across all objectives and quick convergence.

15.3.2 Structural modifications

Several modifications to the basic algorithm scheme were also proposed. They were primarily aimed to further improve its optimization capabilities.

First of all experiments involving Levy flight in the movement towards the best individual were conducted in [13]. Each solution vector is obtained by:

$$x_{pd}^{new} = x_{pd} * R * f_L(\beta) \tag{15.4}$$

with R representing a random value in the $[0, 1]$ interval and $f_L(\beta)$ random value for Levy flight with parameter β. It was demonstrated – for the image thresholding problem – that Levy flight can maximize the diversity of population exploring the search domain bringing a positive outcome in terms of final optimization result. Similarly an additional random component, in the form of random mutation was considered in [26] for the optimal power flow problem, also proving to be beneficial.

More fundamental modification was under investigation in [30] – for general continuous optimization tasks. The paper considers the introduction of dynamic weight modifying the impact of the first component of eq. (15.1) and creating three phases of search space exploration. In addition to that random jumps, for stagnating solutions are introduced. These two components significantly improve GOA performance in aspects of accuracy, stability, and convergence rate.

So called Adaptive GOA – proposed in [29] – seems to represent the most invasive modification of the basic algorithm's scheme. It encompasses natural selection strategy, three elite agents and dynamic adjustment of c, similar to the one presented in [30]. The authors demonstrated that such an approach significantly over-performs standard GOA over a set of target tracking problems.

Finally, the trend to use chaotic maps in metaheuristics finds its representation also for the Grasshopper Optimization Algorithm. Ten different chaotic maps were considered in [25] to form a class of Enhanced Chaotic Grasshopper Optimization Algorithms (ECGOAs). They are used to modify the value of parameter c . No definitive suggestions concerning the choice of the maps were derived, however the paper concludes with the general observation that such chaotic formulation of c brings very positive results. Similar – though more general – experiments were also conducted in [4]. Their outcome is a recommendation to use circle maps to improve GOA search capabilities. Such a map is obtained in this case with:

$$c^{new} = c + b - \frac{P}{2\pi}\sin(2\pi c) \mod (1). \tag{15.5}$$

for fixed values of parameters $P = 0.5$ and $b = 0.2$.

15.3.3 Hybrid algorithms

GOA was also considered as a component of hybrid algorithms – including the use of two and more heuristic approaches. The list of representative contribu-

tions in this field is not long however. In this context GOA was successfully hybridized with the Genetic Algorithm, to deal with generation of security keys [2]. Solutions from both GA and GOA are merged to improve search capabilities of the combined algorithm. Similarly a hybrid with Differential Evolution (DE) was proposed in [11]. It was aimed at solving the Multilevel Satellite Image Segmentation problem. DE is supposed to supplement GOA in the latter stages of optimization preserving the population diversity.

15.4 Application example: GOA-based clustering

15.4.1 Clustering and optimization

Clustering represents a task of dataset division into a set of C disjoint clusters (groups). Its goal is to find elements resembling each other – and to form clusters composed of them. Cluster analysis is used in diverse applications like automatic control [14], text analysis [1], agriculture [7] or marketing [22]. Several clustering algorithms were already presented in the literature of the subject [1]. Here we will demonstrate the use of the Grasshopper Optimization Algorithm to tackle the aforementioned problem.

Let us represent a dataset by data matrix Y of $M \times N$ dimensionality. Each of its N columns represents a feature characterizing data sample. Samples correspond to a row of the matrix, commonly known as the dataset's elements or cases.

Clustering remains an unsupervised learning procedure, frequently with a known number of clusters C being the only information available. Its task is to assign dataset elements $y_1, ..., y_M$ to clusters $CL_1, CL_2, ..., CL_C$. To validate the clustering solution one could compare its results with correct cluster labels. If they are unavailable so called internal validation assessing if the obtained solution reflects the structure of the data and natural groups which can be identified within its records [9] can be employed.

Using any heuristic optimization algorithm requires choosing proper solution representation. In the case of clustering it is natural to represent the solution as a vector of cluster centers $x_p = [u_1, u_2, ..., u_C]$. Consequently the dimensionality D used in the description of GOA, in the case of the data clustering problem, is equal to $C * N$ as each cluster center u_i is a vector of N elements.

Another important aspect is choosing the proper tool for assessing the quality of generated solutions. Here an idea already presented in [16] was implemented. After assigning each data element y_i to the closest cluster center the solution x_p (representing those centers) is evaluated according to the formula:

$$f(x_p) = \frac{1}{I_{CH,p}} + \#_{CL_{i,p}=\emptyset,\, i=1,...,C} \tag{15.6}$$

where the value $I_{CH,p}$ represents Calinski-Harabasz index calculated for solution x_p. It is generally obtained via:

$$I_{CH} = \frac{N-C}{C-1} \frac{\sum_{i=1}^{C} dist(u_i, U)}{\sum_{i=1}^{C} \sum_{x_j \in CL_i} dist(x_j, u_i)} \tag{15.7}$$

wheras $u_i \in R^N$ for non-empty cluster CL_i corresponds to the cluster center defined by:

$$u_i = \frac{1}{M_i} \sum_{y_j \in CL_i} y_j, \quad i = 1, ..., C \tag{15.8}$$

with M_i being cardinality of cluster i and – likewise – U corresponding to the center of gravity of the dataset:

$$U = \frac{1}{M} \sum_{j=1}^{M} y_j \tag{15.9}$$

To penalize solutions which do not include the desired number of clusters a penalty – equal to a number of empty clusters identified in the x_p clustering solution written in (15.6) as $\#_{CL_{i,p}=\emptyset,\, i=1,...,C}$ – is added.

The choice of this index was motivated by our successful experiments on other heuristic algorithms using I_{CH} value [12] as a key component and similar positive results presented in [3].

15.4.2 Experimental setting and results

Clustering as such is an unsupervised learning problem, therefore its evaluation requires an assumption about the availability of correct cluster labels. In practice a labeled dataset containing the information about assignment of data elements to classes can be used as a substitute.

A clustering solution understood as a set of cluster indexes provided for all data points should be then compared with a set of class labels. Such a comparison can be done with the use of the Rand index [23], external validation index which measures similarity between cluster analysis solutions. It is characterized by a value between 0 and 1. A low value of R suggests that the two clusterings are different and 1 indicates that they represent exactly the same solution – even when the formal indexes of clusters are mixed.

For computational experiments a set of standard synthetic clustering benchmark instances known as S-sets was used [8]. Figure 15.1 demonstrates the structure of those two dimensional data sets. It can be seen that they are characterized by different degree of cluster overlapping.

To make the study more representative six additional real-world problems, taken from UCI Machine Learning Repository were taken into consideration [28]. Table 15.1 characterizes all datasets used in this paper for experimental evaluation. For each instance it contains properties like dataset size M,

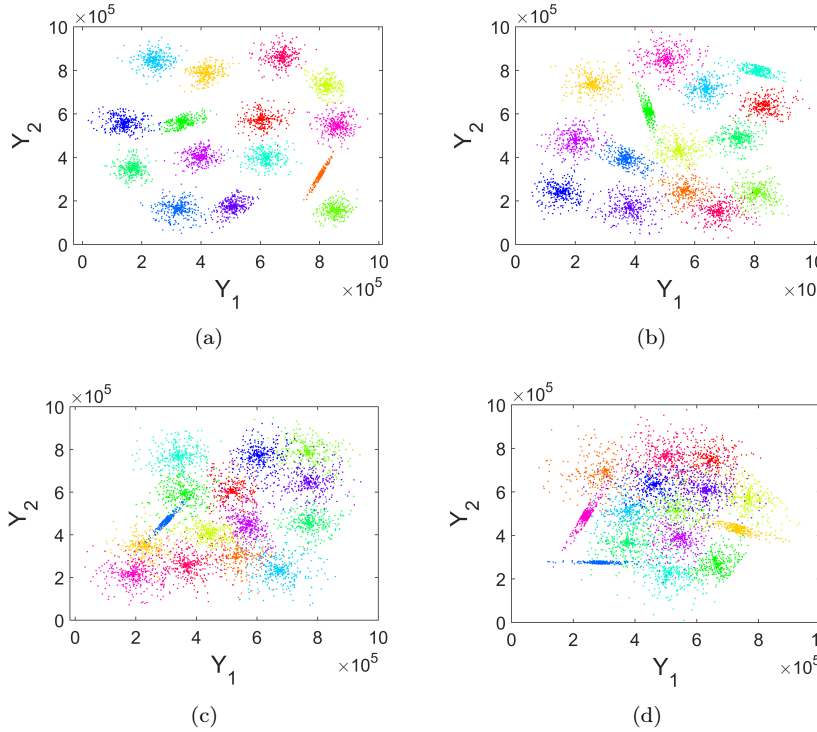

FIGURE 15.1
Scatter plots for *s1* (a), *s2* (b), *s3* (c) and *s4* (d) datasets.

TABLE 15.1
Experimental datasets description.

Dataset	Sample size	Dimensionality	No of clusters
s1	5000	2	15
s2	5000	2	15
s3	5000	2	15
s4	5000	2	15
glass	214	9	6
wine	178	13	3
iris	150	4	3
seeds	210	7	3
heart	270	13	2
yeast	1484	8	10

TABLE 15.2
K-means vs. GOA-based clustering.

	K-means clustering		GOA clustering	
	$\overline{R}$	$\sigma(R)$	$\overline{R}$	$\sigma(R)$
s1	0.980	0.009	0.990	0.006
s2	0.974	0.010	0.984	0.006
s3	0.954	0.006	0.960	0.005
s4	0.944	0.006	0.951	0.003
glass	0.619	0.061	0.643	0.035
wine	0.711	0.014	0.730	0.000
iris	0.882	0.029	0.892	0.008
seeds	0.877	0.027	0.883	0.004
heart	0.522	0.000	0.523	0.000
yeast	0.686	0.033	0.676	0.034

dimensionality N and the number of classes C – used as desired number of clusters for the grouping algorithms. For non-synthetic problems it was naturally assumed that each class manifests itself as a single cluster.

As a point of reference for evaluating performance of clustering methods classic K-means algorithm was used. It is also the case of this contribution.

To evaluate clustering methods they were run 30 times with mean and standard deviation values of Rand index – $\overline{R}$ and $\sigma(R)$ – being recorded. For the GOA-based algorithm a population of $P = 20$ swarm members was used. The algorithm terminates when $C * N * 1000$ cost function evaluations were performed. It is a standard strategy for evaluating metaheuristics – making the length of the search process dependent on data dimensionality.

We used default values of all GOA parameters, with $c = 0.00001$. It means that c quickly approaches values close to zero. The summary of obtained results for this case is provided in Table 15.2. It can be observed that GOA-based clustering outperforms K-means on the majority of the datasets – it is also less prone to getting stuck in local minima. It is indicated by the values of standard deviations, which in the case of K-means algorithm are significantly higher.

For more extensive study on the GOA application in clustering – including alternative values of c and chaotic variant of the algorithm – one could refer to [17].

15.5 Conclusion

This chapter has modified variants of of Grasshopper Optimization Algorithm. We have studied alternative GOA schemes including different optimization

setting, structural modifications and hybridization with other nature-inspired techniques. In addition to that the application of GOA for clustering task was also presented.

The algorithm in its basic form was more extensively discussed in the chapter: Grasshopper optimization algorithm [18]. Along with more precise description of its mechanics, the chapter also provides technical details of GOA implementation and walk-through GOA structural components.

References

1. C. Aggarwal and C. Zhai, "A survey of text clustering algorithms," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds. Springer US, 2012, pp. 77–128.
2. A. Alphonsa, M. M. Mohana, N. Sundaram, "A reformed grasshopper optimization with genetic principle for securing medical data". *Journal of Information Security and Applications*, vol. 47, pp. 410-420, 2019.
3. O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona, "An extensive comparative study of cluster validity indices," *Pattern Recognition*, vol. 46, no. 1, pp. 243 – 256, 2013.
4. S. Arora, P. Anand, "Chaotic grasshopper optimization algorithm for global optimization". *Neural Computing and Applications*, doi: 10.1007/s00521-018-3343-2, 2018.
5. J. C. Bansal, P. K. Singh, M. Saraswat, A. Verma, S. S. Jadon, and A. Abraham. "Inertia weight strategies in particle swarm optimization", *2011 Third World Congress on Nature and Biologically Inspired Computing*, Oct. 2011, pp. 633-640.
6. R.F. Chapman and A. Joern. "Biology of grasshoppers", A Wiley-Interscience publication, Wiley, 1990.
7. M. Charytanowicz, J. Niewczas, P. Kulczycki, P. A. Kowalski, S. Łukasik, and S. Żak, "Complete gradient clustering algorithm for features analysis of X-Ray images," in *Information Technologies in Biomedicine*, ser. Advances in Intelligent and Soft Computing, E. Pietka and J. Kawa, Eds. Springer Berlin Heidelberg, 2010, vol. 69, pp. 15–24.
8. P. Fränti and O. Virmajoki, "Iterative shrinking method for clustering problems," *Pattern Recognition*, vol. 39, no. 5, pp. 761 – 775, 2006.
9. M. Halkidi, Y. Batistakis, and M. Vazirgiannis, "On clustering validation techniques," *Journal of Intelligent Information Systems*, vol. 17, no. 2-3, pp. 107–145, 2001.

10. A. E. Hassanien and E. Emary. *Swarm Intelligence: Principles, Advances, and Applications*, CRC Press, 2018.
11. H. Jia, C. Lang, D. Oliva, W. Song, X. Peng, "Hybrid grasshopper optimization algorithm and differential evolution for multilevel satellite image segmentation". *Remote Sensing*, vol. 11, paper no 1134, 2019.
12. P. A. Kowalski, S. Łukasik, M. Charytanowicz, and P. Kulczycki, "Clustering based on the krill herd algorithm with selected validity measures," in *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Sept. 2016, pp. 79–87.
13. H. Liang, H. Jia, Z. Xing, J. Ma and X. Peng, "Modified Grasshopper Algorithm-Based Multilevel Thresholding for Color Image Segmentation," *IEEE Access*, vol. 7, pp. 11258–11295, 2019.
14. S. Łukasik, P. Kowalski, M. Charytanowicz, and P. Kulczycki, "Fuzzy models synthesis with kernel-density-based clustering algorithm," in *Fuzzy Systems and Knowledge Discovery, 2008. FSKD '08. Fifth International Conference on*, vol. 3, Oct. 2008, pp. 449–453.
15. S. Łukasik and P. A. Kowalski. "Study of flower pollination algorithm for continuous optimization", in *Intelligent Systems* 2014 (P. Angelov, K. T. Atanassov, L. Doukovska, M. Hadjiski, V. Jotsov, J. Kacprzyk, N. Kasabov, S. Sotirov, E. Szmidt, and S. Zadrozny, eds.), Springer International Publishing, 2015, pp. 451-459.
16. S. Łukasik, P. A. Kowalski, M. Charytanowicz, and P. Kulczycki, "Clustering using flower pollination algorithm and calinski-harabasz index," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, July 2016, pp. 2724–2728.
17. S. Łukasik, P. A. Kowalski, M. Charytanowicz, and P. Kulczycki. "Data clustering with grasshopper optimization algorithm", *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 71-74, 2017.
18. S. Łukasik. "Grasshopper Optimization Algorithm", *Swarm Intelligence Algorithms: A Tutorial* (A. Slowik, ed.), CRC Press, 2020.
19. M. Mafarja, I. Aljarah, H. Faris, A.I. Hammouri, A.M. Al-Zoubi, S. Mirjalili, "Binary grasshopper optimisation algorithm approaches for feature selection problems", *Expert Systems with Applications*, vol. 117, pp. 267-286, 2019.
20. S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah. "Grasshopper optimization algorithm for multi-objective optimization problems", *Applied Intelligence*, vol. 48, no. 4, pp. 805–820, 2018.

21. S. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems", *Applied Intelligence*, vol. 48, pp. 805–820, 2018.

22. H. Müller and U. Hamm, "Stability of market segmentation with cluster analysis - a methodological approach," *Food Quality and Preference*, vol. 34, pp. 70 – 78, 2014.

23. H. Parvin, H. Alizadeh, and B. Minati, "Objective criteria for the evaluation of clustering methods," *Journal of the American Statistical Association*, vol. 66, pp. 846?–850, 1971.

24. S. Saremi, S. Mirjalili, and A. Lewis. "Grasshopper optimisation algorithm: Theory and application", *Advances in Engineering Software*, vol. 105, pp. 30-47, 2017.

25. A.Saxena, R. Kumar, "Chaotic Variants of Grasshopper Optimization Algorithm and Their Application to Protein Structure Prediction". in: *Applied Nature-Inspired Computing: Algorithms and Case Studies*, N. Dey, A. Ashour, S. Bhattacharyya, (eds) Springer (Singapore), pp. 151–175, 2020.

26. M.A. Taher, S. Kamel, F. Jurado, M. Ebeed, "Modified grasshopper optimization framework for optimal power flow solution". *Electrical Engineering*, doi: 0.1007/s00202-019-00762-4, 2019.

27. X. S. Yang. *Nature-Inspired Optimization Algorithms*, Elsevier, London, 2014.

28. UCI Machine Learning Repository, `http://archive.ics.uci.edu/ml/`, access Sept 4, 2019.

29. J. Wu, H. Wang, N. Li, P. Yao, Y. Huang, Z. Su, Y. Yu, "Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm". *Aerospace Science and Technology*, vol. 70, pp. 497–510, 2017.

30. R. Zhao, H. Ni, H. Feng, X. Zhu, "A Dynamic Weight Grasshopper Optimization Algorithm with Random Jumping". In: S. Bhatia, S. Tiwari, K. Mishra, M. Trivedi (eds.) *Advances in Computer Communication and Computational Sciences*. Springer, Singapore, pp. 401–413, 2019.