Received: February 15, 2021. Revised: April 4, 2021. 136

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Fine-Tuning the Ant Colony System Algorithm Through Harris's Hawk Optimizer for Travelling Salesman Problem

Shaymah Akram Yasear[1]* **Ku Ruhana Ku-Mahamud[1]**

[1]*School of computing, Universiti Utara Malaysia, Sintok, Changlun, 06010, Malaysia*
* Corresponding author's Email: shayma.akram.yasear@gmail.com

Abstract: The ant colony system (ACS) algorithm is one of the metaheuristics in solving combinatorial optimization problems. The control parameters namely, pheromone coefficient, heuristic coefficient, decision rule, and evaporation rate of the pheromone in ACS are important in determining the performance of the algorithm. However, the initialized values of these parameters stay constant during the search process which leads to performance degradation of the algorithm. This paper aims to tune the parameters of the ACS algorithm by using Harris's hawk optimization (HHO) algorithm in solving the travelling salesman problem (TSP). The proposed hybrid algorithm is called Harris's hawk optimizer ant colony system (HHO-ACS). The final process of HHO-ACS will output the path distance. The performance of the HHO-ACS has been evaluated by using different symmetric TSP instances of various scale size. The results showed the HHO-ACS provided superior performance compared to other well-known metaheuristics namely black hole, particle swarm optimization, dragonfly, genetic and ant colony optimization algorithms. Thus, it can attain a better solution with higher accuracy. The proposed algorithm was able to achieve best known optimal solution in solving bayg29, att48 and berlin52 instances and near optimal solution in solving bays29, eil51, st70, eil76 and eil101 instances. Compared to other algorithms, the HHO-ACS algorithm showed superior performance in solving the TSP instances which indicates its effectiveness. This is possible because, in the HHO-ACS algorithm, the main control parameters of ACS algorithm namely, the pheromone coefficient, heuristic coefficient, evaporation rate of the pheromone, and decision rule were tuned according to the problem instance at hand. Thus, the HHO-ACS algorithm can be used to solve problems of travelling salesman nature with minimum customization.

Keywords: Global optimization, Combinatorial optimization, Metaheuristic, Nature-inspired algorithms, Swarm intelligence.

1. Introduction

The traveling salesman problem (TSP) assumes a set of cities separated by given distances and a salesman who must visit all cities, passing through each city exactly one and only once. The salesman must begin and end with the same city. It is a well-known combinatorial optimization problem [1-3] to find the shortest path that connects all the cities. In general, the distance notion can be replaced by other notions, such as time or money. In the same way, the city notion can be replaced by other notions like the rent or warehouse. In all cases, the goal is minimizing the cost. The TSP has applications such as X-ray crystallography, drilling of printed circuit boards and vehicle routing. The TSP has attracted researchers to design a method that is capable to solve most of its instances in a reasonable time.

The TSP is considered an NP-complete problem and thus the exact methods for solving TSP such as linear programming and branch and bound cannot achieve satisfactory solutions in a reasonable time. Furthermore, the computational complexity of a method increases exponentially with the number of cities [4]. This attracts researchers towards using metaheuristic methods in which an approximate solution can be obtained in a reasonable time. Several metaheuristics have been used to solve TSP [1-3]. The ant colony optimization (ACO) algorithm [5] is one of the most commonly used metaheuristics for solving TSP. This algorithm uses artificial ants to

simulate the foraging behaviour of the real ant. The ant colony system (ACS) algorithm is one of the most successful variants of the ACO algorithm proposed by Dorigo and Gambardella [6]. However, in this algorithm, the initial values of control parameters namely, the pheromone coefficient, heuristic coefficient, evaporation rate of the pheromone, and decision rule are staying constant during the search process. If the parameters of the ACS algorithm are not properly set, the quality of the solution will become very poor and the ACS algorithm requires a large amount of computation time [7]. Thus, instead of using static values, several methods have been proposed to tune the parameters' values in the ACS algorithm.

Pilat and White [8] proposed two approaches to adjust the heuristic coefficient, initial decision rule, and the local pheromone volatility coefficient of ACS. In the first approach, the tournament selection method has been used to select four genetic algorithms (GA) ants from the ant population. The parameter settings of each of the four selected ants are determined at each iteration and before constructing solutions by using GA. The second approach is an offline tuning mechanism based on GA. Gaertner and Clark [9] followed a similar approach proposed in [8], wherein in the early stages, GA is used. In the proposed method, a random parameter combination is used to initialize each of the ants. During the search process, the population and parameter values will change to obtain a better solution. The parameters that are adjusted are the heuristic coefficient, pheromone evaporation rate, and initial decision rule in solving TSP. In Hao et al. [10] a variant of ACS is proposed by where each ant has its parameter setting. In each iteration, the particle swarm optimization (PSO) algorithm [11] was used to modify the values of heuristic coefficient, pheromone evaporation rate, and initial decision rule parameters within a predefined range. Anghinolfi et al. [12] used a local search to adjust the values of parameters (heuristic coefficient and initial decision rule) in the ACS algorithm. The value of parameters is adjusted during the iteration based on the neighbourhood of the current values. The value of parameters is increased or reduced by a fixed amount to obtain the combinations of parameters setting. Each combination is assigned to each group of ants. In each group, the local search is performed to find the best solution. At the end of the search process, the best solution among the groups is chosen to be the best configuration of parameters. In Melo et al. [13] the same problem is solved simultaneously by several colonies of ants where different parameter settings for the pheromone coefficient, heuristic coefficient, evaporation rate of the pheromone, and initial decision rule are used by each colony. The parameters setting of the worst colony is replaced with the parameter value from the best colony by using the mutation operator. In which, the value of the same parameter in the best colony is modified by a small, uniformly random value to update the value of parameters in the worst colony. However, in this approach, the values of the parameters did not consider the state of the search process which affects the convergence of the ACS algorithm. Gomez-Cabrero and Ranasinghe [2] proposed an algorithm to adjust the parameters in the modified ACS by using the PSO algorithm. These parameters are the initial decision rule, heuristic and pheromone coefficients, evaporation rate, and the number of ants. A new parameter was introduced to represent the percentage of vertices and *the* parameter that defines the neighbourhood. Table 1 summarizes the methods that have been presented in adjusting the main control parameters of the ACS algorithm namely, the pheromone coefficient, α, heuristic coefficient, β, decision rule, q_0 and evaporation rate of the pheromone, ρ.

Table 1. Dynamic adjustment methods for the control parameters in the ACS algorithm

Reference	Parameters				Method
	α	β	q_0	ρ	
Pilat and White [8]	-	✓	✓	✓	GA
Gaertner and Clark [9]	-	✓	✓	✓	GA
Hao et al. [10]	-	✓	✓	✓	PSO
Anghinolfi et al. [12]	-	✓	✓	-	Local search
Melo et al. [13]	✓	✓	✓	✓	Multi-colony
Gomez-Cabrero and Ranasinghe [2]	✓	✓	✓	✓	PSO

Most of the proposed methods, adjust the parameters of ACS based on another optimization algorithm. However, this approach has a drawback in which the algorithm (such as PSO and GA) that are used in adjusting the parameters of ACS needs to also adjust their parameters. For example, in Gomez-Cabrero and Ranasinghe [2] and Hao et al. [10] the PSO algorithm, the values for the inertia weight, cognitive and social parameters need to be adjusted. In Pilat and White [8], the value of crossover and mutation factors play an important role in determining the performance of GA. Furthermore, most of these methods adjust some of the parameters and used constant values for the others. For example, in Anghinolfi et al. [12], the values of β and q_0 [12] are adjusted during the search process, while α is set as a

fixed value. Similarly, in other studies [8-10], only the values of β, q_0 and ρ are adjusted. However, the value of a parameter that provides good performance in solving a particular TSP instance, may not be able to achieve good solutions in solving other instances.

In Melo et al. [13], although the four control parameters of ACS have been adjusted, the proposed approach overlooked the state of search space and type of an optimization problem. This can affect the convergence of ACS algorithm. Generally, in solving an optimization problem, the optimal combination of the parameter of an algorithm is different, even for the same type of optimization problem, because the scale of the problem is not the same. Thus, these parameters should be adjusted according to the problem being solved. In previous studies, the most adjusted parameters are β and q_0. However, other parameters also have a significant impact on the performance of ACS algorithm.

In this study, a hybrid algorithm between Harris's hawk optimization (HHO) algorithm [14] and ACS is proposed. The HHO has the characteristics of few control parameters compared to other traditional algorithms such as PSO, dragonfly algorithm (DA) [15] and GA [16]. The HHO algorithm has good searching ability and has been applied to solve different optimization problems [17, 18]. Therefore, in this paper, the HHO algorithm will be used as an adaptive parameter control to adjust the parameters of the ACS algorithm which will be used to solve TSP.

In this paper, the ACS algorithm is hybridized with HHO which optimizes the parameters of ACS based on the TSP instances [19]. The hybridization between the ACS and HHO is called Harris's hawk optimizer ant colony system (HHO-ACS) algorithm. The proposed algorithm requires fewer parameters to be adjusted compared to other metaheuristics such as PSO, DA and GA. In contrast to the other metaheuristics, the ACS algorithm is mainly developed to solve combinatorial optimization problems. Thus, it can be used to solve TSP without modifications. Tuning the parameters of ACS algorithm can be considered as a continuous optimization problem. The HHO algorithm is mainly developed to solve this type of problem. Thus, this paper aims to hybridize the HHO and ACS algorithms to produce an effective method to solve TSP.

The performance of the HHO-ACS has been evaluated by using different TSP instances of various scale size. This paper organized as follow: sections 2 and 3 describe the ACS and HHO algorithms, respectively. The proposed HHO-ACS algorithms are introduced in section 4. In Section 5, experimental design is presented, followed by results and discussion. Finally, the conclusion is presented in section 6.

2. Ant colony optimization algorithm

The idea of an ACO algorithm can be visualized through a graph. In the graph, $G = (V, Z)$, where V is vertices (nodes), and Z is connections between pairs of vertices, known as edges. The weight of an edge represents the distance between its two vertices. The graph is complete if each pair of vertices is joined by an edge, that is, it contains all the possible edges, as shown in Fig. 1.

Generally, a TSP is described as an edge-weighted complete graph, where the cities are the vertices, V, and the paths between these cities are the edges. A tour of the TSP is a Hamiltonian cycle and the optimal tour is the shortest Hamiltonian cycle. In other words, the TSP is equivalent to finding a Hamiltonian cycle that has a minimum sum of weights, which are assigned to each edge. Mathematically, TSP is formulated as shown in Eq. (1) [20].

$$TSP(\pi) = \sum_{i=1}^{n} d(\pi(i), \pi(i+1)) \qquad (1)$$

where $\pi \in S_n$ of the cities. The aim is to find a feasible solution $\pi = \langle \pi(1),\ldots,\pi(n) \rangle$ that minimizes the total tour length [20]. d is the assigned distance between two cities. The ants are distributed through the cities and then probabilistically choose the next city, based on the intensity of pheromones. The higher value of the pheromones for a route, the greater the probability that the ant will choose that route. After all the artificial ants completed their routes in the graph, the length or cost of their routes is measured using a representative technique for the problem. The update of the pheromone trail is performed according to the quality of the route, which is determined according to a decision rule based on probabilities.

The ACS algorithm is developed based on the ant system algorithm. However, in the ACS algorithm,

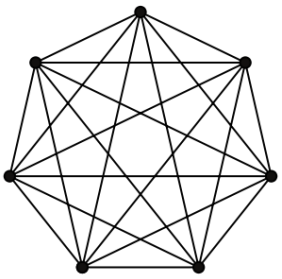

Figure. 1 A Complete graph with seven vertices

the ants adopt the state transition rules differently from the AS algorithm. In ACS, the ant *k* in node *i* moves to the next node *j* based on probability given by the pseudo-random-proportional rule, which is controlled by the parameter q_0 as shown in Eq. (2) [21].

$$j = \begin{cases} arg\ max_{l \in N_i^k} \in \{\tau_{il}[\eta_{il}]^\beta\}, \text{if } q \le q_0 \\ J \qquad otherwise \end{cases} \quad (2)$$

The transition between exploitation and exploration processes is determined according to the value of the parameter, q0. q is a uniformly distributed random number in [0,1]. If q > q0, the next node is selected based on the value of J which is a random variable selected based on the probability distribution given by Eq. (3) [21].

$$P_{ij}^k = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{ij}]^\alpha [\eta_{ij}]^\beta}, \quad \text{if } j \in N_i^k \quad (3)$$

where Pij is the probability to choose city j from city i. τ is the pheromone trail; $\eta = 1/d_{ij}$; is the inverse of the distance between the i and j cities, which represents the heuristic information. $N_i^k$is the remaining cities to be visited by ant, k, located on the city, i. The pheromone on the path of the ant is updated for solution quality improvement and the convergence speed of the algorithm. The update includes local and global pheromone updates.

2.1 Local pheromone update rule

As each ant left a certain node *i* to the next node *j*, the pheromone trail on the edge arc (*i*, *j*) is updated according to Eq. (4) [21]. This pheromone update method makes the ants more inclined in the path construction process and chooses a path different from the last one.

$$\tau_{ij} = (1 - \xi)\tau_{ij} + \xi\tau_0 \quad (4)$$

where ξ = 0.1 is the local pheromone volatility coefficient, $\xi \in$ (0, 1); τ_0=1/nC^{nn} is the initial pheromone trails on each path; *n* is the number of cities and C^{nn} is the length of a nearest-neighbour tour.

2.2 Global pheromone update rule

In ACS, the global pheromone update is performed after all ant cycles are completed and only the pheromone on the current optimal path is updated.

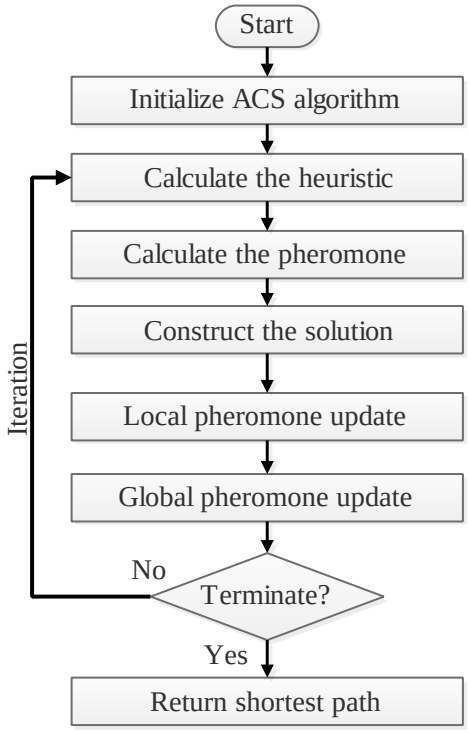

Figure. 2 Flowchart of the ACS algorithm

Using this update method will make ant's path search more targeted. The optimal path is found at the end of iterations. Eq. (5) and Eq. (6) show the formula of the global update rule [21].

$$\tau_{ij} = (1 - \rho).\tau_{ij} + \rho\Delta\tau_{ij}^{bs}, \forall(i,j) \in T^{bs} \quad (5)$$

$$\Delta\tau_{ij}^{bs} = \frac{1}{C^{bs}} \quad (6)$$

where $1 - \rho$ is the residual factor pheromone. T^{bs} is the best-so-far tour and C^{bs} is its length. Fig. 2 shows the flowchart of the ACS algorithm.

The algorithm starts by initializing parameters values. At the initial phase, *m* ants were randomly placed on *n* cities, and the ants used the state transition rules multiple times to establish a path (that is, a feasible solution for TSP). In the process of establishing paths, ants are guided by pheromone information (edges with high pheromone intensity are more attractive to ants) and heuristic information (tend to choose the shortest path). At the same time, the pheromone ants on the paths that have been visited are modified by applying local update rules. When all ants have established a complete path, the global update rule is applied to modify the amount of information on the path again. This is until the end of the entire search process.

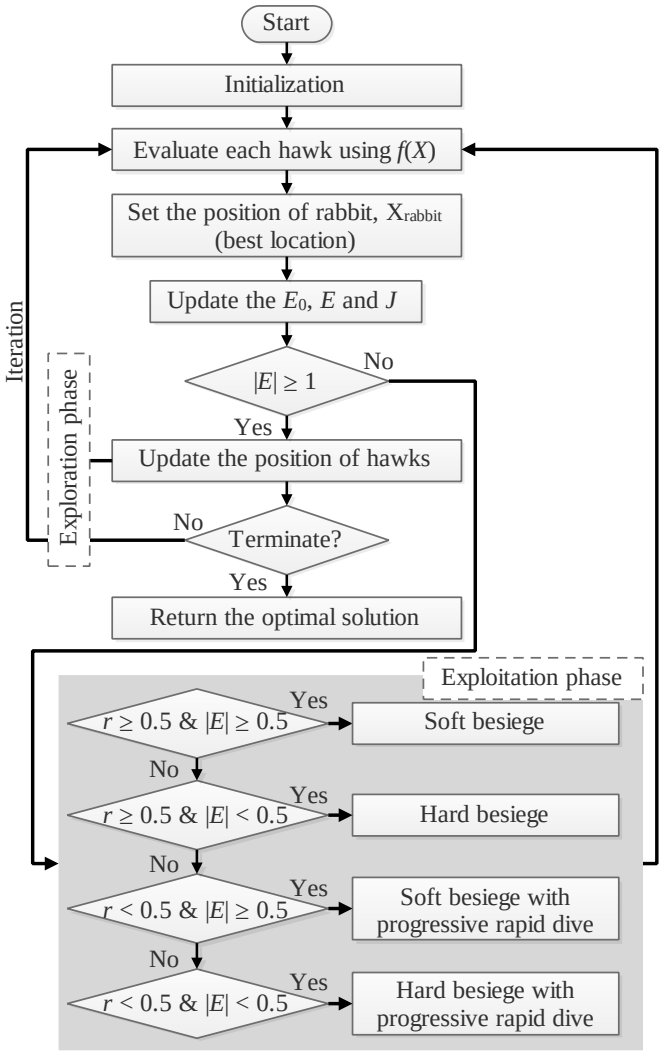

Figure. 3 Harris's hawk optimization algorithm

3. Harris's hawk optimization

Harris's hawk optimization algorithm is developed imitating the hunting behaviour of Harris's hawks in nature. Fig. 3 shows the steps of the HHO algorithm. Each hawk's position represents a candidate solution in solving the optimization problem. The prey position represents the best solution found so far.

The optimization processes of HHO include three phases, namely, exploring the search space, surprise pounce (exploitation), and different attacking strategies of Harris's hawks [14].

3.1 Exploration phase

The exploration process of the HHO algorithm includes two strategies to update the position of hawks with probability q, as shown in Eq. (7) [14].

$$X(t+1) = \begin{cases} X_{rand}(t) - r_1|X_{rand}(t) - 2r_2X(t)|; & q \geq 0.5 \\ X_{rabbit}(t) - X_m(t) - & \\ \quad r_3\big(LB + r_4(UB - LB)\big); & q < 0.5 \end{cases} \tag{7}$$

where $X(t+1)$ is the new position of a hawk. X_{rabbit}, $X(t)$, X_{rand} and X_{m} represent the positions of prey, the hawk's position at iteration t, randomly selected hawk, and the average position of hawks in the population, respectively. The coefficients r_1, r_2, r_3, r_4 and q LB and UB, represent the lower and upper bounds of the decision variables. The random values are used to increase the diversification ability of the HHO to explore different regions of the search space. The average position of hawks is calculated as shown in Eq. (8) [14].

$$X_m(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t) \tag{8}$$

where N is the total number of hawks in the population. In the HHO, the transition between exploration and exploitation is performed based on the escape energy of a prey, E. This value of escape energy linearly decreases from 2 to 0 with the iteration, as shown in Eq. (9) [14].

$$E = 2E_0\left(1 - \frac{t}{T}\right) \tag{9}$$

where T represents the maximum number of iterations. E_0 is the initial value of the energy in $(-1; 1)$]. When $|E| \geq 1$, the HHO algorithm will perform the exploration process which indicates that the prey escape in the entire solution space. Otherwise, it performs the exploitation process by searching the neighbourhood of the solutions.

3.2 Exploitation phase

The HHO algorithm describes whether the prey escaped successfully by the factor $r \in [0; 1]$. When $r < 0.5$, it indicates that the prey escaped successfully, otherwise, it failed. Whether the prey escaped or not, the hawks will perform a hard or soft besiege according to the relative amount of the escape energy with a probability of 0.5. The soft besiege occurs when $|E| \geq 0.5$, otherwise, the hard besiege will occur. Based on that, four cases were proposed to model the exploitation phase.

Case1: Soft besiege

When $r \geq 0.5$ and $|E| \geq 0.5$, it means that the energy of the prey is abundant. In this case, the prey tries to escape by performing random jumping, but eventually fails and is captured by the hawks by performing the surprise pounce. This modelled as shown in Eq (10) and Eq. (11) [14].

$$X(t+1) = \Delta X(t) - E|JX_{rabbit}(t) - X(t)| \quad (10)$$

$$\Delta X(t) = X_{rabbit}(t) - X(t) \quad (11)$$

where $\Delta X(t)$ refers to the difference between the rabbit and the hawk's current position. $J = 2(1 - r_5)$ and r_5 are random numbers in the intervals (0,2) and (0,1), respectively.

Case2: Hard besiege

when $r \geq 0.5$ and $|E| < 0.5$, indicates that the energy of the prey is low and it is directly captured by the hawks, which perform the surprise pounce. This behaviour modelled as shown in Eq. (12) [14].

$$X(t+1) = X_{rabbit}(t) - E|\Delta X(t)| \quad (12)$$

Case3: Soft besiege with progressive rapid dives

When $r < 0.5$ and $|E| \geq 0.5$, it means that the prey energy is enough to ensure successful escape, but the hawks still perform a soft besiege before the surprise pounce. This formulated as shown in Eqs (13 -15)).

$$X(t+1) = \begin{cases} Y, & \text{if } F(Y) < F(X(t)) \\ Z, & \text{if } F(Z) < F(X(t)) \end{cases} \quad (13)$$

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X(t)| \quad (14)$$

$$Z = Y - S \times LF(D) \quad (15)$$

where S is a random vector D is the dimension of the problem and LF is the levy flight function [14].

Case4: Hard besiege with progressive rapid dives

When $r < 0.5$ and $|E| < 0.5$, it means that the prey energy, E, is low. In this case, the hawk dives, and hard besieges are performed to reduce the average distance from the prey and its average position. This update is performed by Equation (13) where Z and Y are calculated using Eq. (15) and Eq. (16), respectively. The best hawk in the population is taken as the optimal solution of the current generation.

$$Y = X_{rabbit}(t) - E|JX_{rabbit}(t) - X_m(t)| \quad (16)$$

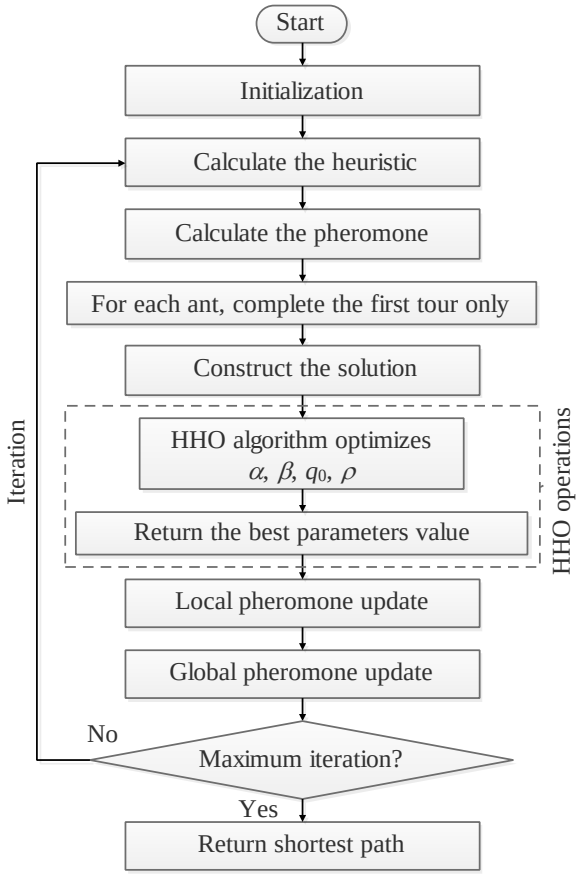

Figure. 4 Flowcharts of the HHO-ACS algorithm

4. Hybridizing the HHO with the ACS algorithm

The basic idea of HHO-ACS depends on the strengths of the two algorithms. In other words, the algorithms complement each other. The algorithm uses HHO to optimize the four control parameters (α, β, ρ, q_0) in the ACS algorithm and uses the ACS algorithm to improve the path-finding behaviour of the traditional ACS algorithm. In the proposed HHO-ACS algorithm, the tour length obtained by the ACS algorithm represents the objective function, $f(X)$, of the HHO algorithm and the position of each hawk, X, represents the values of the parameters of the ACS. This is to guide each hawk to move in a better direction which will produce a better combination of parameters. Thus, the hawks with the shortest tour length represent the best solution in the current iteration. In other words, the best combination of parameters $X = [\alpha, \beta, \rho, q_0)]$ will enable the function $f(\alpha, \beta, \rho, q_0)$ in getting the optimal path. Fig. 4 shows the flowcharts of the HHO-ACS algorithm.

In Fig. 4, the doted box includes the operations of the HHO algorithm (refer to Fig. 3), while the others represent the operations of the ACS algorithm (refer to Fig. 2). The HHO-ACS algorithm starts by randomly distributing the ants to the cities. All ants

will complete only their first tour with respect to the inter-city distances. Then, the values of α, β, ρ, q_0 parameters are obtained by using the HHO algorithm. Each hawk is evaluated by using the objective function which represents the tour length. The rabbit location represents the best parameters value, which is used to calculate the pheromone of all ants. The HHO-ACS is terminated when the maximum number of iterations designated for the ACS algorithm is reached. The best solution obtained represents the shortest path.

5. Results and discussion

Symmetric TSP instances [19] namely bays29, bayg29, att48, eil51, berlin52, st70, eil76 and eil101 have been used to evaluate the performance of the HHO-ACS algorithm. The dimension of the problems has ranged from 29 to 101 cities. For example, bays29 is a 29-city problem; eil76 is a 76-city problem, and so forth. The instances are categorized according to the number of cities i.e. small scale (<20), medium scale (20<cities<100), and large scale (>100) [22]. The performance of the HHO-ACS algorithm has been compared with the GA [16], PSO [11], DA [3], black hole (BH) [1], and ACO algorithms. The parameters settings of these algorithms are the same as the recommended settings in [1, 3]. For each algorithm, the maximum number of iterations is set to 200, and 100 is the population size. For the HHO algorithm, 10 decision variables are used to perform the search process and the maximum number of iterations is set to two. The mean, standard deviation (SD), and best and worst distances over five independent runs are recorded for comparison purposes. Table 2 shows the mean, standard deviation (SD), the best and worst distance for the eight instances over five independent runs obtained by HHO-ACS and other algorithms [1, 3] used for comparison. The best results are highlighted.

The HHO-ACS algorithm showed competitive performance compared to other algorithms, namely, GA, ACO, BH and PSO on all instances. The mean and the SD values obtained by HHO-ACS algorithm for all eight datasets are better than other algorithms, indicating that the HHO-ACS algorithm has better search ability and convergence than other algorithms. Fig. 5 shows the best path (the shortest path) obtained by HHO-ACS compared to the optimal path of each problem. Red (thick line) and blue (thin line) colours represent the optimal path and obtained path by HHO-ACS respectively in solving a TSP.

Optimal solutions in solving the bayg29, att48 and berlin52 instances were obtained by the HHO-ACS algorithm. In solving other instances, namely,

Table 2. Experimental results of algorithms on the test data sets

Problem (number of cities)	Algorithm	Best	Worst	Mean	SD
bays29 (29)	HHO-ACS	**9073.0**	**9094.00**	**9079.60**	**8.2341**
	ACO	9239.2	11014.5	9823.20	722.415
	PSO	9120.3	9498.17	9195.91	168.972
	GA	9751.4	10513.9	10015.2	319.879
	BH	9396.5	9507.17	9463.25	60.9588
	DA	9387.03	9611.78	9480.29	64.37
bayg29 (29)	HHO-ACS	**9074.0**	**9094.00**	**9077.20**	9.3915
	ACO	9447.5	11033.6	9882.22	675.833
	PSO	9329.2	11332.7	9947.03	799.407
	GA	9579.12	10411.2	9771.95	127.113
	BH	9375.44	9375.44	9375.44	**0.000**
	DA	9464.41	9704.98	9547.75	64.75
att48 (48)	HHO-ACS	**33522.0**	**33606.0**	**33580.2**	**33.5365**
	ACO	35230.9	46204.2	39436.2	4874.3
	PSO	36996.4	61421.9	47018.4	9685.89
	GA	35312.5	50671.5	43620.6	2004.00
	BH	34200.9	35528.5	34473.8	589.802
	DA	37225.9	38683.2	37759.7	425.69
eil51 (51)	HHO-ACS	**428.000**	**431.000**	**429.600**	**1.5166**
	ACO	454.39	469.053	461.018	6.2974
	PSO	469.155	737.526	574.802	107.237
	GA	448.84	462.114	453.477	9.4157
	BH	437.893	526.898	458.925	38.6365
	DA	471.58	491.65	475.16	4.51
berlin52 (52)	HHO-ACS	**7542.00**	**7657.00**	**7589.00**	**62.1088**
	ACO	7757.03	10541.1	8522.90	1152.20
	PSO	9218.47	14279.4	11089.5	2067.93
	GA	8779.76	9565.37	9288.45	1301.21
	BH	8188.07	9356.75	8455.83	508.987
	DA	9400.75	9610.15	9486.70	72.54
st70 (70)	HHO-ACS	**678.000**	**692.000**	**685.200**	**5.1672**
	ACO	711.652	855.203	757.754	59.6079
	PSO	1030.85	1756.12	1321.81	269.279
	GA	1112.31	1242.20	1158.85	52.1734
	BH	723.269	1081.11	797.575	125.227
	DA	797.47	887.08	839.01	24.28
eil76 (76)	HHO-ACS	**543.000**	**555.000**	**548.600**	**4.6152**
	ACO	574.240	665.999	594.144	40.2152
	PSO	804.267	1195.90	975.64	152.406
	GA	619.226	679.786	652.059	122.097
	BH	566.243	925.842	659.102	152.175
	DA	624.92	674.48	644.89	13.02
eil101 (101)	HHO-ACS	**640.000**	**661.000**	**654.200**	**8.3187**
	ACO	725.099	868.205	763.921	59.9684
	PSO	1158.71	1973.82	1499.99	319.749
	GA	828.881	854.438	838.831	9.9642
	BH	720.384	1249.87	897.381	210.145

	DA	812.80	997.60	898.52	47.90

bays29, eil51, st70, eil76 and eil101, the HHO-ACS was able to achieve lengths near to the optimal solutions. In the HHO-ACS, the parameters adjustment of the ACS algorithm is based on the characteristics of the TSP instance being solved [23]. The optimal combination of parameters obtained by the HHO-ACS algorithm in solving each instance of the TSP is presented in Table 3.

Table 3. Values of parameters obtained by HHO-ACS algorithm in solving TSPs

Problem	**Best Parameters value**			
	α	β	ρ	q_0
bays29	1.0687	1.9995	0.0993	0.8970
bayg29	1.0327	1.9957	0.0994	0.8957
att48	0.9822	1.9926	0.0999	0.8942
eil51	0.8929	1.9912	0.0997	0.8909
berlin52	1.1919	1.9904	0.0999	0.8907
st70	0.9419	1.9942	0.0995	0.8944
eil76	0.8848	1.9914	0.0995	0.8938
eil101	1.1288	1.9983	0.0994	0.8941

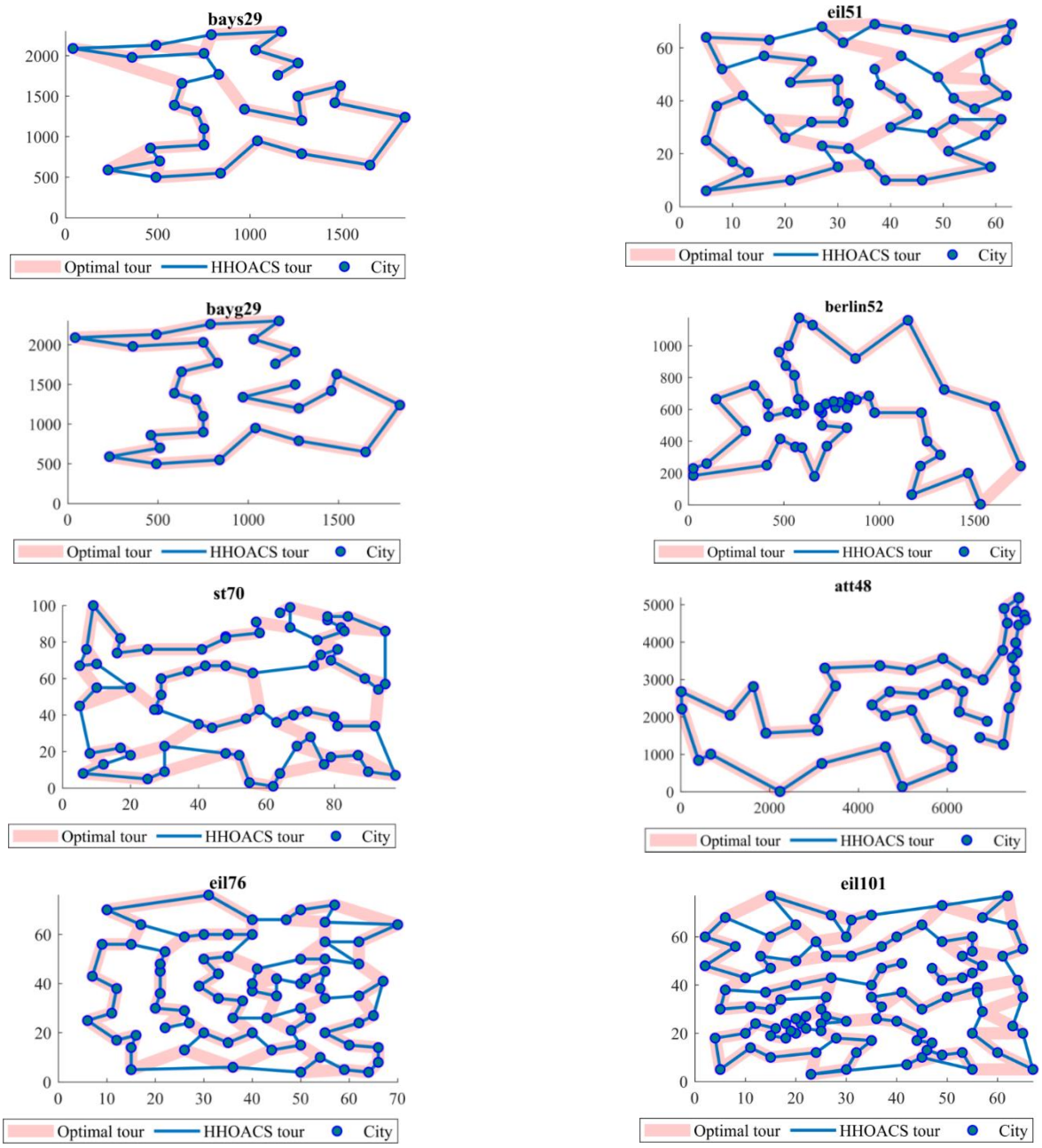

Figure 5. The TSP solutions found by the HHO-ACS algorithm on the test data

6. Conclusion

The control parameters namely, pheromone coefficient, heuristic coefficient, decision rule, and evaporation rate of the pheromone have a significant effect on the performance of the ACS algorithm. The random combination of parameters makes the algorithm fall into local optima which resulted in stagnation. To overcome this limitation, the HHO-ACS algorithm is proposed to tune the parameters according to the TSP instances. The proposed algorithm requires fewer parameters to be adjusted compared to other metaheuristics. It was able to achieve the best known optimal solution in solving bayg29, att48 and berlin52 instances and near optimal solution in solving other instances, namely, bays29, eil51, st70, eil76 and eil101. The obtained results indicate that the HHO-ACS worked well across different instances compared to other metaheuristics. For future work, it is possible to test the performance of the HHO-ACS algorithm in solving other combinatorial optimization problems of TSP nature.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Shaymah Akram Yasear; methodology, software, formal analysis, resources, data curation, writing—original draft preparation and visualization, Ku Ruhana Ku-Mahamud; writing-review, editing, and supervision.

Acknowledgments

The Higher Education Ministry of Malaysia has funded this work under the Fundamental Research Grant Scheme, FRGS/1/2017/ICT02/UUM/02/1 (S/O code 13794).

References

[1] A. Hatamlou, "Solving Travelling Salesman Problem Using Black Hole Algorithm", *Soft Computing*, Vol. 22, No. 24, pp. 8167-8175, 2018.

[2] D. Gomez-Cabrero and D. Ranasinghe, "Fine-tuning the Ant Colony System Algorithm Through Particle Swarm Optimization", In: *Proc. of the International Conference on Information and Automation*, 2005.

[3] A. I. Hammouri, E. T. A. Samra, M. A. Al-Betar, R. M. Khalil, Z. Alasmer, and M. Kanan, "A dragonfly algorithm for solving traveling salesman problem", In: *Proc. of 8th IEEE International Conference on Control System, Computing and Engineering (ICCSCE)*, Penang, Malaysia, pp. 136-141, 2018.

[4] 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, pp. 1950004, 2019.

[5] M. Dorigo and G. Di Caro, "Ant Colony Optimization: A New Meta-Heuristic", In: *Proc. of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, Washington, DC, USA, pp. 1470-1477, 1999.

[6] M. Dorigo and L. M. Gambardella, "A Study of Some Properties of Ant-Q", In: *Proc. of International Conference on Parallel Problem Solving from Nature*, Nature Berlin, Germany, pp. 656-665, 1996.

[7] S. K. Sahana, "An Automated Parameter Tuning Method for Ant Colony Optimization for Scheduling Jobs in Grid Environment", *International Journal of Intelligent Systems and Applications*, Vol. 11, No. 3, pp. 11, 2019.

[8] M. L. Pilat and T. White, "Using Genetic Algorithms to Optimize ACS-TSP", In: *Proc. of International workshop on ant algorithms*, Brussels, Belgium, pp. 282-287, 2002.

[9] D. Gaertner and K. L. Clark, "On Optimal Parameters for Ant Colony Optimization Algorithms", In: *Proc. of the 2005 International Conference on Artificial Intelligence (IC-AI)*, Las Vegas, Nevada, USA, pp. 83-89, 2005.

[10] Z.-F. Hao, R.-C. Cai, and H. Huang, "An Adaptive Parameter Control Strategy for ACO", In *Proc. of 2006 International Conference on Machine Learning and Cybernetics*, Dalian, China, pp. 203-206, 2006.

[11] J. Kennedy and R. Eberhart, "Particle swarm optimization", In: *Proc. of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, pp. 1942-1948, 1995.

[12] D. Anghinolfi, A. Boccalatte, M. Paolucci, and C. Vecchiola, "Performance Evaluation of an Adaptive Ant Colony Optimization Applied to Single Machine Scheduling", In *Proc. of Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 411-420, 2008.

[13] L. Melo, F. Pereira, and E. Costa, "MC-ANT: A Multi-Colony Ant Algorithm", In: *Proc. of International Conference on Artificial Evolution (Evolution Artificielle)*, Strasbourg, France, pp. 25-36, 2009.

[14] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris Hawks Optimization: Algorithm and Applications", *Future Generation Computer Systems*, Vol. 97, pp. 849-872, 2019.

[15] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems", *Neural Computing and Applications*, Vol. 27, No. 4, pp. 1053-1073, 2016.

[16] R. Leardi, "Genetic algorithms", *Comprehensive Chemometrics*, Vol. 1, pp. 631-653, 2009.

[17] I. M. Diaaeldin, S. H. A. Aleem, A. El-Rafei, A. Y. Abdelaziz, and M. Ćalasan, "Optimal Network Reconfiguration and Distributed Generation Allocation Using Harris Hawks Optimization", In: *Proc. of 2020 24th International Conference on Information Technology (IT)*, Zabljak, Montenegro, pp. 1-6, 2020.

[18] B. P. Sahoo and S. Panda, "Load Frequency Control of Solar Photovoltaic/Wind/Biogas/Biodiesel Generator Based Isolated Microgrid Using Harris Hawks Optimization", In: *Proc. of the 2020 First International Conference on Power, Control and Computing Technologies (ICPC2T)*, Raipur, India, pp. 188-193, 2020.

[19] G. Reinelt, "TSPLIB—A Traveling Salesman Problem Library", *ORSA journal on computing*, Vol. 3, No. 4, pp. 376-384, 1991.

[20] E. Çela, V. G. Deineko, and G. J. Woeginger, "The Multi-Stripe Travelling Salesman Problem", *Annals of Operations Research*, Vol. 259, No. 1-2, pp. 21-34, 2017.

[21] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, Cambridge: ed: MIT Press, London, England, 2004.

[22] C. Liu and A. Kroll, "On Designing Genetic Algorithms for Solving Small-and Medium-Scale Traveling Salesman Problems", In: *Proc. of International Symposium on Evolutionary Computation*, Zakopane, Poland, pp. 283-291, 2012

[23] T. Stützle, M. López-Ibáñez, P. Pellegrini, M. Maur, M. Montes de Oca, M. Birattari, M. Dorigo, *Parameter Adaptation in Ant Colony Optimization*, Springer, Berlin, Heidelberg, 2011