RESEARCH ARTICLE OPEN ACCESS

Cooperative Discrete Firefly Algorithm to Solve the Traveling Salesman Problem

Abdulqader M. Mohsen [1], Wedad Al-sorori [2], Abdullatif Ghallab [3]
[1],[2],[3] University of Science and Technology, Sana'a - Yemen

ABSTRACT
Most real-world optimization problems consist of huge num-ber of variables that need unreasonable time and resources to identify the best solutions. One of the most important benchmark problem is the traveling salesman problem (TSP) which represents a large number of real-world optimization problems. Many exact methods and meta-heuristics have tried to solve TSP but each of them has its limitations in_nding the optimal solution. As a recent meta-hueristic, the Firey algo-rithm (FA) has successfully solved a variety of optimization problems. In this research, we propose a new enhanced cooperative variant of discrete FA, named CDFA. The key aim of CDFA is to overcome the limitations of the basic FA, such as trapping in local optima and premature conver- gence. Comparative tests employing thirteen TSP benchmark problems from TSBLIB are used to verify the validity of CDFA. In the majority of cases, CDFA surpasses state-of-the-art discrete _rey algorithms, such as EDFA and MDFA, in terms of average and best solutions.

***Keywords*:** - Traveling Salesman Problem, Discrete Firey Algorithm, Parallel Firey Algorithm, Crossover Operator, Diversi_cation and Intensi_cation.

I. INTRODUCTION

In 2008, Yang introduced a new meta-heuristic algorithm, named firefly algorithm (FA), which imitates the process of flashing light from tropical fireflies that occurs as a communication system to either attract mating partners or to warn possible predators. FA was initially developed as a meta-heuristic technique to solve continuous optimization problems. FA has demonstrated its ability to be employed in a wide range of domains and applications. As a result of FA's attractive success, a number of researchers have attempted to enhance the basic form of FA through a number of interventions, as stated in surveys [1, 2].

Furthermore, the FA method was improved in a variety of ways, including hybridization and parallelization. The majority of the enhancement effort is focused on combining FA with different methods. In compared to other previously proposed methods, combining FA with other algorithms increases the probability of obtaining a high-quality solutions. Thus, FA has been hybridized with many approaches to tackle continuous optimization problems. For example, FA was used with the Lagrangian approach to update the Lagrangian multipliers as a new way to address the unit commitment problem. Farahani et al. combined FA with Learning Automata for parameter adapting and Genetic algorithm for improving global search to solve continues numerical optimization problem [3]. To solve complex nonlinear problem, the standard FA combined with the evolutionary operations of Differential Evolution method [4]. A hybrid swarm model for microarray data to predict cancer was also proposed using FA and ACO to solve multimodal optimization problems [5]. Yang in [6] proposed the Eagle Strategy, a novel meta-heuristic search approach that combines the Levy flight search with the FA to address numerical optimization problems.

FA proves to be a promising combinatorial problem solver such as vehicle routing problem (VRP), scheduling and TSP. For VRP, FA was hybridized firstly with local search methods, and then with crossover and mutation operations [7]. In addition, a cooperative version of the previous research was proposed [8]. To improve its capability to address these types of problems, FA was hybridized with a variety of techniques. For example, to solve the graph 3-coloring problem, a local search heuristic was integrated with FA [9]. FA was also combined with local search to address the problem of permutation flow shop scheduling [10]. Horng used LBG method for FA initialization to build the vector quantization algorithm [11]. Another hybrid FA was developed to solve the monoalphabetic substitution cipher utilizing genetic operators including crossover and mutation [12]. To solve TSP, many hybridized variants of FA were proposed. For instance, Jati and Suyanto developed an evolutionary discrete firefly algorithm (EDFA) [13], that incorporated evolutionary mutation and selection but was also stuck in local optima in some instances. In addition, FA was merged with a greedy technique by SAraei et al. [14]. Although its good result, it has a drawback of taking a huge time to obtain the optimal solution due to the repeated cycle of greedy mutation jump. FA was integrated with the k-opt method and the multiple population approach by Zhou (MDFA) [15, 16]. In terms of convergence speed and solution quality, the findings revealed that MDFA outperformed EDFA. But It may still be trapped into local optima in spite of using small size instances for validating its performance. If we look at the FA, we may point to the study that was just proposed in [17]. The authors of this work provided the modified FA to improve its convergence

and search performance. To do the modifications, FA was hybridized with 3-opt and 2-opt methods, and then with crossover and mutation operations. [18] also introduced a hybrid method in which an FA is paired with a GA. in order to prevent the algorithm from sliding into local optimums, the distance of the FA was redefined by proposing a swap sequence and a swap operator. [19] presents a more detailed analysis that tried to address an extension of the TSP that allows more than one salesman to be utilized in the solution which known as the Multiple TSP. [20] proposes a new variant of swap-based FA hybridized with deferent methods including Fixed Radius Near Neighbor 2-opt operator, Nearest Neighborhood initialization, a movement strategy and reset method. [21, 22] are two more remarkable and useful studies. The first utilized the neighborhood search algorithm's dynamic mechanism, while the second uses the k opt method. [23, 24] are two more recent works that focus on the FA application.

For parallelization, there were only two researches that parallelized continuous FA [25, 26]. In [25] a GPU-based FA was proposed with a fixed-interaction distance and a uniform-grid acceleration data structure that was parallelized for multi-modal functions. FA was also parallelized in [26] in attempt to address an unconstrained continuous optimization problem.

A new cooperative discrete variation of FA was proposed in this paper to solve TSP. the work has three main contributions, which are briefly explained as follows. Firstly, to accelerate FA convergence toward the optimal solution, FA was combined with 2-opt and 3-opt. Secondly, the crossover operator is used in the second step to ensure that the current search space is efficiently exploited. The third method involves using parallelization principles of GA island models in order to preserve diversity and prevent being stuck in local minima.

The rest of this work is arranged in the following manner. The fundamental FA, discrete FA, and crossover operations are briefly discussed in Section 2. Section 3 describes our parallel discrete version of FA for the TSP (CDFA). The findings of numerical tests on a set of benchmark instances of TSP selected from the TSPLIB library were detailed explained and discussed in Section 4. The conclusion and future work are presented in Section 5.

II. PRELIMINARIES

A brief revision of the basic FA, its discrete variant and the crossover operation is introduced in the following subsections.

A. Firefly Algorithm2.1 Firefly Algorithm

Xin-She Yang developed FA, a population-based meta-heuristic algorithm, at Cambridge University in late 2007 and early 2008 [27, 28]. FA imitates a nature phenomenon of fireflies' flashing light. The flashing pattern plays an important role to serve these fireflies to accomplish different tasks such as communication and attracting prey. The light intensity I, which is inversely proportional to the square of the distance r 2 between two fireflies, is calculated using a physical formula. This formula taking into accounts the light absorption by the medium, which causes the light to become weaker as the distance between the two fireflies grows.

FA was designed by utilizing the following three principles for idealizing the flashing properties of fireflies: i) Each firefly, regardless of its sex, is attracted to other fireflies. ii) Attractiveness β is proportional to light intensity (brightness) I, even though they're inversely proportional to distance. As a result, any flashing fireflies will be attracted (moved) to the brighter (best) one. If there isn't a brighter firefly nearby, the current one will relocate at random. iii) The fitness function's landscape determines the brightness of a firefly.

The major steps of the basic FA are given below:

Step1. (Initialization of Fireflies): To assure the diversity of the solution, a population of fireflies is initialized with random values in this step.

Step 2. (Fitness Evaluation of Fireflies): According to the problem under consideration, the light intensity I is calculated and the fitness function of each firefly in the population is evaluated. For each firefly i, the solution is x_i and its light intensity I_i is proportional to the fitness function value I(xi) $\propto$ f(xi). Equation 1 shows how I is calculated.

$$I = I_0 exp - \gamma r^2 \quad (1)$$

where I_0 is the source's light intensity and gamma is a preset light absorption coefficient that is used to estimate the medium's light absorption.

Step 3. (Updating the Fireflies): According to their attraction, each firefly i goes toward another brighter (better) one j, resulting in the formation of a new brighter firefly position (solution). The attractiveness of fireflies is related to the intensity of their light I, as illustrated by Equation 2:

$$\beta = \beta_0 exp - \gamma r^2, \quad (2)$$

where β_0 denotes attractiveness at a distance of r = zero. While the traveling distance, of a firefly from current location i to the new location j, is calculated as indicated in Equation 3:

$$x_i^{t+1} = x_i^t + \beta_0 exp - \gamma r_{ij}^2 (x_j^t - x_i^t) + \alpha \epsilon_i^t, \quad (3)$$

where x_i^t represents the firefly position i at the previous iteration t and the attractiveness at distance r = zero represented by β_0, γ is the coefficient of the light absorption represented as a scaling factor, α represents a randomization parameter which reduced gradually to ensure that the algorithm will converge properly, ϵ_i^t indicates a vector generated as Gaussian distribution random numbers and r_{ij} is the Cartesian distance at positions i and j between the two fireflies x_i and x_j, that is calculated according to Equation 4:

$$r_{ij} = \|x_i - x_j\| = \sqrt{\sum_{k=1}^{k=n} (x_{ik} - x_{kj})^2}, \quad (4)$$

Step4. (Determining the brightest firefly): Determination of the current brightest firefly x* in the population at iteration t based on the fitness function $f(x_i)$ is conducted in this step.

Step5. (Checking of the termination criterion): Checking if the maximum number of iterations is reached or the optimum solution is obtained, returning the brightest firefly x* as the global best solution; if not the Steps 2, 3 and 4 are repeated.

B. Discrete variant of FA for TSP

As previously stated, FA cannot be used to solve discrete problems directly. Because TSP is a discrete problem, a population of fireflies is initialized as a two dimensional array (N*M). The number of fireflies (solutions) is represented by N where the elements of solution (cities) are represented by M. The TSP solution is a permutation of cities in the population represented by a firefly. In a given tour (solution), each firefly element represents a city.

To adapt FA to solve TSP, Zhou et al. [16] combined FA with the k-opt technique, which simulates firefly movement. The hamming distance, which represents the number of non-corresponding elements in the solution, was used to determine the distance between any two fireflies positions i and j.

TSP's movement philosophy is determined by the order in which cities appear in the solution vector. By modifying the order of cities in that solution vector, different solutions may be formed. The 2-opt move and 3-opt move strategies may be used to shift the order of cities, similarly to the MDFA [16], improved hybrid discrete bat algorithm (IHDBA) [29], multi-population discrete bat algorithm (MPDBA) [30], and improved bat algorithm (IBA) [31]. Firefly movement is represented by this alteration. As a result, during iteration t, each firefly x_iwill travel from one place i-1 to a new place i according to either 2-opt move or 3-opt move as described in Equations 5 and 6:

$$x_i^t = 2 - opt\,(x_i^t - 1), \quad (5)$$

$$x_i^t = 3 - opt\,(x_i^t - 1), \quad (6)$$

The value of the firefly's light intensity is controlling this change. The light intensity I in this version of FA is calculated using Equation 7:

$$I_i = random\left(1, r_{ij}\right), \quad (7)$$

$$r_{ij} = HammingDistance\,(x_i^t, x^*), \quad (8)$$

C.

where I_i=i represents the light intensity of firefly x_i which equals to a randomly selected number ranging from 1 to r_{ij}. r_{ij} indicates the hamming distance between the current firefly x_i and firefly x_j that refers to the best firefly x* in the population. Light intensity also determines the neighboring locations in a tour (solution). The movement behavior of the fireflies varies in some way, thus they seems to get some kind of intelligence. As a result, the movement of a firefly changes depending on how distant it is from the population's brightest firefly x*. Consequently, before moving, a firefly will check its light intensity I_i^t. The firefly i will use 2-opt local search to do a short move if the intensity value is less than half of the cities; otherwise, it will use 3-opt to perform a lengthy move as stated previously in Equations 5 and 6.

D. Crossover

Evolutionary algorithms inspired a crossover operation (EA). This process produces new solutions that are close to two previously chosen solutions from a population. As a result, using crossover is an effective operation to do a local search and exploit the search space. [32] presented a variety of crossover operators with varying implementations. Partially matched or mapped crossover (PMX) is the one that TSP users are most familiar with. In PMX, two crossover locations are picked at random from a population's two solutions. The portion of solutions between the two crossing locations determines a matching selection that influences cross through position-by-position exchange processes. For example, consider these two parents:

P1: 4 5 1 2 | 9 8 7 3 | 6 10
P2: 6 4 5 1 | 7 8 2 10 | 3 9

We can get the following offspring by using PMX crossover:

O1: 4 5 1 2 | 7 8 10 | 9 6 3
O2: 6 4 5 1 | 9 8 7 3 | 10 2 3

III. COOPERATIVE DISCRETE FIREFLY ALGORITHM FOR TSP (CDFA)

The cooperative discrete FA model (CDFA) is a discrete FA version simulates the GA island model, which was, in turn, inspired by nature. The parallel FA model has two key characteristics that have been absorbed. The first is to prevent early convergence, which is caused in the basic FA, in order to preserve population diversity and enhance the solution of the problem under consideration. The second is to accelerate convergence by using many populations and assessing all fireflies inside each population concurrently. This cooperative model specifies a logical structure that may be implemented on a wide range of architectures. In any case, this study established the concept as a collection of populations that use multi-thread architecture to disperse themselves over the available processors. From time to time, each population autonomously searches out the best fireflies from the other populations and transfers them with others. In the suggested cooperative model, some fireflies from a population P[i], where i is the population index, are transmitted to another population P[(Rand(1; n)], where n is the number of populations. The following steps demonstrate the detailed overview of the design of CDFA algorithm.

Step 1: Parameters Initialization. All parameters are initialized initially in this stage to optimize the performance of the CDFA algorithm. The following are the parameters that are defined.

I. Basic FA parameters: The brightness (light intensity) and population size are the two most fundamental FA parameters. Light intensity reflects the advantages and disadvantages of firefly location and decides its movement direction. Population size indicates the number of the fireflies within each population which is determined at the beginning.

II. Number of populations (NP): The number of populations in CDFA is specified by this parameter. All populations are usually created with the same number of fireflies, allowing each population to be allocated to its own thread in the parallel computing system.

III. Exchange Interval (EI): The value of this parameter denotes the number of iterations required before fireflies begin to exchange across populations.
IV. Exchange Rate (ER): The percentage of fireflies to be moved from one population to another is determined by this parameter.
V. Exchange Topology (ET): The source and destination populations for swapping are determined by this parameter.
VI. Exchange Policy (EP): This parameter determines how the firefly from the source population are selected for exchanging, as well as how they are substituted in the destination.
VII. Number of Iterations: This parameter specifies the number of iterations necessary to obtain the best solution. This parameter, in addition to the best solution, represents the algorithm's termination criterion.

Step 2: Population Initialization. From the available range of values, each population is initialized with random values for each firefly as stated in Section 2.2. This step ensures the diversity of solutions. Then, depending to the problem under consideration, calculate light intensity and evaluate each firefly in the population using the fitness function.

Step 3: Population Updating. Create a new better firefly position (solution) by shifting each firefly xi approaching the brightest one xj depending on their attractiveness. This may be accomplished using either 2-opt or 3-opt procedures. Concurrently, the updated firefly's fitness function f(x_i) is calculated.

Step 4: Fireflies Exchanging. When the predetermined EI value is reached, choose the source and destination populations, and based on a predefined EP, exchange a number of fireflies equal to the ER between these two populations.

Step 5: Termination Criterion Checking. When the optimal solution is obtained or the maximum number of iterations is met, the CDFA algorithm is stopped.

IV. EXPERIMENTAL RESULTS

In this section, the performance of the CDFA algorithm is reviewed and analyzed in terms of the obtained computational results. For this analysis, two different tests were carried out, each using a different symmetric TSP standard benchmark with varying lengths taken from TSPLIB (http://comopt.ifi.uniheidelberg.de/software/TSPLIB95/). Both were ran on an Intel Core i5 processor. After performing the experiments 10 times for each instance, the results were gathered and reported as presented in this section.

CDFA parameters were set up as described in Table I. Each population of 50 fireflies was used to launch CDFA. There were a total of 500 iterations. Number of threads (number of populations) was 10, Exchange rate (ER) was set to 10 and exchange interval was equals 20 iterations. The exchange policy was to randomly select best fireflies from one population and substitute them with the worst ones since each population can communicate with each population in the model.

TABLE 1. PARAMETER SETTING OF CDFA PARAMETER

Parameter	Value
Number of Iterations	500
Population size	50
populations (threads) Number	10
Exchange Interval(EI)	20
Exchange Rate(ER)	10
Exchange policy	Select the best and replace with the worst
Exchange Topology	Fully connected graph

The three CDFA algorithm versions are analysed and compared in the first experiment. The first is DFA-opt in which the 2-opt and 3-opt strategies were integrated with fundamental discrete FA algorithm. The second is DFA-xover in which DFA-opt was combined with crossover operation. The third type is CDFA, which is the final improvement over the basic discrete FA. CDFA is DFA-xover with the parallelized cooperative model.

Table II demonstrates how the computing results improved when six symmetric TSP standard benchmarks were used with the different proposed versions of FA. Both DFA-xover and DFA-opt achieve similar results to the optimal solution in the following instances: Bayg29, berlin52, St70, Eil51, and Eil76, as can be seen from the tabulated values. In the instance of tsp225, however, DFA-xover exceeds DFA-opt in terms of optimum solution. DFA-xover, on the other hand, outperformed DFA-opt in nearly half of the examined cases when it refers to the average solution. The reason for this is due to the use of a crossover operation, which takes advantage of the algorithm's detected promising solutions and exploits the search space to speed up its learning capabilities. Similarly, CDFA outperformed DFA-xover in about 83 precent of all examined cases, in addition, CDFA outperformed DFA-opt in all instances. This advantage of CDFA is attributed to the use of a parallelized cooperative model in the algorithm's search process, which increases the diversity in the event that solutions become stuck in local optima.

TABLE II. CDFA RESULTS COMPARED TO DFA-OPT, DFA-XOVER. THE FINDINGS WERE COMPILED FROM TEN DIFFERENT RUNS. THE MOST PROMISING OUTCOMES ARE HIGHLIGHTED IN BOLD.

Instance	Optimal	DFA-opt		DFA-xover		CDFA	
		Avg.	Best	Avg.	Best	Avg.	Best
Bayg29	1610	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**
Eil51	426	426.3	**426**	426.4	**426**	**426**	**426**
berlin52	7542	**7542**	**7542**	**7542**	**7542**	**7542**	**7542**
St70	675	675.7	**675**	675.6	**675**	**675**	**675**
Eil76	538	539.8	**538**	539	**538**	**538**	**538**
tsp225	3845	3943.1	3923	3941.9	3920	**3916**	**3916**

TABLE III. CDFA RESULTS COMPARED TO MDFA AND PDFA. THE CDFA FINDINGS WERE COMPILED FROM TEN DIFFERENT RUNS. THE MOST PROMISING OUTCOMES ARE HIGHLIGHTED IN BOLD.

Instance	Optimal	CDFA				EDFA				MDFA			
		Best	Avg.	PDbest	PDavg	Best	Avg.	PDbest	PDavg	Best	Avg.	PDbest	PDavg
a280	2578	**2579**	**2579**	**0.039**	**0.039**	-	-	-	-	-	-	-	-
bayg29	1610	**1610**	**1610**	**0.000**	**0.000**	1624	1639	0.870	1.801	**1610**	1614	**0**	0.248
bays29	2020	**2020**	**2020**	**0.000**	**0.000**	**2020**	2066	0.000	2.277	**2020**	2036	**0**	0.792
berlin52	7542	**7542**	**7542**	**0.000**	**0.000**	8752	9135	16.044	21.122	7681	7933	1.843	5.184
eil51	426	**426**	**426**	**0.000**	**0.000**	497	540	16.667	26.761	432	443	1.409	3.991
eil76	538	**538**	**538**	**0.000**	**0.000**	789	813	46.654	51.115	554	574	2.974	6.691
gr202	40160	**40161**	**40161**	**0.003**	**0.003**	-	-	-	-	-	-	-	-
gr666	294358	**295584**	**295584**	**0.417**	**0.417**	-	-	-	-	-	-	-	-
pcb442	50778	**50913**	**50913**	**0.266**	**0.266**	-	-	-	-	-	-	-	-
st70	675	**675**	**675**	**0.000**	**0.000**	985	1039	45.926	53.926	682	706	1.037	4.593
tsp225	3845	**3916**	**3916**	**1.847**	**1.847**	-	-	-	-	-	-	-	-
ulysses16	6859	**6860**	**6860**	**0.015**	**0.015**	-	-	-	-	-	-	-	-
ulysses22	7013	**7014**	**7014**	**0.014**	**0.014**	-	-	-	-	-	-	-	-

In the second experiment, we used thirteen symmetric TSP standard benchmarks to compare our proposed method, CDFA, to the state-of-the-art discrete FA algorithms, PDFA and MDFA.

Table III compares CDFA to the state-of-the-art discrete FA algorithms, PDFA and MDFA, in terms of best and average solution using thirteen symmetric TSP benchmarks. In most cases, CDFA was able to obtain optimal results of ability to attain the optimal solution, with regard to the best and average solutions (ulysses16, ulysses22, bayg29, bays29, berlin52, eil51, eil76, st70 and tsp225). In the following four cases, CDFA came close to achieving an optimal solution for both the best and average solution. Furthermore, when compared to the average solution, CDFA surpassed the MDFA and PDFA algorithms in reaching the best solutions for all instances. Generally, in comparison to previous algorithms, the CDFA was able to search for the best solution until it met the termination criterion without premature convergence or stagnation, notably for medium and large cases. In general, the findings show that the CDFA structure, which is based on the integration of numerous ideas such as crossover, local search approaches, and the cooperative model, achieved the desired balance of diversification and intensification. As a result, the CDFA algorithm is able to escape local optima and accelerate convergence. Therefore, CDFA outperforms other algorithms such as PDFA and MDFA in obtaining suboptimal/optimal solutions to TSP problems.

V. CONCLUSION

This work used a new proposed FA version named the cooperative discrete firefly algorithm (CDFA), which is based on three primary contributions. The first is the use of local search techniques containing 2-opt and 3-opt to accelerate convergence toward optimum solutions. The second is the adoption of the crossover operator, which increases the intensification and so aids CDFA in efficiently exploiting the existing search space. The third step is to construct a multi-population model with a carefully managed communication strategy. This paradigm allows for the preservation of the diversity while also speeding up the execution. In terms of finding the optimal/near-optimal solutions for numerous benchmark TSP problems in the early iterations, the experimental analysis demonstrated that the CDFA variation outperforms the MDFA and PDFA. These findings show that CDFA can solve large-scale TSP and, as a result, more difficult optimization problems in real-world applications. CDFA's effectiveness may pave the way for new algorithms to improve the quality of solutions to challenging optimization problems.

REFERENCES

[1] I. Fister, X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," Swarm and Evolutionary Computation, vol. 13, pp. 34–46, 2013.

[2] I. Fister, X.-S. Yang, D. Fister, and I. Fister Jr, "Firefly algorithm: a brief review of the expanding literature," in Cuckoo Search and Firefly Algorithm. Springer, 2014, pp. 347–360.

[3] S. M. Farahani, A. A. Abshouri, B. Nasiri, and M. Meybodi, "Some hybrid models to improve firefly algorithm performance," International Journal of Artificial Intelligence, vol. 8, no. 12, pp. 97–117, 2012.

[4] A. Abdullah, S. Deris, M. S. Mohamad, and S. Z. M. Hashim, "A new hybrid firefly algorithm for complex and nonlinear problem," in Distributed Computing and Artificial Intelligence. Springer, 2012, pp. 673–680.

[5] A. Rajini and V. K. David, "A comparative performance study on hybrid swarm model for micro array data," Int. J. Comput. Appl, vol. 30, no. 6, pp. 10–14, 2011.

[6] X.-S. Yang and S. Deb, "Eagle strategy using l´evy walk and firefly algorithms for stochastic optimization," in Nature Inspired Cooperative Strategies for Optimization (NICSO 2010). Springer, 2010, pp. 101–111.

[7] A. M. Altabeeb, A. M. Mohsen, and A. Ghallab, "An improved hybrid firefly algorithm for capacitated vehicle routing problem," Applied Soft Computing, vol. 84, p. 105728, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494619305095

[8] A. M. Altabeeb, A. M. Mohsen, L. Abualigah, and A. Ghallab, "Solving capacitated vehicle routing problem using cooperative firefly algorithm," Applied Soft Computing, vol. 108, p. 107403, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1568494621003264

[9] I. Fister Jr, X.-S. Yang, I. Fister, and J. Brest, "Memetic firefly algorithm for combinatorial optimization," arXiv preprint arXiv:1204.5165, 2012.

[10] M. Sayadi, R. Ramezanian, and N. Ghaffari-Nasab, "A discrete firefly metaheuristic with local search for makespan minimization in permutation flow shop scheduling problems," International Journal of Industrial Engineering Computations, vol. 1, no. 1, pp. 1–10, 2010.

[11] M.-H. Horng, "Vector quantization using the firefly algorithm for image compression," Expert Systems with Applications, vol. 39, no. 1, pp. 1078–1091, 2012.

[12] J. Luthra and S. K. Pal, "A hybrid firefly algorithm using genetic operators for the cryptanalysis of a monoalphabetic substitution cipher," in Information and Communication Technologies (WICT), 2011 World Congress on. IEEE, 2011, pp. 202–206.

[13] G. K. Jati et al., Evolutionary discrete firefly algorithm for travelling salesman problem. Springer, 2011.

[14] M. SARAEI, R. ANALOUEI, and P. MANSOURI, "Solving of travelling salesman problem using firefly algorithm with greedy approach," Cumhuriyet Science Journal, vol. 36, no. 6, pp. 267–273, 2015.

[15] L. Zhou, L. Ding, and X. Qiang, "A multi-population discrete firefly algorithm to solve tsp," in Bio-Inspired Computing-Theories and Applications. Springer, 2014, pp. 648–653.

[16] L. Zhou, L. Ding, X. Qiang, and Y. Luo, "An improved discrete firefly algorithm for the traveling salesman problem," Journal of Computational and Theoretical Nanoscience, vol. 12, no. 7, pp. 1184–1189, 2015.

[17] A. M. Mohsen and W. Al-Sorori, "A new hybrid discrete firefly algorithm for solving the traveling salesman problem," in Applied Computing and Information Technology. Springer, 2017, pp. 169–180.

[18] L. Teng and H. Li, "Modified discrete firefly algorithm combining genetic algorithm for traveling salesman problem," TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 16, no. 1, pp. 424–431, 2018.

[19] M. Li, J. Ma, Y. Zhang, H. Zhou, and J. Liu, "Firefly algorithm solving multiple traveling salesman problem," Journal of Computational and Theoretical Nanoscience, vol. 12, no. 7, pp. 1277–1281, 2015.

[20] H. S. Chuah, L.-P. Wong, and F. H. Hassan, "Swap-based discrete firefly algorithm for traveling salesman problem," in International Workshop on Multi-Disciplinary Trends in Artificial Intelligence. Springer, 2017, pp. 409–425.

[21] L. Zhou, L. Ding, X. Qiang, and Y. Luo, "An improved discrete firefly algorithm for the traveling salesman problem," Journal of Computational and Theoretical Nanoscience, vol. 12, no. 7, pp. 1184–1189, 2015.

[22] L. Jie, L. Teng, and S. Yin, "An improved discrete firefly algorithm used for traveling salesman problem," in International Conference on Swarm Intelligence. Springer, 2017, pp. 593–600.

[23] M. Saraei and P. Mansouri, "Hmfa: A hybrid mutation-base firefly algorithm for travelling salesman problem," in Fundamental Research in Electrical Engineering. Springer, 2019, pp. 413–427.

[24] Y. WANG, Q.-P. WANG, and X.-F. WANG, "Solving traveling salesman problem based on improved firefly algorithm," Computer Systems & Applications, vol. 8, p. 37, 2018.

[25] A. V. Husselmann and K. Hawick, "Parallel parametric optimisation with firefly algorithms on graphical processing units," in Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2012, p. 1.

[26] M. Subutic, M. Tuba, and N. Stanarevic,

[27] "Parallelization of the firefly algorithm for unconstrained optimization problems," Latest Advances in Information Science and Applications, vol. 22, no. 3, pp. 264–269, 2012.

[28] X.-S. Yang, Nature-inspired metaheuristic algorithms. Luniver press, 2010.

[29] ——, Engineering optimization: an introduction with metaheuristic applications. John Wiley & Sons, 2010.

[30] W. Al-sorori, A. Mohsen, and W. Aljoby ßer, "An improved hybrid bat algorithm for traveling salesman problem," in Bio-inspired Computing – Theories and Applications, M. Gong, L. Pan, T. Song, and G. Zhang, Eds. Singapore: Springer Singapore, 2016, pp. 504–511.

[31] W. Al-Sorori and A. M. Mohsen, "Multi-population discrete bat algorithm with crossover to solve tsp," in Proceedings of the 16th International Conference on Hybrid Intelligent Systems (HIS 2016), A. Abraham, A. Haqiq, A. M.

Alimi, G. Mezzour, N. Rokbani, and A. K. Muda, Eds. Cham: Springer International Publishing, 2017, pp. 466–478.

[32] E. Osaba, X.-S. Yang, F. Diaz, P. Lopez-Garcia, and R. Carballedo, "An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems," Engineering Applications of Artificial Intelligence, vol. 48, pp. 59–71, 2016.

[33] P. Thakur and A. J. Singh, "Study of various crossover operators in geneticalgorithms," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 4, no. 3, 2014.