Received: August 20, 2021. Revised: September 16, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Adaptive Grey Wolf Optimization Algorithm with Neighborhood Search Operations: An Application for Traveling Salesman Problem

Ayad Mohammed Jabbar[1]* **Ku Ruhana Ku-Mahamud[2]**

[1]*Department of Computer Science, Shatt Al-Arab University College, Basra, Iraq*
[2]*School of Computing, Universiti Utara Malaysia, Malaysia*
*Corresponding author's Email: AyadMohammed@sa-uc.edu.iq

Abstract: Grey wolf optimization (GWO) algorithm is one of the best population-based algorithms. GWO allows sharing information in the wolf population based on the leadership hierarchy using the hunting mechanism behavior of real wolves in nature. However, the algorithm does not represent any key exchange information sharing for the traveling salesman problem because of two issues. The candidate solutions are improved dependently, similar to local search concepts, losing their capability as a population-based algorithm. The algorithm is limited in its search process in finding only the local regions and ignoring any chance to explore search space effectively. This study introduced an adaptive grey wolf optimization algorithm (A-GWO) to solve the information-sharing problem. The proposed A-GWO maintains sufficient diverse solutions among the best three wolves and the rest of the population. It also improves its neighborhood search by obtaining more locally explored regions to enhance information sharing among the wolves. An adaptive crossover operator with neighborhood search is proposed to inherit the information between the wolves and provide several neighborhoods to find more solutions in the local region. Experiments are performed on 25 benchmark datasets, and results are compared against 12 state-of-the-art algorithms based on three scenarios. The credibility of the proposed algorithm produces approximately 53%, 58%, and 63% better tour distance in the first, second, and third scenarios, respectively. The proposed A-GWO achieves approximately 87% better minimum tour distance compared with the GWO algorithm.

Keywords: Crossover operator, Exploration, Exploitation, Machine learning, Position update, Swarm algorithms.

1. Introduction

The search for optimal solutions in artificial intelligence aims to find the "best" solution among various solutions in the search space. This type of problem is known as combinatorial optimization and is considered an NP-hard problem [1]. Several examples of combinatorial optimization problems are vehicle routing problem (VRP) [2], traveling salesman problem (TSP), clustering [3], classification [4], and feature selection [5]. Stochastic methods have been introduced with alternative randomness called metaheuristics because of the many complexities and limitations in most combinatorial optimization problems. This result concerns the exponential expansion in the area of search for the best solutions and avoids problems to have early convergence and local optima problems. Metaheuristics is a problem-independent algorithm in finding several near-optimal solutions. The main characteristic of metaheuristics combines several heuristic methods that perform in higher-level metaphors [7, 8]. These metaphors are inspired by different behaviors, representing the swarm of insects, including foraging, dancing of bees, collection of eggs of ants, and odor for membership recognition in a colony. The swarm approach, an intelligence system, describes the collective behavior of social insects while interacting with their environment and one another to solve a specific problem [9, 10]. The interaction occurs because of external influences representing positive feedback used as a communication in the population. For example, pheromones in the ant colony optimization algorithm (ACO) make the insects converge and perform a

unique behavior following one another [10]. The key aspects of swarm intelligence are decentralization and self-organization, wherein the population represents the power concept. One of the popular population bio-inspired behaviors is the evolutionary genetic algorithm (GA). GA represents a key overlapping feature, as introduced by Holland (1973). Survival of the fittest is a fundamental concept applied in which the candidate solutions are allowed to procreate further solutions on the basis of results obtained by previous iterations. The crossover and mutation operators are used iteratively to produce a new candidate solution.

Information sharing is a fundamental element of swarm intelligence. However, such sharing is implemented in several population-based algorithms, such as ACO, artificial bee colony (ABC), grey wolf optimizer (GWO), and GA. The crucial elements for swarm intelligence algorithms include self-organization, stigmergy, and positive feedback concept. The principle of self-organization is a process of building an organized and cohesive system from a disordered system on the basis of basic issues that are solved simultaneously [11]. Flocking of birds and ant foraging are typical examples that demonstrate the self-organization element of activities [12]. Stigmergy is another important concept that is represented as a chemical substance used in a swarm intelligence system in which the information is exchanged among different members that belong to the same population in the colony. GA does not have any stigmergy concept to exchange information, whereas self-organization and positive feedback are utilized to optimize the objective function. Following the same concept of swarm intelligence, GWO has a population of different wolves responsible for various tasks. The three main wolves are called alpha, beta, and delta, which are responsible for heading the hunting activity. The remaining wolves in the pack are called omega wolves, which update their positions based on the main three wolves.

Different combinatorial optimization problems have employed GWO, which shows promising results in optimizing the objective function of the problem and exploring the search space using the best wolves in the population. However, Sopto (2019) showed a shortcoming in the mechanism of updating the positions of omega wolves, which was locally performed [13]. Therefore, the omega wolves would update their positions based only on local changes, similar to what local search algorithms do. Each omega position is updated based on the neighborhood search of the omega itself without looking for global solutions of alpha, beta, and delta positions. The algorithm is limited in its search process in finding only the local regions and ignoring any chance to explore other parts of the search space. The algorithm does not represent any key exchange or sharing of information between the wolves, allowing the algorithm to improve different solutions during the algorithm run.

In this study, an adaptive algorithm is proposed with two new modifications for information sharing. The modifications are on adaptive crossover operator and neighborhood search to enhance the neighborhood search in the best local regions at the leadership hierarchy. The rest of the paper is organized as follows. Section 2 discusses related works. Section 3 introduces the proposed adaptive algorithm. Section 4 presents the evaluation of the proposed algorithm. Section 5 concludes the work and highlights future research.

2. Related works

Stochastic algorithms ensure the construction of solutions according to the quality of solutions guided by some objective functions and compact with some randomization to explore the search space and avoid stacking in local optima. Stochastic algorithms utilize different concepts, such as local search and tuning parameters, to optimize the objective function deeply, wherein the problem is formulated as the optimization problem [14-16]. The three main approaches popular in optimization are exact, estimation, and approximation. Exact algorithms can produce the optimal solution to an optimization problem within a dependent runtime instance. However, exact algorithms require exponential time, especially with complex optimization problems, and thus are difficult to use in complex problems. The estimation approach does not guarantee an optimal solution because the results are produced according to a predefined range of inputs [17]. The optimal or near-optimal solutions can be generated in a short time using the approximation approach. Although this approach does not guarantee finding the optimal solution always, it can produce reasonable solutions. This approach optimizes the problem on the basis of a single or population approach. The single approach, such as simulated annealing, tracks the improvement of one single solution candidate, whereas the population approach iteratively modifies a set of candidate solutions based on the algorithm feedback [18]. This popular approach can be either a swarm or an evolutionary algorithm (EA).

EAs are a stochastic iterative search method, which simulates the evolution of species in nature. A set of candidate solutions evolves iteratively. These

candidate solutions are known as the population of the algorithm, where each individual has its fitness function for survival through life. The main operations in EAs are selection, recombination, and mutation. Each operation is responsible for increasing the accuracy of the solution. The probabilistic application is the search process used in EAs to find better solutions. It is guided by the objective function of the optimization problem, which represents the survival of the best individuals. The most known algorithm under this category is the GA. GA is one of the best EAs that have been successfully applied in several application domains. It has been used as the main part of many modern algorithms because of its components' popularity and simplicity. The algorithm inspired by nature represents the theory of Charles Darwin and includes a selection of the best individuals for optimization. The process consists of selecting individuals who are most fit for reproduction to achieve the best offspring. The offspring for the next generation completes the cycle of Charles Darwin's theory. The initial step in GA initializes each chromosome as a single solution. The reproduction consists of using a crossover operator. The operator uses a set of chromosomes to be mated to produce offspring better than the older population. Parts of a chromosome are moved to other parts of the chromosome to create new offspring. However, a crossover can be performed in many ways. Many genes are used, and the location of the genes on a chromosome plays a vital role in producing a better solution. A mutation operator is used to maintain the diversity of the solution in GA. The operator alters one or more genes in the chromosome, such as moving one gene (city) to other locations and modifying the route. Several researchers have used an adaptive strategy in the GA algorithm. An adaptive GA utilized three crossover strategies [19]. A fit offspring was determined by selecting the best strategy while running the algorithm, which was adaptively performed. Information reinforcement was conducted by utilizing pheromone information, which was updated using the best strategy according to the ACO model pheromone update. However, the algorithm did not use the evaporation procedure of ACO, which in the end may converge quickly on one of these strategies because of the wrong decision selected in the initial algorithm run.

Sung and Jeong (2014) proposed an algorithm that adaptively changed the crossover parameter and the mutation parameter during the algorithm run to generate a new population [20]. Nevertheless, the algorithm showed better results only with small datasets, where adaptive search could find the best solution. Similar research proposed the adaptive crossover algorithm using the 1Bit Adaptive Crossover, where the specified crossover factor was coded in the genotype [21]. Riff and Bonnaire (2002) introduced this concept by using more operators [22]. It was extended in 2004 by encoding rate and reward operators, where the value change according to the fitness of the offspring was better or worse than its parents [23]. Cruz-Salinas and Perdomo (2017) extended this work by having a population of operators that were exposed to evolution and mutation and maintaining the operator selection method [24] employed in [23]. However, the algorithm used operators that were not suitable for the ordered nature of TSP. Similar research has proposed an adaptive two-opt mutation in the process of mutation in the GA algorithm [25]. The strategy changed the two-opt mutation to another operation called exchange cities, where a pair of cities were swapped one at a time adaptively. Nevertheless, the adaptive strategy employed did not memorize either the two-opt mutation or exchanges during the time. It only used the best one according to the fitness of the solution.

A kind of biology-inspired concept is swarm intelligence, which expresses the regular behavior practiced by living organisms to communicate or solve different problems. Swarm intelligence is motivated by groups of insects, such as ants, bees, and bacteria, behaving intelligently as groups instead of a single insect. Several algorithms, such as ACO, ABC, and GWO, are examples of the swarm intelligence category because they use the basic swarm intelligence concepts, including self-organization, stigmergy, and positive feedback. Developed by Dorigo [26], ACO is a publication-based swarm algorithm inspired by the foraging behavior of ants. The algorithm has been successfully applied in different NP-hard combinatorial problems, including classification, clustering [27, 28], and feature selection [29, 30]. Dorigo modeled the foraging behavior to solve TSP when the stigmergy concept was used as an indirect communication guide for the colony to find the shortest route for TSP [31]. The algorithm has many parameters that control the algorithm's performance to produce better results in different application domains, including clustering, classification, and feature selection. The density of a pheromone laid by an ant controls the probability of choosing the best arc according to the value of the pheromone, which is the core of the algorithm and has been adaptively optimized by different researchers. In ACO, stochastic methods are performed by integrating randomization through the

ant movement from city to city. This integration guarantees that the algorithm can avoid the local optima problem and explore the search space efficiently. The most important part of this algorithm is its model, which simulates foraging behavior, such as using the evaporation process, which is set by a parameter (evaporation rate) optimized by different researchers. The pheromone trail evaporates during the time. Therefore, long-distance routes are forgotten because ants have no desire to pass them. Different modifications in ACO have been proposed in the literature since the initial ant system proposed by Dorigo and the ant colony system [32].

In 2012, the pheromone decay parameter (pheromone evaporation rate) was adapted to avoid local optima and adjust the convergence rate during the algorithm run [33]. However, the adjustment was performed by using significant value changes in a descending manner, which did not guarantee to find the best value (best evaporation rate) that represented a particular dataset. Adaptive-related research has utilized different groups of ants to select pheromone arcs with different concentrations [34]. Nevertheless, the major problem of the algorithm was the high diversity of solutions and reinforcing the algorithm to converge on the best solution in a long time. This shortcoming occurred because different groups used different transition probability rules during the algorithm run. In 2021, another hybrid Harris's Hawk and ACS, a variant of the ACO algorithm (HHO-ACS), was proposed to optimize the ACS parameters [35]. Five parameters are subjected to optimization. In the end, the algorithm's performance is determined by the pheromone coefficient, heuristic coefficient, decision rule, and the evaporation rate of the pheromone. This kind of optimization problem is called online parameter tuning based on an extended algorithm. HHO algorithm optimizes the five ACS parameters during the algorithm run, thus find the best value for each parameter in solving TSP. HHO-ACS algorithm achieved better results than the ACS algorithm but the exponential time required in the test compared to both algorithms is long. Other related swarm algorithms, namely the black hole algorithm (BH) proposed to solve the problem of TSP [36]. The algorithm produced promising results compared with other swarm algorithms. However, two limitations have been investigated in the BH algorithm. First, the performance of the algorithm is achieved based on the population that is randomly initialized. The second showed a low exploitation-based search, where its standard deviation was high compared with other swarm algorithms.

TSP has also been solved by the ABC algorithm, which is swarm-based inspired by a bee colony looking for food, representing the foraging principle in insects. The simple steps of the algorithm allow the researchers to apply the algorithm in several application domains. The algorithm consists of three main bees. The core engine of the algorithm includes the scout bee, employee bee, and onlooker bee. Each bee has its tasks, such as exploration, that are relevant to a scout bee. Exploration in the neighborhood of solutions is related to an employee bee task. An onlooker bee performs and exploits a task to find a deeper region with high-quality solutions. However, the algorithm has a limit parameter set statically by users, indicating how many times the solution can be accepted once the algorithm achieves the same result for a while. This parameter has been optimized by different researchers who included adaptive and self-adaptive strategies [2, 36–39]. The purpose of the parameter is to restart the search space process frequently during the algorithm and allow the finding of a better region on the search space by avoiding the local optima solution. The algorithm generates the initial solution randomly (initial city sequence), whereby each bee represents a single solution comprising a set of cities visited one at a time. Each employee explores the local region by using a neighborhood structure, modifying the sequence locations of the cities. However, this step is considered by some researchers as an exploitation phase because the employee bee performs and improves locally. This step ensures finding better solutions in the neighborhood region.

The adaptive ABC algorithm for a TSP proposed in 2013 was adaptive-count initialized for each bee, which changed dynamically during the algorithm run [40]. Nevertheless, this study did not clarify which count value was suitable for each dataset. Another limitation was that the scout had multiple jumps, not ensuring that all feasible solutions in the local region were found. Another study on numerical optimization was done in 2017 [41]. It proposed adaptive ABC on the basis of source food ranking. Each source food was ranked higher when more opportunities could be selected. However, the algorithm could only imitate if the selected food sources in the early search stages had high fitness, forcing the algorithm to converge quickly without exploring the local neighborhood region. The same idea of food source perturbation using a synergetic mechanism was proposed in 2020, enhancing the population diversity in algorithm initialization using a chaotic round map [42]. Another study used adaptive elitism-based immigration, which replaced the worse wolf in the population during the algorithm run with an elite individual with better fitness function with a controlled-mutation parameter. However, the mutation value was

controlled in some cases to a random value, which was distributed to an individual after the mutation process in an unpromising region. A 2019 study examined the performance of different variants of ABC using several statistical tests with 15 TSP instances. The test also included the convergence rate and the parameter setting of the ABC algorithm, such as the limit parameter. The limit parameter significantly controlled the ABC algorithm; reporting suitable parameter values for TSP [43].

Particle swarm algorithm (PSO) for a traveling salesman problem has been improved by introducing the best current solution [44]. The basic idea is to use the best current iteration in the moving steps, improving the movement of the particles toward the best regains that have the best quality of solutions in the search space. However, the algorithm quickly stuck locally because the search process was only guided by the objective function, which was limited to the stochastic search to explore the search space effectively. Another related study proposed a hybrid PSO and ACO algorithm based on the best-worst ant [45]. The improvement starts with the population, which PSO initializes. The ACO algorithm is then performed to improve the solutions. The first step ensured that the generation of the initial population was iteratively better than what ACO does in a constructive manner. A new swarm algorithm, chicken swarm optimization, was proposed in 2021 to solve the TSP problem [46]. Although the algorithm produced promising results, it has limitations in maintaining quality when feeding back the algorithm to find more solutions in the neighborhood. Other hybrid algorithm proposed discrete whale optimization (DWO) with ACO algorithm to improve the performance of DWO [47]. The initialization of individuals improved by the ACO algorithm as the initial phase of DWO algorithms. Although DWO has been improved, the algorithm is still easily stacked in the local optima if the ACO algorithm is produced an initial population with high-quality solutions. DWO was improved by other researchers in 2021 using a variable neighborhood algorithm. [48]. This improvement increases the exploitation-based capability to find more local regions in the search space. However, the proposed algorithm cannot explore the search space when stagnation occurs. In 2019, another swarm algorithm, dragonfly algorithm (DA), was proposed to solve the problem of TSP [49] and showed promising results compared with other swarm algorithms.

The GWO literature indicates that the most important issue is the use of the updated position equations when GWO is utilized for different combinatorial optimization problems. Maintaining the balance between the two phases is the core engine of the algorithm, and both phases are affected by the updated position equations. The first three wolves (alpha, beta, and delta) represent the best positions of the hierarchy (in the search space of the problem). The remaining wolves (omega) are improved during the run according to that hierarchy level. The omega wolf position depends only on three wolves. A study done in 2019 had a mechanism for updating omega position [13]. Its limitation is that it was locally performed only when each solution was updated based on the neighborhood search without looking for a global region. Thus, omega wolves could be densely settled in the same region or certain regions during the prey catching process. In other cases, the mechanism of updating the omega position is locally performed only when each solution is updated based on the neighborhood search, preventing the exchange of information between the omega and the first three wolves (alpha, beta, and delta). This issue has been observed in the TSP problem, where omega wolves are improved by only using the neighborhood structure. No real update position is available to move the omega wolves to the best region and provide a chance during the run to explore a more promising region in the search space [13]. Two methods have been used in other related works to improve the performance of GWA for competitive traveling salesman problems [50]. The benefit of both methods is to increase the exploration and exploitation of the algorithm during the algorithm run. The static method used as the first method to divide cities evenly among salesmen, whereas the parallel method was used as the second method for all cities available. However, the methods are limited in the sharing of information between agents.

3. Proposed adaptive algorithm

The grey wolf optimization algorithm is metaheuristic. It simulates the social behavior and leadership hierarchy of grey wolves in real life. Social behaviors, such as hunting, organization, and decision-making, make the algorithm a unique model used by researchers to solve different optimization problems. Each group of wolves usually contains between 5 and 12 members. The members are sorted according to the hierarchy level, where members are known as alpha (α), beta (β), delta (δ), and omega (ω). Each of the omega wolves in the group has its task in the pack regardless of its position. The α wolf is responsible for the leadership and decision to hunt in addition to other tasks, such as sleeping and waking up for all wolves. The β wolf helps the α wolf in the

A-GWO algorithm

```
Input: Data (TSP Instances)
Output: Best Solution
Generate initial wolf population;
Calculate fitness f(X_i);
Identify three best wolves as Xα, Xβ, and Xδ;
IterationIndex=1;
WHILE (IterationIndex < Max)
Repeat
Update Position of X_i;
S*=Crossover();
IF(S*<X_i)
X_i= S*;
S*= NeighborhoodSearch(X_i); /
IF(S*<X_i)
X_i= S*;
Compute break point;
Until termination condition meet
UpdateCoefficient();
Calculate fitness f(X_i)
UpdatePosition();
IterationIndex = IterationIndex + 1;
UNTIL (IterationIndex = Max)
END-WHILE
```

Figure. 1 A-GWO algorithm

decision-making and is considered the best successor for the group if the α wolf dies. The δ wolf, who is the third level in the hierarchy, is responsible for organizing the ω wolves. Without the δ wolf, the group will encounter internal chaos. In hunting, grey wolves form the main nerve for group survival, which involves searching, tracking, encircling, and attacking prey. Those tasks are mathematically modeled as the GWO algorithm.

The process of encircling the prey is mathematically modeled as the movement of the grey wolf's location to the prey location and circling it. This task can be achieved by finding the distance between the grey wolf and prey location in the search space. The movement of the grey wolf involves the exploration and exploitation processes in finding a new location during the search and avoid falling into the local optima problem. This study proposes an A-GWO algorithm by introducing two new modifications to the present GWO algorithm. These modifications are essential to enhance the search process of the algorithm. The adaptive crossover operator and neighborhood search operations are proposed as the core engine of the algorithm. The standard GWO algorithm allows the sharing of information in the wolf population using leadership hierarchy. However, it ignores information sharing between omega wolves and the three leadership hierarchy wolves for the TSP problem [13]. In the A-GWO algorithm, information sharing between the best three wolves and the rest of the population is maintained, improving the search in the neighborhood regions. Both modifications are used to explore the best solution found in global and local regions from the search space. Figure 1 illustrates the A-GWO algorithm.

The A-GWO algorithm has three stages: encircling, hunting, and attacking the prey (Steps 1–3). The modifications are performed in those stages, including the crossover operator encircling the prey, selecting the best solution, performing neighborhood search in hunting the prey, and computing the breakpoint in attacking the prey. The algorithm starts with the initialization of all parameters and wolf population. Step 4 calculates the fitness function of all wolves, and Step 5 determines the best three wolves, α, β, and δ, according to the fitness function of each wolf in the population. Step 6 starts the algorithm iterations by initializing the value of the iteration index to 1. Step 7 initializes iterations by checking each wolf, and Step 8 starts the algorithm cycle. Step 9 checks each wolf's position in the search space. Step 10 has the first modification, which represents encircling the prey, which is

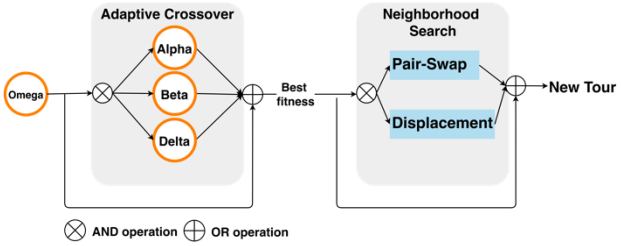

Figure. 2 Adaptive crossover and neighborhood search processes

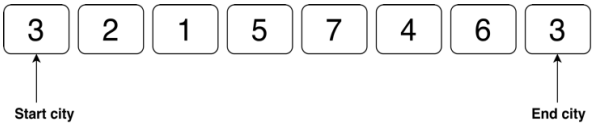

Figure. 3 Agent representation in A-GWO algorithm

mathematically modeled as the movement of the grey wolf's location to the prey location to be surrounded. This task can be achieved by finding the distance between the grey wolf and prey location in the search space. The movement of the grey wolf maintains the exploration and exploitation to find a new position during the search and avoid falling into the local optima problem. This process is articulated using the crossover operator, as shown in Step 10. This step represents the sharing of information between each ω wolf with the α, β, and δ wolves. However, the crossover operator is performed according to a predefined breakpoint, which is adaptively changed during the run and represents the matting process. The best matting is selected as the new position for the current ω wolf, which, in the end, represents the prey hunting stage of steps 11–12. The hunting starts by surrounding the position of the prey in the search space. However, no real knowledge about the prey position as in nature is available in the mathematical model. The only information provided is the positions of the three best wolves in the search space α, β, and δ, representing the best solutions in the algorithm. Therefore, the three candidate solutions are used to guide the rest of the ω wolves toward the best location that is surrounded by α, β, and δ. As a result, each ω wolf updates its position according to the best crossover operator between ω wolf and the three best wolves. The process of attacking the prey can be represented as the time of performing the attack and determining the best time to make the attack (steps 14–16). In the algorithm, the breakpoint represents exploration and exploitation, controlled according to the value of a that changes linearly. The second modification (i.e., the neighborhood search) (step 13) increases the probability of finding better solutions in the neighborhood region of wolves during the exploration that is performed in the crossover operator.

Differences are observed between the proposed algorithm, the original algorithm GWO in 2014 [51], and the algorithm in 2019 to solve TSP [13]. The prey encircling process in the original GWO uses absolute distance to modify the position. In contrast, the A-GWO algorithm uses the crossover operator but did not explicitly specify this process. Sharing of information can be seen in the original algorithm GWO between the omega wolves and the leaders, but no information sharing is found between the wolves in [13] solving TSP. Finally, GWO and A-GWO utilize a parameter in the attacking stage, where the value changes linearly during the algorithm run. However, in [13], no parameter is used, making the algorithm unable to explore and exploit the search space according to the changes in the parameter value during the algorithm run.

Details of Steps 9–15 in Fig. 1, consisting of both modifications, are highlighted in Fig. 2. A-GWO constructs many solutions by performing high exploration in the early search stages adaptively based on the crossover operator, which linearly changes during the algorithm run. The neighborhood search using displacement and pair-swap operations increases the probability of finding high-quality solutions in the neighborhood region of wolves.

4. Updated position in A-GWO

Each agent (wolf) is given a unique number representing a city to solve the problem of finding the minimum tour length in TSP. Each agent denotes one candidate solution of length (n+1), where n is the maximum number of cities. The tour starts and ends in the same city, which signifies a complete rule. In A-GWO, each agent randomly generates a candidate solution of one dimension, representing the initial solution, as shown in Fig. 3.

The first modification, which is the adaptive crossover, is linearly changed during the algorithm run. The second is the use of neighborhood search to explore more regions in the local neighborhood structure [40, 51]. The crossover operation starts with high exploration by exchanging the cities between two tours. The exchange includes an omega tour with alpha, beta, and delta tours, as shown in Fig. 4. The purpose is to move omega wolves to the best position near the three best wolves. In the beginning, the exploration is high to explore more positions in the search space. This condition increases the probability of finding other prey positions with better fitness functions (minimum tour distance). The crossover operation gradually decreases in the advanced search process to improve the prey position's search.

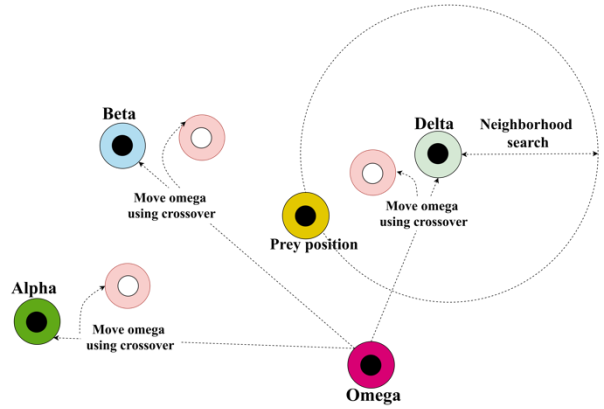

Figure. 4 Position updating in A-GWO

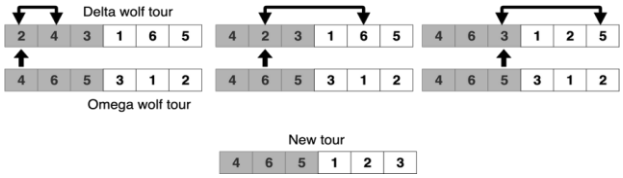

Figure. 5 Example of crossover operator using partially mapped crossover technique

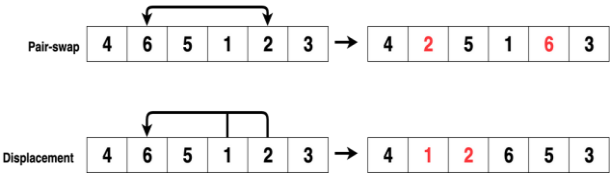

Figure. 6 Neighborhood search using pair-swap and displacement

In each iteration, the crossover is performed if the fitness function is better than the omega. Otherwise, it is rejected, and the older tour is kept. The crossover utilizes the technique called partially mapped crossover. This technique changes one city randomlyand swaps cities from two tours and within the tour itself. This process guarantees that no city is visited twice, which is a violation of the TSP rule. This process is illustrated in Fig. 5. In the example between omega and delta wolf tours, three cities are moved from the omega wolf tour into the delta wolf tour. The omega wolf updates its position according to the new tour produced if it has better fitness, ensuring that the wolf moves to a better position in the search space.

The second modification improves the exploitation capability, especially in the advanced search space using the neighborhood search [52, 53]. This modification aims to avoid stagnation when wolves are located in the same location. The slight modification in the position of cities in the tour can move the wolves into new locations within the neighborhood structure. The neighborhood search changes cities in the same tour (new tour in Fig. 5) using two neighborhood operation approaches, namely, displacement and pair-swap operations. The pair-swap operation swaps only two cities randomly to ensure moving the agent into a new location. Meanwhile, displacement operation moves a subsequence of cities (a length of the subsequence of cities randomly) and instates them into a new position in the tour, as illustrated in Fig. 6. The figure shows the pair-swap between city numbers 6 and 2. The displacement operation moves a subsequence of cities 1 and 2, inserting them before city number 6.

- Pair-swap: Two cities (two city positions) are randomly selected from the tour to be swapped.
- Displacement: A random position from the tour is selected with a random subsequence of city positions, and the subsequence of cities is moved before the selected random position.

The tour is accepted as a new better tour if and only if its quality is better than the omega tour Eq. (1). However, the vital issue is how can the omega tour provide better quality tour fitness. Such tour fitness can be produced either based on the crossover operator between the omega tour and the three best wolves, alpha, beta, and delta or by keeping the current omega tour if the fitness function of omega wolf is better than that of crossover operation, as reflected in the proposed Eq. (1). In Eq. (1), two solutions are S* and S0, where S* represents the improved solution, and S0 represents the non-improved solution. S* is obtained according to Eq. (2), which performs three crossover operations; only one is accepted as the best solution, represented by S*. S0, the non-improved solution, is accepted when it has better fitness quality than S*. The crossover is linearly changed during the iterations according to a parameter proposed in this algorithm, namely, break point. The breakpoint is used in the crossover to cut the tour in each iteration. The crossover stops until the value of breakpoint equals 0 or less, which ensures that the crossover starts with high diversity in the early search process and ends with low diversity in the advanced search process. The breakpoint value, as shown in Eq. (3), is computed according to the value of a. This step is followed by the neighborhood search, as shown in step 13 in Fig. 1., to improve the tour fitness using either pair-swap or displacement.

$$\text{The acceptance criterion} = \begin{cases} S*, \ if \ \text{Quality}\,(S*) > \text{Quality}\,(omega) \\ \\ S0, \ Otherwise \end{cases} \quad (1)$$

$$\text{Quality}\,(S*) = \begin{cases} S1 = Crossover\,(Omega, Alpha), \\ S2 = Crossover\,(Omega, Beta), \\ S3 = Crossover\,(Omega, Delta), \end{cases} \quad (2)$$

$$\text{Breakpoint} = (\text{a.M}x\text{\_cities}) - \text{M}x\text{\_cities}) \quad (3)$$

The greedy process in Eq. (1) improves the tour solution by accepting only the best solution from the crossover based on the linear value changed during the run. In the algorithm, the modifications are made at the "compute breakpoint value" and "update current position X_i.' The calculation to compute the breakpoint is performed using Eq. (3) is used to calculate the current position of X_i using Eq. (1). The breakpoint in Eq. (3) decreases during the algorithm run. Thus, the crossover operator in Eq (2) changes the length of the solution adaptively, linearly decreasing the long length. This step is followed by using the neighborhood search to select the best solution either from pair-swap or displacement. This stage aims to increase the algorithm's exploitation capability in finding more neighborhood solutions from the best global region.

5. Performance evaluation

The performance of the proposed A-GWO algorithm is evaluated using 25 TSP benchmark datasets taken from TSPLIB [56]. These datasets differ in the number of cities, including small and medium numbers of cities, and the maximum and minimum distances between the cities, as shown in Table 1. Table 1 also lists the distribution of the problems and the optimal solution of each problem. Experimental results of A-GWO are performed in three scenarios. The first scenario compares A-GWO with four state-of-the-art algorithms that are best known for providing the best tour distance: ACO, GA [57], GWO [13], and the producer scrounger method (PSM) [58], based on the minimum tour distance across all cities in the first set of TSP instances. The setting of the parameters for all the algorithms is set similar to that in [13]. GA and PSM represent GA and PSM, respectively. The population size is initialized equally, where each algorithm's population size is set to 100. The number of iterations is set to 500, and the number of runs is set to 10 for all algorithms, which meant that all algorithms had the same improvement time to produce the result within the same parameters. The parameter mutation is set to a small value to increase the diversity of solutions, usually applied with a low probability of approximately 0.001. A high probability indicated that GA is reduced to a random search. The crossover is set to 0.8, which guides to algorithm toward the best quality solutions during the algorithm run. The evaporation rate is set to 0.01 in ACO, which is responsible for increasing and decreasing exploration capability, where low evaporation means fast convergence and vice versa. The other parameters, such as the coefficients in GWO and A-GWO, decreased from 2 to 0 throughout the iterations. Such a decrease forces the algorithm to increase exploitation capability in the advance iterations, which is used to attack the prey when it stops moving in the GWO algorithm. All parameters were initialized as the literature in [57, 13] for a fair comparison. The second scenario of the comparison is performed with other benchmarks and other optimization algorithms. The target of the scenario is to indicate the effectiveness of the proposed technique by finding the optimal solutions only. The TSP instance used in the second scenario, as shown in Table 1, belongs to the standard library and is selected to test eil76, eli51, berlin52, kroa100, st70, oliver30, pr76, pr107, ch150, d198, tsp225, and f1417. The algorithms used in the second scenario are discrete whale optimization algorithm (DWOA) [47], discrete whale optimization algorithm with variable neighborhood search (VDWOA) [48], bat algorithm (BA) [59], GWO, moth-flame optimization (MFO) [60], and PSO [61]. In this scenario, each algorithm is performed 50 times, and the optimal solution provided is used in the comparison. The parameters of DWOA and VDWOA are similarly set in the experiment, where constant-coefficient b is set to 1, and the ε value is set to 0.35. In BA, the maximum pulse frequency value is set to 1, whereas the minimum value is set to 0. The coefficient of sound loudness is 0.9, the search frequency the enhancement factor of is 0.9, the loudness of the sound is between (0, 1), and the pulse emission rate is between (0, 1). The spiral shape parameter is set to 1 in MFO. The inertia weight factor is set to 0.2, and the acceleration factor is set to 2 in the PSO algorithm.

The third scenario was performed with other optimization algorithms. Six optimization algorithms used to indicate the effectiveness of the proposed technique include HHO-ACS [35], ACO [57], PSO [61], GA [57], BH [36], and DA [49]. The benchmarks used in this scenario, as shown in Table 1, contain eight datasets: bays29, att48, eil51, berlin52, st70, eil76, and eil101. In this scenario, the maximum number of iterations is set to 200, and the population size is set to 100 for all algorithms. All parameters are set based on [35].

Table 2 shows the average results of the minimum tour of the algorithms. The figures in brackets are the standard deviation of the results, and the best result

Table 1. Benchmark characteristics of each TSP dataset

NO	Dataset	Number of cities	Distribution	Optimal solution
1	burma14	14	14-cities in Burma	33.23
2	ulysses16	16	Odyssey of Ulysses	68.59
3	gr17	17	17-city problem (Groetschel)	2085
4	gr21	21	21-city problem (Groetschel)	2707
5	ulysses22	22	Odyssey of Ulysses (Groetschel and Padberg)	70.13
6	gr24	24	24-city problem (Groetschel)	1272
7	fri26	26	26-city problem (Fricker)	637
8	bays29	29	29-cties in Bavaria (street distance)	2020
9	hk48	48	48-city problem (Held/Karp)	11461
10	eil51	51	51-city problem (Christofides/Eilon)	426
11	berlin52	52	52-locations in Berlin (Germany)	7542
12	st70	70	70-city problem (Smith/Thompson)	675
13	eil76	76	76-city problem (Christofides/Eilon)	538
14	gr96	96	Africa-Subproblem of 666-city TSP (Groetschel)	55209
15	kroa100	100	100-city problem A (Krolak/Felts/Nelson)	21282
16	oliver30	30	30-city problem	420
17	pr76	76	76-city problem (Padberg/Rinaldi)	108159
18	pr107	107	107-city problem (Padberg/Rinaldi)	44303
19	ch150	150	150-city problem (Churritz)	6528
20	d198	198	Drilling problem (Reinelt)	15780
21	tsp225	225	A TSP problem (Reinelt)	3916
22	fl417	417	Drilling problem (Reinelt)	11861
23	bays29	29	29 Cities in Bavaria	-
24	att48	48	48 capitals of the US	-
25	eil101	101	101-city problem (Christofides/Eilon)	629

Table 2. Average tour distance for all algorithms (first experiment)

TSP instance	GA	ACO	PSM	GWO	A-GWO
burma14	31.83	31.21	30.89	30.87	**30.25**
ulysses16	74.79	77.13	74.2	73.99	**69.05**
gr17	2458.36	2332.58	2375.39	2332.58	**2321.08**
gr21	3033.82	2954.58	2838.22	2714.65	**2711.11**
ulysses22	79.62	86.81	76.68	76.08	**72.65**
gr24	1402.01	**1267.13**	1372.57	1289.23	1277.03
fri26	689.49	646.48	675.24	644.67	**639.06**
bays29	9981.49	9964.78	9917.59	**9219.40**	9390.40
hk48	16033.31	12731.07	13870.94	**12117.05**	12122.63
eil51	592.3	504.83	474.58	463.29	**459.02**
berlin52	10413.61	**8088.95**	8865.08	8289.11	8115.18
st70	1203.35	**748.65**	845.40	800.14	755.12
eil76	926.4	**601.77**	631.58	629.24	618.92
gr96	1092.04	590.67	618.68	660.48	**584.21**
kroa100	57940	**24623.01**	30210.57	28340.42	25150.04

for each dataset is highlighted. The A-GWO algorithm produced the best results in eight datasets, followed by the ACO and GWO algorithms with five and two datasets, respectively. A-GWO, GWO, and ACO have more control over the exploration and exploitation than GA and PSM, enabling the algorithms to produce better results regardless of the datasets' characteristics. However, ACO has better stability than GWO and A-GWO, as indicated by the small values of the standard deviation. Furthermore, the average standard deviation shows that ACO, A-GWO, GWO, PSM, and GA receive 196, 2128, 2736, 4473, and 6802, respectively. The average standard deviation indicates that the proposed A-GWO is ranked as the second algorithm producing a better standard deviation.

The second scenario experiments with VDWOA, DWOA, BA, GWO, MFO, and PSO algorithms, as shown in Table 3. The experiment is performed according to the optimal solution (minimum value as)

Table 3. Optimal Solution of seven algorithms (second experiment)

TSP instance	VDWOA	DWOA	BA	GWO	MFO	PSO	A-GWO
Oliver30	420	420	420	422	423	424	420
Eil51	429	445	439	441	449	445	**427**
Berlin52	**7542**	7727	7694	7898	8184	7862	7544
St70	676	712	718	726	710	732	**675**
Eil76	554	579	561	565	577	595	**545**
Pr76	108,353	111,511	111,989	114,261	114,377	115,265	**108,321**
KroA100	21,721	22,471	23,424	22,963	23,456	23,480	**21,717**
Pr107	**45,030**	45,780	46,419	46,083	47,437	46,919	45,042
CH150	6863	7329	7440	7384	7329	7833	**6858**
D198	**16,313**	16,603	16,849	17,109	16,911	18,130	16,509
Tsp225	4136	4399	4427	4620	4469	5049	**4133**
Fl417	**12,462**	13,886	15,532	15,492	14,087	18,688	12,476

Table 4. Average tour distance for all algorithms (third experiment)

TSP instance	HHO-ACS	ACO	PSO	GA	BH	DA	A-GWO
bays29	**9079.60**	9823.20	9195.91	10015.2	9463.25	9480.29	9390.86
bayg29	9077.20	9882.22	9947.03	9771.95	9375.44	9547.75	**9011.100**
att48	33580.2	39436.2	47018.4	43620.6	34473.8	37759.7	**33570.223**
eil51	**429.600**	461.018	574.802	453.477	458.925	475.16	457.98
berlin52	**7589.00**	8522.90	11089.5	9288.45	8455.83	9486.70	8120.13
st70	**685.200**	757.754	1321.81	1158.85	797.575	839.01	745.543
eil76	**548.600**	594.144	975.64	652.059	659.102	644.89	619.432
eil101	654.200	763.921	1499.99	838.831	897.381	997.60	**654.180**

provided in 50 runs by each algorithm according to the setting in [48]. It indicates that the proposed A-GWO algorithm produces the best results in seven TSP instances: Eil51, St70, Eil76, Pr76, KroA100, CH150, and Tsp225 (approximately 58%). It also shows that the VDWOA algorithm produced the best results in four TSP instances: Berlin52, Pr107, D198, and F1417 (about 33%). Table 3 reports that the A-GWO algorithm produces the best results because it is the best algorithm compared with DWOA, BA, GWO, MFO, and PSO algorithms. The algorithm produces better results (100%) compared with all mentioned algorithms. Table 3 illustrates the results of algorithm performance based on the optimal solution by each algorithm.

In the third scenario, an experiment was performed with six optimization algorithms to indicate the effectiveness of the proposed algorithm. The several algorithms used in the experiment are HHO-ACS, ACO, PSO, GA, BH, and DA, as reported in Table 4. The experiment is performed according to the average tour distance of each algorithm. It indicates that the A-GWO algorithm produces better results than DA and BH algorithms in all TSP instances (100%). The comparison between A-GWO and GA and PSO algorithms indicated that A-GWO produces the best results in seven TSP instances. In contrast, GA and PSO algorithms produce the best results only in one TSP instance. The performance of the A-GWO algorithm is evaluated with other swarm algorithms, particularly the ACO algorithm. A-GWO outperforms the ACO algorithm in seven TSP instances (i.e., bays29, bays29, att48, eil51, berlin52, and st70 [75%]), whereas the ACO algorithm produces the best results only in one TSP instance (i.e., eli76). The final comparison between HHO-ACS and A-GWO algorithm indicates that HHO-ACS results better than A-GWO, where the former outperforms A-GWO in five TSP instances (approximately 63%). The HHO-ACS algorithm produces the best results in bays29, eil51, berlin52, st70, and eil76, whereas the A-GWO algorithm produces the best results only in bayg29, att48, and eil101 (approximately 38%). The reason is that the HHO-ACS algorithm can optimize its parameter better than the A-GWO. Table 3 illustrates the results of algorithm performance based on the optimal solution by each algorithm. However, A-GWO and HHO-ACS have better stability, providing a standard deviation than other optimization algorithms. HHO-ACS, as indicated by the third experiment, provides small values of the standard deviation because it is the best algorithm in the ranking. The average standard deviation showed by HHO-ACS (approximately 16.6111). The second algorithm in the ranking is A-GWO, which provides a standard deviation of a value of approximately 18.3766. DA

with 89.6325 is the third algorithm in the ranking, BH with 210.7414 is the fourth, and GA with 493.2315 is the fifth. ACO has an average standard deviation of approximately 948.8546. The last algorithm that provides a high standard deviation is PSO with 1696.3588. The average standard deviation proves that the proposed A-GWO algorithm converges on the same results during the algorithm run. However, the HHO-ACS algorithm is approximately the same in each algorithm run. HHO-ACS convergence is better than A-GWO because its parameters have been optimized, forcing the algorithm to provide similar results in the search history.

The proposed algorithm produces a better minimum distance tour than all other algorithms. The modifications (i.e., the adaptive crossover operators and neighborhood search) improve the results by enhancing information sharing among the wolves during the algorithm run and intensifying the search process for more promising regions in the neighborhood of the three best wolves' locations.

6. Conclusion and future work

This study aims to solve the problem of exchange information among the leadership hierarchy in the traveling salesman problem. The proposed adaptive algorithm (A-GWO) has two contributions using the crossover operator as the first contribution and neighborhood search as the second contribution. The crossover operator allows the information to be inherited between the leadership hierarchy during the algorithm run. The neighborhood search provides different neighborhoods regions with different solutions quality during the run, which could generate several landscapes to support the algorithm in finding more solutions in the local region of the best solution. The scientific contribution of this study is to employ a linear crossover that changes during the algorithm run. The benefit is a high exploration ratio at the beginning of the run.

The other scientific contribution increases the exploitation using neighborhood search. Thus, this search is locally performed to find the global solutions in the local region based on the quality of best solutions reached. The advantage of both contributions is the trade-off between the exploration search-based and the exploitation search-based, guiding the search toward the best regions in the search space.

The limitation of the study is that it requires long convergence because of the crossover operator that is linearly exchanged through the time run. Due to the advantage of both modifications, the improvement is achieved using a crossover operator between the omega wolf and the hierarchy level (alpha, beta, and delta). Cumulative iterations with neighborhood search including displacement and pair-swap operations improve the quality of tour distance in the neighborhood region of the hierarchy level. The proposed algorithm's performance provides a better minimum tour distance among all optimization algorithms. The evaluation conducted using 25 TSP instances is differed in the number of cities against 12 state-of-the-art algorithms. The 12 algorithms are GA, ACO, PSM, GWO, VDWOA, DWOA, BA, MFO, PSO, HHO-ACS, BH, and DA. The experiment indicates that the proposed A-GWO is approximately 58% better than all algorithms, except the HHO-ACS algorithm.

Future research will focus on applying the algorithm directly to similar problems, such as VRP, and employing online parameter adaption to optimize the parameter of A-GWO to include self-adaptive and search-based strategies. Other neighborhood search operators can be tested in the proposed algorithm with other application problems, such as clustering and classification, to guide future research plans.

Conflicts of interest

The author declares no conflict of interest.

Author contributions

The main author ("Ayad") contributed to coding, implementation, discussion of results, and preparation. The co-author "Ku Ruhana Ku-Mahamud" contributed to the planning, presentation and supervision.

Acknowledgments

The author would like to thank Shatt Al-Arab University College and Universiti Utara Malaysia for supporting this manuscript financially.

References

[1] H. N. K. A. Behadili, R. Sagban, and K. R. K. Mahamud, "Hybrid ant colony optimization and iterated local search for rules-based classification", *Journal of Theoretical and Applied Information Technology*, Vol. 98, No. 4, pp. 657–671, 2020.

[2] M. Alzaqebah, S. Abdullah, and S. Jawarneh, "Modified artificial bee colony for the vehicle routing problems with time windows", *Springerplus*, Vol. 5, No. 1, 2016.

[3] D. Arimbi, A. Bustamam, and D. Lestari, "Implementation of Hybrid Clustering Based on

Partitioning Around Medoids Algorithm and Divisive Analysis on Human Papillomavirus DNA", In: *Proc. of AIP Conference Proceedings*, Vo.1825, pp. 1–8, 2017.

[4] H. Ismanto, A. Azhari, S. Suharto, and L. Arsyad, "Classification of the mainstay economic region using decision tree method", *Indonesian Journal of Electrical Engineering and Computer Science*, Vol. 12, No. 3, pp. 1037–1044, 2018.

[5] H. Almazini and K. R. K. Mahamud, "Grey Wolf Optimization Parameter Control for Feature Selection in Anomaly Detection", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 2, pp. 474–483, 2021.

[6] M. Kohli and S. Arora, "Chaotic grey wolf optimization algorithm for constrained optimization problems", *Journal of Computational Design and Engineering* , Vol. 5, No. 4, pp. 458–472, 2018.

[7] L. Xinwu, "Research on Text Clustering Algorithm Based on Improved K-means", In: *Proc. of International Conf. on Future Computer and Communication*, Vol. 4, pp. 573–576, 2010.

[8] U. Chandrasekhar and P. Naga, "Recent trends in Ant Colony Optimization and data clustering: A brief survey", In: *Proc. of International Conf. on Intelligent Agent & Multi-Agent Systems*, pp. 32–36, 2011.

[9] C. Huang, W. Huang, H. Chang, C. Yeh, and C. Tsai, "Hybridization strategies for continuous ant colony optimization and particle swarm optimization applied to data clustering", *Applied Soft Computing*, Vol. 13, No. 9, pp. 3864-3872, 2013.

[10] K. Ye, C. Zhang, J. Ning and X. Liu, "Ant-colony algorithm with a strengthened negative-feedback mechanism for con- straint-satisfaction problems", *Information Sciences*, Vol. 4, pp. 29-41, 2017.

[11] E. Bonabeau, M. Dorigo, and G. Theraulaz, "A Primer on Multiple Intelligences", *Cham: Springer*, pp. 213-250,1999.

[12] M. Worall and M. Worall, "Homeostasis in nature: Nest building termites and intelligent buildings", *Intelligent Buildings International*, pp. 87–95, 2011.

[13] S. Sopto, S. Ayon, M. Akhand, and N. Siddique, "Modified Grey Wolf Optimization to Solve Traveling Salesman Problem", In: *Proc. of International Conf. on Innovation in Engineering and Technology*, pp. 1–4, 2019.

[14] B. Anari, J. A. Torkestani, and A. M. Rahmani, "A learning automata-based clustering algorithm using ant swarm intelligence", *Expert Systems*, Vol 35, No. 6, pp. 1-26, 2018.

[15] R. Xu, J. Xu, and D. Wunsch, "A Comparison Study of Validity Indices on Swarm-Intelligence-Based Clustering", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 42, No. 4, pp. 1243–1256, 2012.

[16] S. Zhu and L. Xu, "Many-objective fuzzy centroids clustering algorithm for categorical data", *Expert Systems with Applications*, Vol. 96, pp. 230–248, 2018.

[17] A. M. Jabbar, "Rule Induction with Iterated Local Search", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 4, pp. 289–298, 2021.

[18] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: overview and conceptual comparison", *ACM Computing Surveys*, Vol. 35, No. 3, pp. 189–213, 2003.

[19] S. Mukherjee, S. Ganguly, and S. Das, "A strategy adaptive genetic algorithm for solving the travelling salesman problem", In: *Proc. of International Conf. on Swarm, Evolutionary, and Memetic Computing*, pp. 778–784, 2012.

[20] J. Sung and B. Jeong, "An Adaptive Evolutionary Algorithm for Traveling Salesman", *The Scientific World Journal*, Vol. 14, pp. 1–11, 2014.

[21] J. Mcdonnell, R. Reynolds, and D. Fogel, "Adapting crossover in evolutionary algorithms", In: *Proc. of in the Fourth Annual Conf. on Evolutionary Programming*, pp. 367-384, 1995.

[22] M. Riff and X. Bonnaire, "Inheriting parents operators: A new dynamic strategy for improving evolutionary algorithms", In: *Proc. of International Conf. on Methodologies for Intelligent Systems*, pp. 333-343, 2002.

[23] J. Gomez, "Self adaptation of operator rates in evolutionary algorithms", In: *Proc. of Genetic and Evolutionary Computation Conf.*, pp. 1162-1173 , 2004.

[24] A. C. Salinas and J. Perdomo, "Self-adaptation of genetic operators through genetic programming techniques", In: *Proc. of Genetic and Evolutionary Computation Conf.*, pp. 913-920, 2017.

[25] Y. Yang, H. Dai, and H. Li, "Adaptive genetic algorithm with application for solving traveling salesman problems", In: *Proc. of International Conf. on Internet Technology and Applications*, pp. 1–4, 2010.

[26] T. Stützle and H. H. Hoos, "MAX –MIN Ant System", *Future Generation Computer Systems*, Vol. 16, pp. 889–914, 2000.

[27] A. G, Pardo, J. Jung, and D. Camacho, "ACO-

based clustering for Ego Network analysis," *Future Generation Computer Systems*, Vol. 66, pp. 160–170, 2017.

[28] H. Menéndez, F. Otero, and D. Camacho, "SACOC:A Spectral-Based ACO Clustering Algorithm", In: *Proc. of International Conf. on Intelligent Distributed Computing*, pp. 185–194, 2014.

[29] H. Kanan, K. Faez, and S. M. Taheri, "Feature selection using Ant Colony Optimization (ACO): A new method and comparative study in the application of face recognition system", In: *Proc. of Industrial Conference on Data Mining*, pp. 63–76, 2007.

[30] P. Shunmugapriya and S. Kanmani, "A hybrid algorithm using ant and bee colony optimization for feature selection and classification", *Swarm and Evolutionary Computation*, Vol 36, pp. 27-36, 2017.

[31] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem", *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, pp. 53–66, 1997.

[32] M. Dorigo and L. M. Gambardella, "Ant colonies for the travelling salesman problem", *Biosystems*, Vol. 43, No. 2, pp. 73–81, 1997.

[33] L. Yangyang, S. Xuanjing, and C. Haipeng, "An Adaptive Ant Colony Algorithm Based on Common Information for Solving the Traveling Salesman Problem", In: *Proc. of International Conf. on Systems and Informatics*, Vol. 35, pp. 1263–1277, 2012.

[34] G. Ping, X. Chunbo, C. Jing, and L. Yanqing, "Adaptive ant colony optimization algorithm", In: *Proc. of International Conf. on Mechatronics and Control*, pp. 95–98, 2015.

[35] S. A. Yasear and K. R. K. Mahamud, "Fine-Tuning the Ant Colony System Algorithm Through Harris's Hawk Optimizer for Travelling Salesman Problem", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 4, pp. 136–145, 2021.

[36] A. Hatamlou, "Solving travelling salesman problem using black hole algorithm", *Soft Computing*, Vol. 22, No. 24, pp. 8167–8175, 2018.

[37] S. Anuar, A. Selamat, and R. Sallehuddin, "A modified scout bee for artificial bee colony algorithm and its performance on optimization problems", *Journal of King Saud University - Computer and Information Sciences*, Vol. 28, No. 4, pp. 395–406, 2016.

[38] N. Veček, S. Liu, M. Črepinšek, and M. Mernik, "On the importance of the artificial bee colony control parameter 'limit", *Information Technology And Control*, Vol. 46, No. 4, pp. 566–604, 2017.

[39] S. Mortada and Y. Yusof, "A Neighbourhood Search for Artificial Bee Colony in Vehicle Routing Problem with Time Windows", *International Journal of Intelligent Engineering and Systems*, Vol. 14, No. 3, pp. 255–266, 2021.

[40] A. Rekaby, A. Youssif, and A. S. Eldin, "Introducing Adaptive Artificial Bee Colony algorithm and using it in solving traveling salesman problem", In: *Proc. of Science and Information Conf.*, pp. 502–506, 2013.

[41] C. Laizhong, L.Genghui, W. Xizhao, L.Qiuzhen, C. Jianyong, L.Na, and L. Jian "A ranking-based adaptive artificial bee colony algorithm for global numerical optimization", *Information Sciences*, Vol. 417, pp. 169–185, 2017.

[42] Y. Wang, T. Wang, S. Dong, and C. Yao, "An Improved Grey-Wolf Optimization Algorithm Based on Circle Map", In: *Proc. of International Conf. on Machine Learning and Computer Application*, pp. 1-7, 2020.

[43] D. Karaboga and B. Gorkemli, "Solving Traveling Salesman Problem by Using Combinatorial Artificial Bee Colony Algorithms", *International Journal on Artificial Intelligence Tools*, Vol. 28, No. 1, 2019.

[44] M. Yousefikhoshbakht, "Solving the Traveling Salesman Problem: A Modified Metaheuristic Algorithm", *Complexity Journal*, Vol. 20, pp. 1-13, 2021.

[45] M. Qamar, S. Muhammad, T. Shanshan, A. Farman, A. Ammar, F. Muhammad, A. Fayadh, M. Fazal, A. Asar, and N. Alnaim, "Improvement of traveling salesman problem solution using hybrid algorithm based on best-worst ant system and particle swarm optimization", *Applied Sciences*, Vol. 11, No. 11, 2021.

[46] Y. Liu, Q. Liu, and Z. Tang, "A discrete chicken swarm optimization for traveling salesman problem", In: *Proc. of International Conf. on Physics, Mathematics and Statistics*, pp. 1-7, 2021.

[47] J. Li and M. Le, "Application of Discrete Whale Optimization Hybrid Algorithm in Multiple Travelling Salesmen Problem", In: *Proc. of Advanced Information Technology, Electronic and Automation Control Conf.*, pp. 588–595, 2019.

[48] J. Zhang, L. Hong, and Q. Liu, "An improved whale optimization algorithm for the traveling salesman problem", *Symmetry Journal*, Vol. 13, No. 1, pp. 1–13, 2021.

[49] A. Hammouri, E. Samra, M. A. Betar, R. Khalil, Z. Alasmer, and M. Kanan, "A dragonfly algorithm for solving traveling salesman problem", In: *Proc. of International Conf. on Control System, Computing and Engineering*, pp. 136–141, 2019.

[50] M. Taha, B. A. Khateeb, Y. Hassan, O. Ismail, and A. Rawash, "Solving competitive traveling salesman problem using gray wolf optimization algorithm", *Periodicals of Engineering and Natural Sciences*, Vol. 8, No. 3, pp. 1331–1344, 2020.

[51] S. Mirjalili, M. Mirjalili, and A. Lewis, "Grey Wolf Optimizer", *Advances in Engineering Software*, Vol. 69, pp. 46–61, 2014.

[52] P. Pellegrini, T. Stützle, and M. Birattari, "A critical analysis of parameter adaptation in ant colony optimization", *Swarm Intelligence Journal*, Vol. 6, No. 1, pp. 23–48, 2012.

[53] L. Manuel, M. Maur, and M. Oca, "Parameter Adaptation in Ant Colony Optimization", *Autonomous Search Journal*, Vol. 3, No. 1, pp. 191–215, 2010.

[54] C. Fan, Q. Fu, G. Long, and Q. Xing, "Hybrid artificial bee colony algorithm with variable neighborhood search and memory mechanism", *Journal of Systems Engineering and Electronics*, Vol. 29, No. 2, pp. 405–414, 2018.

[55] P. Hansen and N. Mladenović, "Variable neighborhood search", in *Handbook of Heuristics*, 2018.

[56] TSPLIB, "Symmetric traveling salesman problem (TSP)", 1995.

[57] S. Alharbi and I. Venkat, "A Genetic Algorithm Based Approach for Solving the Minimum Dominating Set of Queens Problem", *Journal of Optimization*, Vol. 6, No. 2, pp. 1–9, 2017.

[58] M. H. Akhand, P. Shill, and M. Hossain, "Producer-Scrounger Method to Solve Traveling Salesman Problem", *International Journal of Intelligent Systems and Applications*, Vol. 7, No. 3, pp. 29–36, 2015.

[59] E. Osaba, X. Yang, F. Diaz, P. L. 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.

[60] A. Helmi and A. Alenany, "An enhanced Moth-flame optimization algorithm for permutation-based problems", *Evolutionary Intelligence*, Vol. 13, No. 4, pp. 741–764, 2020.

[61] X. Xu, X. Cheng, Z. Yang, X. H. Yang, and W. L. Wang, "Improved particle swarm optimization for Traveling Salesman Problem", In: *Proc. of International Conf. on Intelligent Computing*, pp. 857–862, 2013.