ISSN (Print) : 2319-8613
ISSN (Online) : 0975-4024

LCMFO: An Improved Moth-Flame Algorithm for Combinatorial Optimization Problems

Heba AbdElhamid[#1], Ahmed Helmi[*2], Ibrahim Ziedan[#3]

[#] Computer and Systems Department, Faculty of Engineering Zagazig University
Computer and Systems Department - Faculty of Engineering - Zagazig University - Egypt
[1] softintel8@gmail.com
[3]ieziedan@gmail.com

[*] Computer and Systems Department. Faculty of Engineering Zagazig University
Computer and Systems Department- Faculty of Engineering - Zagazig University - Egypt
[2]amhm162@gmail.com

***Abstract*— Combinatorial optimization problems (COPs) are challenging class of problems in the field of optimization. Permutations are preferred as solution representation scheme in most cases. Metaheuristic techniques can be used to look for good solutions for COPs with low cost. Moth-flame algorithm (MFO) is one recent population-based metaheuristic technique for continuous optimization problems. In this work improvement of MFO when used to solve COPs is studied. An improved version of MFO (called LCMFO) where Lévy-flight function is used to prepare initial solutions is proposed. Also crossover functions of genetic algorithms are used together with the basic technique of MFO to generate new solutions. Both MFO and LCMFO are tested with travelling salesman problem (TSP) as one popular COP. Experimental results show that there is a notable improvement of about 20-40% in the quality of solutions found by LCMFO over MFO only.**

Keyword - Combinatorial Optimization Problems (COP), Moth-Flame Optimization (MFO), Crossover Function, Lévy-Flight Distribution, Travelling Salesman Problem (TSP)

I. INTRODUCTION

Combinatorial optimization problems (COPs) are NP-hard problems [1],[2]. It takes time of an exponential order for an exact algorithm to find the optimal solution [3]. Many real-world instances belong to such discrete optimization class such as travelling salesman problem (TSP) [1],[4],[5], assignment problem [6],[7], constraint satisfaction problem [8], knapsack problem [9], minimum spanning trees [10], scheduling problems [11], vehicle routing problem[12], and others. For COPs it requires enumerating the whole combinatorial search space in a brute force manner if only optimal solution is required and nothing else. The most famous COP in literature is probably TSP. In TSP, one has to make a tour of n cities, starting from a root city, passing by each city just once, then returning back to the root one. Solutions of TSP are best encoded as permutations of cities [13]. Here the optimal tour has the minimal total covered distance. Assuming that travels between any pair of cities are the same (i.e. symmetric TSP) and there is no constraint that may reject some tour then it easily noted that the size of the solution space is $\frac{(n-1)!}{2}$. This is considering the worst-case of TSP. However for large n, asymmetric TSP and TSP with constraints have a search space of size $\Omega\left(\left(\frac{n}{e}\right)^n\right)$. Moreover, going beyond the brute force method and using tree- based algorithms [13] (commonly called branch and x methods including branch and bound, branch and cut, and branch and price) an exponential cost may result.

Besides exact methods of COPs, approximation algorithms such as metaheuristic techniques [13] found a great interest of the research community to solve different classes of optimization problems in a reasonable time. However, there is no guarantee of reporting one optimal solution by a metaheuristic algorithm but finding a near-optimal solution in a reasonable time may be accepted. Almost all metaheuristic techniques depend on real values in the range (0,1) to encode solutions of the studied problem. Also finding a new neighbourhood of a current solution follows a systematic way during the search. Thus there is a difficulty facing metaheuristic methods when solving COPs exemplified in the different encoding schemes for solutions generated by an applied technique as well as the final solutions of the problem (i.e. solutions required by the cost function to calculate the fitness). Mapping from a real-valued solution to a permutation one degrades the quality of generated solutions regardless of how much the mapping methods are good. This decreases the convergence rate of applied algorithm. Also the ability to overcome local minima areas in the search space becomes limited. Therefore it is worthy to investigate how a metaheuristic technique is improved when solving a COP.

From the dense forest of metaheuristic techniques [14],[15]and many others. We pick the Moth-flame optimization (MFO) algorithm of Seyedali Mirjalili [16] to solve various instances of TSP is selected. MFO is a recent population based algorithm that proved to have a high performance in different problems [17]-[19]. Like many metaheuristic techniques, MFO is inspired from nature. It is based on the navigation method of moths in light. Generating new candidate solutions is going through a systematic way using a mathematical equation modeling the moths' orientation. One approach of improving the quality of solutions for COPs is hybridization between metaheuristics with other metaheuristics or metaheuristics with exact methods as was well classified by Talbi [20] and Blum [21].

In this work, we adopt a different point of view to improve the performance of a technique like MFO for solving TSP problem. Instead of calling another technique for help in improving the quality of solutions generated by the original algorithm, we examine a kind of transplantation of good operators in MFO algorithm itself to get an improvement during the search process. Next subsection is illustrating our proposed approach.

A. Main idea of the proposed algorithm

In order to increase the ability of investigating new areas in the search space we think of using a perturbation operator from a totally different approach like the genetic algorithm. A crossover function [22]-[25] is used to move from current solutions to next stage solutions together with the systematic equations of MFO. Emerging generated solutions by a crossover function helps to escape from local minima which are highly expected if we follow the same method to look for new solutions during each iteration of the algorithm. We also use Lévy-flight distribution function [26],[27] to reduce the effect of any probable bad initialization on the final reported solution. Lévy-flight function has proved effectiveness when used with many optimization algorithms [28]-[30]. In view of this, the proposed new technique is called Lévy-Crossover-MFO or LCMFO.

This paper is organized as follows. In section 2 the basic version of MFO algorithm is introduced. Section 3 sheds light on TSP. The proposed LCMFO technique is explained in section 4. Section 5 shows the experimental results of testing both MFO and LCMFO against standard TSP datasets. Finally Section 6 contains the paper conclusions

II. MOTH-FLAME OPTIMIZATION ALGORITHM

MFO is one recent member in the family of population-based metaheuristic techniques. As well introduced in [16], the main inspiration of the algorithm is the navigation method of moths at night. In MFO the candidate solutions are moths (i.e., moths form the search space) and the problem's variables are the position of moths in space. Best positions or best solutions so far are known as the flames. The set of moths is represented in a matrix as shown in Eq. 1 where n is the number of moths and d is the problem dimension. The corresponding fitness values for all moths are stored in an array as shown in Eq. 2. Flames are stored in a similar matrix as shown in Eq. 3. Also the array of Eq. 4 is used to store the corresponding fitness values of flames. The logarithmic spiral mechanism used to update moths with respect to a flame is illustrated by Eq. 5 that updates the i-th moth, Eq. 6 that indicates the distance between the i-th moth and j-th flame, and Eq. 7 determines the decrement rate of flames along the search depending on the maximum number of flames N, the maximum number of iterations T and current iteration number l. MFO has a few internal parameters that control the shape of the search path like a constant b and a random number t in the range$[-1,1]$. Fig. 1 summarizes the general steps of MFO (as given in [16]) which incorporates initialization of solution, updating solutions and the termination criteria.

$$M=\begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & \dots & m_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ m_{n,1} & m_{n,2} & \dots & \dots & m_{n,d} \end{bmatrix} \quad \text{Eq. 1}$$

$$OM=\begin{bmatrix} OM_1 \\ OM_2 \\ \vdots \\ OM_n \end{bmatrix} \quad \text{Eq. 2}$$

$$F=\begin{bmatrix} F_{1,1} & F_{1,2} & \cdots & \cdots & F_{1,d} \\ F_{2,1} & F_{2,2} & \cdots & \dots & F_{2,d} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ F_{n,1} & F_{n,2} & \cdots & \dots & F_{n,d} \end{bmatrix} \quad \text{Eq. 3}$$

$$OF=\begin{bmatrix} OF_1 \\ OF_2 \\ \vdots \\ OF_n \end{bmatrix} \quad \text{Eq. 4}$$

$$M_i = D_i.e^{bt} \cdot \cos(2\pi t) + F_j \quad \text{Eq. 5}$$

$$D_i = \left|F_j - M_i\right| \quad \text{Eq. 6}$$

$$Flame\_no = round(N - l * N - 1/T) \quad \text{Eq. 7}$$

Algorithm 1: Moth-flame Optimization (MFO)

```
Initialize MFO search parameters.
Initialize the population of moths.
Repeat until maximum number of iterations is reached
        Update flame number using Eq. 7.
        Calculate the fitness of initial moths.
        if iteration number == 1 then
            Sort the initial population of moths according
        to fitness.
            Update flames positions and their fitness.
        else
            Merge both previous population and flames
        according to fitness.
            Update flames positions and their fitness.
        end
        Update the positions of best flame so far and its fitness.
        Set the moths as the previous population.
        for all population of moths
            Calculate the distance to the corresponding
            flame using Eq. 6.
            Update moth positions with respect to its
            corresponding flame using Eq. 5.
        end
end
```

Fig. 1. General steps of Moth-flame algorithm

III. Traveling salesman problem (tsp)

This problem has various formulations in literature. Probably, the most famous one is representing the problem as a graph [31]. A non-directed graph representation means that there is symmetry in distance between each pair of connected cities. If distances differ when reversing the direction between any pair of cities then this is equivalent to a directed graph. Also, it is obvious to consider this problem as a permutation-based one as we are looking for the optimal order of cities.

Whatever the current dataset of the problem represents a directed or undirected graph, the solution of TSP can be always expressed as a permutation. The optimal solution is that one with a minimal length of a complete tour or a Hamiltonian cycle. TSP starts from a node called depot and salesman visits all cities (each city visited only once) and returns back to the starting node [32]. Adjacency matrix (square matrix) is used to represent TSP instances. Given n cities and the distance between each pair of cities d_{ij} where d_{ij}=d_{ji} (undirected graph). Distance between each pair of connected cities is Euclidean distance and is computed in Eq. 8.

$$d_{ij} = \sqrt{\left(x_i - x_j\right)^2 + \left(y_i - y_j\right)^2}\ , 1 \le i,j \le n \quad \text{Eq. 8}$$

Where $(x_i\ , y_i)$ are the dimensions of the i-th city in dataset. Let π denote a permutation of cities then the cost of a TSP solution as $\left(\pi := \pi_1\ , \pi_2, \pi_3, \ldots . \pi_n\right)$ is computed in the cost function $F(\pi)$[33] as given in Eq. 9:

$$F(\pi) = \sum_{i=1}^{n-1} d_{\pi_i,\pi_{i+1}} + d_{n,1} \quad \text{Eq. 9}$$

In Eq. 9, the distances between adjacent cities in the solution, i.e. (π_i, π_{i+1}) starting from (π_1, π_2) until(π_n, π_1), sum up to the total cost of the current feasible solution. Thus, the minimal $F(\pi)$ tells about the optimal solution.

IV. PROPOSED LÉVY –CROSS MOTH-FLAME OPTIMIZATION(LCMFO)

LCMFO used Lévy-flight function to produce a population of initial solutions. Lévy-flight function as suggested in [34] is implemented. Lévy-flights are another kind of random walks whose step lengths are drawn from Lévy distribution. Lévy-flights are more efficient than Brownian random walks in exploring a large-scale search space. This is due to the variance of Lévy flights [26]. We are also encoding a scheme to pass from a continuous space (real numbers) to a combinatorial space (a solution is expressed as a permutation of integer numbers [35]). Then fitness function is applied to generated permutations. We are using a crossover method as a perturbation operator to reproduce new solutions. Besides using the systematic way (or equations of update solutions) of MFO algorithm alone to generate new candidate solutions, crossover operator provides a totally different neighborhood generator. This makes a variance in solutions so as to increase exploration of the solution space for next generation [20],[36]. When applying crossover method to generated solutions by MFO we aim to escape from local minima that is highly expected due to following same steps for navigating search space. This leads to new promising areas in the solution space and allows for improvement of best solution so far.

A. Lévy – flight function

Lévy-flights are another kind of random walks whose step lengths are drawn from Lévy distribution [27],[37]. Step length is $0 < \beta < 2$ and Lévy-function is used to produce a set of candidate solutions as initial ones using gamma function as in Eq. 10.

$$\varphi=\left[\Gamma(1+\beta)x\sin\left(\pi x\frac{\beta}{2}\right)/\Gamma\left(\left(\frac{(1+\beta)}{2}\right)x\beta x\,2^{\frac{(\beta-1)}{2}}\right)\right]^{\left(\frac{1}{\beta}\right)} \qquad \text{Eq. 10}$$

B. Crossover Operation

Crossover operation means mating between a pair of solutions to generate a new pair [38]. A randomly selected cut point in parent solutions is determined and the tails of two parents are swapped to get new off-spring [39] as in Fig. 2.

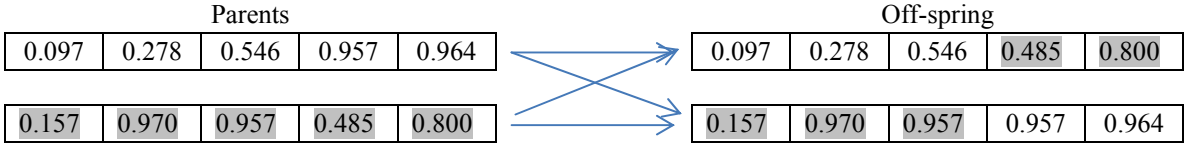

Fig. 2.Crossover Operation.

C. LCMFO Algorithm

Figure. 3 summarizes general steps of proposed LCMFO. Initial population is generated using Lévy-flight function. Then initial population of moths is sent to the fitness function that applies required mapping steps to obtain a permutation-based solution and calculates the fitness.

In the first iteration of the algorithm, moths are the generated initial population. This population is sorted to form the flames. Moths represent the so-called previous population in the next iteration. The best solution so far is determined as the best candidate in the flames. Lastly, positions of moths are updated according to the corresponding flames. Next iteration starts by updating population size in terms of reducing the number of flames for the purposes of convergence. After that a crossover function is applied to flames with a predetermined probability $P > 0$ which decides whether to generate a third population or not. In the crossover function, a random value r in $(0,1)$ is generated each iteration and if $r > P$ then crossover operation is taking place between each two consecutive flames. The third generated population in the search travel is called cross-flames. Now, a triple-population matrix of previous population, flames and cross-flames is sorted according to fitness. Only the top population can survive to be the new flames. The best solution so far is updated to the top one of the flames. Updated moths from first iteration become the previous population. And so on until the number of search iterations is exhausted.

Algorithm 2:Lévy-Cross Moth-flame Optimization (LCMFO)

```
Initialize LCMFO search parameters.
Initialize a population of moths using Lévy-flight using Eq. 10.
Calculate the fitness of initial population of moths.
Repeat until the maximum number of iterations is reached
Update flame number using Eq. 7.
if iteration number == 1 then
        Sort initial population of moths according to fitness.
    Update flames positions and their fitness.
else
  Run out crossover function with probability P on flames to get cross-flames.
  Merge previous population, flames and cross-flames according to fitness.
    Update flames positions and their fitness.
end
Update positions of best flame obtained so far and its fitness.
Set the moths as the previous population.
   for all moths in current population
     Calculate the distance to the corresponding flame using Eq. 6.
     Update the moth positions with respect to its corresponding flame using Eq. 5.
   end
end
```

Fig. 3.Main steps of LCMFO optimization algorithm

TABLE I. Parameters Settings

Parameter	Max_Iteration	Pop_size	lb	ub	P	No. of Runs	β
Parameter value	500	500	-10	10	0.01	10	1.5

TABLE III. Experimental Results

Data set	**Technique**	**Average**		**Best**	
		Value	**Improvement ratio %**	**Value**	**Improvement ratio %**
Ulysses 16	MFO	171.4052	19.97	147.7626	10.66
	LCMFO	137.1812		132.0055	
Bayg 29	MFO	19099	43.05	17631.3401	46.34
	LCMFO	10876		9460.2486	
St 70	MFO	2024.7	34.12	1820.0815	34.57
	LCMFO	1333.8		1190.9025	
Gr 96	MFO	3172.5	39.88	2787.7799	42.08
	LCMFO	1907.2		1614.6946	
Ch 150	MFO	34002	24.96	32322.0798	29.93
	LCMFO	25516		22649.1671	
Gr 202	MFO	11076	20.47	10196.2219	19.73
	LCMFO	8808.4		8184.6954	

V. EXPERIMENTS AND RESULTS

MFO and LCMFO were tested for different TSP datasets ranging from 16 to 202 dimensions [40]. TABLE I refers to the settings of search parameters. All datasets run in experiments for 500 iterations (denoted as Max_iterations in TABLE I) and a population of search agents of size 500 (denoted as POP_size in TABLE I). Real values of a solution were generated between two bounds, namely lower bound of -10 and upper bound of 10 (denoted as lb and ub in TABLE I respectively). A crossover probability of 0.01 was used (denoted as *P* in TABLE I) and the step length of Lévy distribution β was set to 1.5. Each experiment is repeated for 10 of computer runs.

Fitness value of average of solutions and best solution in each run were reported as shown in TABLE IIIIV (columns Average and Best respectively). The improvement ratio was calculated as$1 - \frac{fitness\ (best\ solution\ by\ LCMFO)}{fitness\ (best\ solution\ by\ MFO)}$. Results show a notable improvement in the quality of reported solutions when using the crossover operator. The improvement ratio obtained on average by LCMFO ranges between about 20% to 40% for different dimensions. Also, similar improvement ratio was registered for best reported solution by LCMFO over all runs except for the dataset Ulysses 16 that is an instance of a problem with dimension 16. Of course the distances matrix for each dataset plays a crucial role of determining feasible and unfeasible permutation solutions. Thus we don't expect that there should be a relationship between problem dimension and reported improvement ratio.

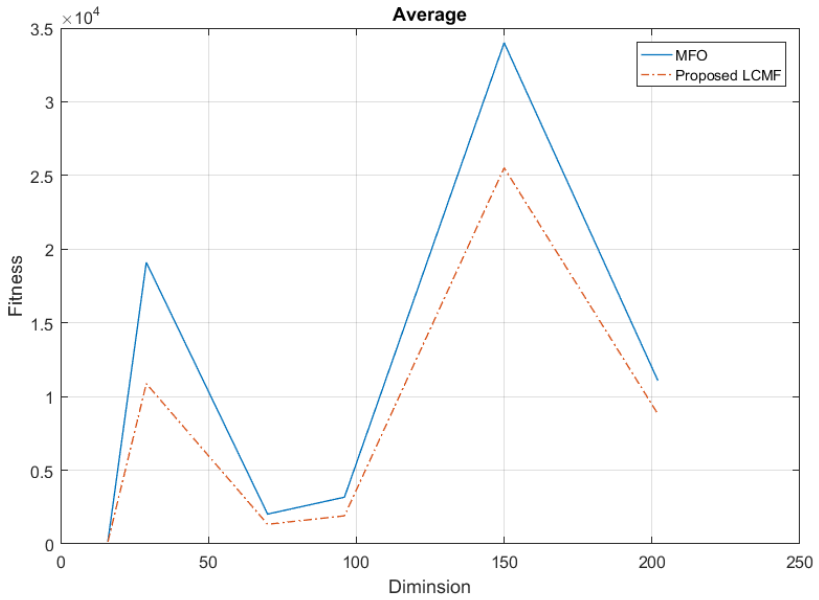

Fig. 4 Fitness of average solution of LCMFO and MFO

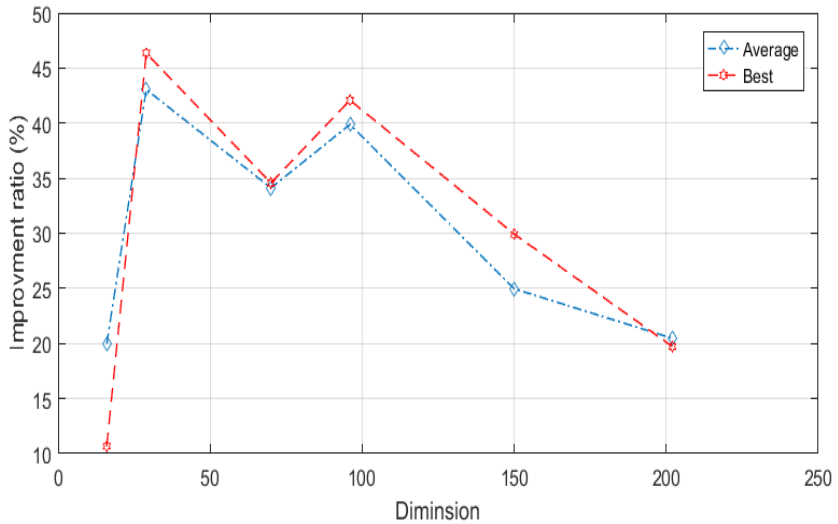

Fig .5 Improvement ratios for Average and Best

Reported results in TABLE VVI were also graphically plotted for more clarification about LCMFO algorithm behaviour. Fig. 4 shows a comparison between absolute fitness values of the average of solutions obtained by both techniques (column Value of Average in TABLE VIIVIII). Fig. 5 shows a plot of improvement ratio for both average of solutions and best solution (columns Improvement ratio of Average and Best in TABLE IXX) by LCMFO algorithm. Such figures also tell that the proposed algorithm behaves reasonably where the dataset with minimum improvement ratio on average has also the minimum improvement ratio for its best solution (namely Ulysses 16). Same observation holds for that dataset with maximum reported improvement ratio (namely Bayg 29).

VI. CONCLUSION

Although LCMFO algorithm decreased the speed of convergence rate of MFO yet there is still a change in solution with the increase of the number of iterations. LCMFO leads to obtaining more efficient solutions which are more close to the optimal one. In other words, LCMFO algorithm reduces the gap between quality of reported solutions and optimal one on the cost of speed of convergence. But this result may be considered quite fair regarding the difficulty of the problem under study like TSP. Moreover, mapping form continuous space to discrete space increase the challenge of applied search technique. It is obvious that the quality of an obtained continuous-values solution degrades after mapping it to a discrete form. It will be motivating to investigate other permutations-based problems in the class of NP-hard problems using LCMFO algorithm. Also, testing LCMFO with other mapping techniques (from continuous values into permutations) is of interest.

REFERENCES

[1] G. Gutin and A. P. Punnen, The traveling salesman problem and its variations vol. 12: Springer Science & Business Media, 2006.
[2] C. Walshaw, "A multilevel approach to the travelling salesman problem," Operations research, vol. 50, pp. 862-877, 2002.
[3] G. J. Woeginger, "Exact algorithms for NP-hard problems: A survey," in Combinatorial Optimization—Eureka, You Shrink!, ed: Springer, 2003, pp. 185-207.
[4] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, The traveling salesman problem: a computational study: Princeton university press, 2006.
[5] E. L. Lawler, J. K. Lenstra, A. R. Kan, and D. B. Shmoys, The traveling salesman problem: a guided tour of combinatorial optimization vol. 3: Wiley New York, 1985.
[6] E. M. Loiola, N. M. M. de Abreu, P. O. Boaventura-Netto, P. Hahn, and T. Querido, "A survey for the quadratic assignment problem," European journal of operational research, vol. 176, pp. 657-690, 2007.
[7] J. M. Kurtzberg, "On approximation methods for the assignment problem," Journal of the ACM (JACM), vol. 9, pp. 419-439, 1962.
[8] V. Kumar, "Algorithms for constraint-satisfaction problems: A survey," AI magazine, vol. 13, p. 32, 1992.
[9] J. J. Bartholdi, "The knapsack problem," in Building Intuition, ed: Springer, 2008, pp. 19-31.
[10] B. Y. Wu and K.-M. Chao, Spanning trees and optimization problems: CRC Press, 2004.
[11] S. Brah, J. Hunsucker, and J. Shah, "Mathematical modeling of scheduling problems," Journal of Information and Optimization Sciences, vol. 12, pp. 113-137, 1991.
[12] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo, "Vehicle routing," Handbooks in operations research and management science, vol. 14, pp. 367-428, 2007.
[13] M. Mehdi, "Parallel hybrid optimization methods for permutation based problems," University of Luxembourg, Luxembourg, Luxembourg, 2011.
[14] L. D. Whitley, T. Starkweather, and D. A. Fuquay, "Scheduling problems and traveling salesmen: The genetic edge recombination operator," in ICGA, 1989, pp. 133-40.
[15] K. Helsgaun, "General k-opt submoves for the Lin–Kernighan TSP heuristic," Mathematical Programming Computation, vol. 1, pp. 119-163, 2009.
[16] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," Knowledge-Based Systems, vol. 89, pp. 228-249, 2015.
[17] B. S. Yıldız and A. R. Yıldız, "Moth-flame optimization algorithm to determine optimal machining parameters in manufacturing processes," Materials Testing, vol. 59, pp. 425-429, 2017.
[18] S. Said, A. Mostafa, E. H. Houssein, A. E. Hassanien, and H. Hefny, "Moth-flame Optimization Based Segmentation for MRI Liver Images," in International Conference on Advanced Intelligent Systems and Informatics, 2017, pp. 320-330.
[19] H. Buch, I. N. Trivedi, and P. Jangir, "Moth flame optimization to solve optimal power flow with non-parametric statistical evaluation validation," Cogent Engineering, vol. 4, p. 1286731, 2017.
[20] E.-G. Talbi, Metaheuristics: from design to implementation vol. 74: John Wiley & Sons, 2009.
[21] C. Blum and A. Roli, "Hybrid metaheuristics: an introduction," in Hybrid Metaheuristics, ed: Springer, 2008, pp. 1-30.
[22] A. Umbarkar and P. Sheth, "CROSSOVER OPERATORS IN GENETIC ALGORITHMS: A REVIEW," ICTACT journal on soft computing, vol. 6, 2015.
[23] D. E. Goldberg, Genetic algorithms: Pearson Education India, 2006.
[24] J. Carr, "An introduction to genetic algorithms," Senior Project, vol. 1, p. 40, 2014.
[25] R. L. Haupt, S. E. Haupt, and S. E. Haupt, Practical genetic algorithms vol. 2: Wiley New York, 1998.
[26] X.-S. Yang, T. Ting, and M. Karamanoglu, "Random walks, Lévy flights, Markov chains and metaheuristic optimization," in Future information communication technology and applications, ed: Springer, 2013, pp. 1055-1064.
[27] Z. Li, Y. Zhou, S. Zhang, and J. Song, "Lévy-flight moth-flame algorithm for function optimization and engineering design problems," Mathematical Problems in Engineering, vol. 2016, 2016.
[28] Y. Ling, Y. Zhou, and Q. Luo, "Lévy Flight Trajectory-Based Whale Optimization Algorithm for Global Optimization," IEEE access, vol. 5, pp. 6168-6186, 2017.
[29] A. Iglesias, A. Gálvez, P. Suárez, M. Shinya, N. Yoshida, C. Otero, et al., "Cuckoo Search Algorithm with Lévy Flights for Global-Support Parametric Surface Approximation in Reverse Engineering," Symmetry, vol. 10, p. 58, 2018.
[30] X.-S. Yang, "Firefly algorithm, Levy flights and global optimization," in Research and development in intelligent systems XXVI, ed: Springer, 2010, pp. 209-218.

[31] M. Ahmadvand, M. Yousefikhoshbakht, and N. M. Darani, "Solving the traveling salesman problem by an efficient hybrid metaheuristic algorithm," Journal of Advances in Computer Research, vol. 3, pp. 75-84, 2012.
[32] A. Gupta and S. Khurana, "Study of traveling salesman problem using genetic algorithm," vol, vol. 2, pp. 575-588, 2012.
[33] R. E. Burkard, V. G. Deineko, and G. J. Woeginger, "The travelling salesman problem on permuted monge matrices," Journal of combinatorial optimization, vol. 2, pp. 333-350, 1998.
[34] İ. Aydoğdu, A. Akın, and M. P. Saka, "Design optimization of real world steel space frames using artificial bee colony algorithm with Levy flight distribution," Advances in Engineering Software, vol. 92, pp. 1-14, 2016.
[35] A. Ouaarab, B. Ahiod, and X.-S. Yang, "Random-key cuckoo search for the travelling salesman problem," Soft Computing, vol. 19, pp. 1099-1106, 2015.
[36] P. K. Yadav and N. Prajapati, "An overview of Genetic algorithm and modelling," International Journal of Scientific and Research Publications, vol. 2, pp. 1-4, 2012.
[37] M. Gutowski, "L\'evy flights as an underlying mechanism for global optimization algorithms," arXiv preprint math-ph/0106003, 2001.
[38] J. Magalhaes-Mendes, "A comparative study of crossover operators for genetic algorithms to solve the job shop scheduling problem", WSEAS transactions on computers, vol. 12, pp. 164-173, 2013.
[39] K. Borna and V. H. Hashemi, "An improved genetic algorithm with a local optimization strategy and an extra mutation level for solving traveling salesman problem," arXiv preprint arXiv:1409.3078, 2014.
[40] C. Sur, S. Sharma, and A. Shukla, "Solving travelling salesman problem using Egyptian vulture optimization algorithm–a new approach", in Language Processing and Intelligent Information Systems, ed: Springer, 2013, pp. 254-267.