DOI: 10.1109/TAAI54685.2021.00019

Please cite this article as:

C. G. Tan, S. S. Choong, and L. P. Wong, "A machine-learning-based approach for parameter control in bee colony optimization for traveling salesman problem," in Proceedings of the 2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI 2021), IEEE, 2021, pp. 54–59.

A Machine-Learning-based Approach for Parameter Control in Bee Colony Optimization for Traveling Salesman Problem

Chong Gee Tan
School of Computer Sciences
Universiti Sains Malaysia
Penang, Malaysia
chonggee@student.usm.my

Shin Siang Choong
School of Computer Sciences
Universiti Sains Malaysia
Penang, Malaysia
css15_com047@student.usm.my

Li-Pei Wong*
School of Computer Sciences
Universiti Sains Malaysia
Penang, Malaysia
lpwong@usm.my
* Corresponding author

***Abstract*— Metaheuristics are a set of algorithms which is capable of solving Combinatorial Optimization Problems (COPs). When a metaheuristic is used to solve a COP, one of the major aspects is to determine an appropriate parameter setting. Poor practice of determining parameter values may lead to inability to find optimal solutions or getting invalid conclusions from the experimental results. This research proposes a machine-learning-based parameter control mechanism for a metaheuristic, i.e. the Bee Colony Optimization (BCO) algorithm. The proposed mechanism consists of three main phases: Data Collection, Model Training, and Deployment. In order to examine the performance of the BCO algorithm with the parameter control mechanism, a set of 16 TSP instances is used as test bed. The experimental results show that it is significantly better than the BCO implementation using the parameter values that are determined via a manual tuning process. The proposed parameter control mechanism overcomes the shortcomings of manual parameter tuning and dynamically adjust the parameter values throughout the BCO optimization process.**

Keywords—metaheuristic, parameter tuning, supervised learning, random forest regression, combinatorial optimization, swarm intelligence.

I. INTRODUCTION

A metaheuristic is a higher-level problem independent procedure or algorithm that is able to find a near optimal solution for Combinatorial Optimization Problem (COP). The solution found by a metaheuristic is often reasonably good, even when the metaheuristic operates with incomplete information or limited computational capacity. Metaheuristics can be classified as single-solution-based (e.g. Simulated Annealing (SA) [1, 2] and Tabu Search (TS)) [3, 4], or population-based (e.g. Genetic Algorithm (GA) [5, 6], Ant Colony Optimization (ACO) [7, 8], Artificial Bee Colony (ABC) algorithm [9-12], and Bee Colony Optimization (BCO) [13-15]). Metaheuristics are commonly used to solve different COPs such as Traveling Salesman Problem (TSP), Job Shop Scheduling Problem (JSSP), Quadratic Assignment Problem (QAP), and Vehicle Routing Problem (VRP).

The success of a metaheuristic in solving a COP highly depends on a balance between exploration and exploitation mechanisms during the search for solutions with acceptable quality. Metaheuristics owns a set of parameters needed to be tuned in order to strike a balance in terms of exploration versus exploitation. There are two common parameter setting methods in metaheuristics: parameter tuning and parameter control. Eiben and Smith [16] define parameter tuning as finding good parameter values before the execution of metaheuristic which are then kept fixed throughout an execution. On the other hand, parameter control dynamically changes parameter values in a metaheuristic based on instant feedbacks during the solution searching process.

Generally, when a metaheuristic is applied to solve a problem, the default parameter settings as recommended in the literature are utilized. As mentioned by Birattari and Kacprzyk [17], most researchers utilize a trial-and-error parameter tuning approach to determine the parameters of a metaheuristic. Besides trial-and-error approach, some scientific methods such as one factor at a time (OFAT) [18], F-Race [19] and design of experiment related methods [20-22] can also be utilized to tune the parameters. There are four major drawbacks accompanied with these parameter tuning approaches, i.e. time-consuming, labour intensive, high skill and knowledge requirements, and high risks to invalidate any result and conclusion drawn from the experiment. Very often, when the problem instance changes, parameters needed to be tuned from scratch again to better suit the new problem. In addition, these drawbacks of parameter tuning are especially apparent in design of experiment related approaches such as factorial design [13]. Assuming that each parameter is discretized and restricted to only m possible values, if there are *n* parameters, there will be *mn* combinations of parameter values needed to be tested. Therefore, parameter tuning itself can also be considered as a combinatorial optimization problem [23].

A population-based metaheuristic, namely: the Bee Colony Optimization (BCO) algorithm was proposed to solve various types of COPSs including TSP [13]. The BCO algorithm is inspired by the foraging behavior of bees. Similar to other metaheuristics, BCO also needs a suitable parameters configuration to yield good performance. This paper aims to address the limitations of the manual parameter tuning in BCO. Specifically, a Machine Learning (ML) based parameter control method which dynamically adapts to the feedbacks from solution instances during execution process is proposed as an alternative to the manual tuning method.

The organization of the paper is as follows. Section I introduces the general ideas and the research problems as well as the research objectives. Section II introduces the TSP. Section III reviews the related literatures. Section IV illustrates and describes the mechanisms of the proposed ML-based parameter control method for BCO [13]. Section V presents the experimental results and findings. Finally, Section VI ends the paper with conclusions drawn from experiment results and highlights the future work.

II. Traveling Salesman Problem

As mentioned in Section I, this paper proposes a machine-learning-based parameter control for the BCO algorithm. In order to validate the usefulness of the proposed parameter control mechanism, a combinatorial optimization problem, namely: TSP, is selected as the test bed. This section briefly describes TSP.

TSP is an NP-hard COP [24].Suppose that a number of nodes are distributed in some geometric region,. TSP can be modelled as an undirected weighted graph, $G = (E, V)$, in which E is a set of edges ($E = \{(a, b) : a, b \in V\}$) and V is a set of n nodes ($V = \{v_1, v_2, \ldots, v_n\}$). When solving a TSP, a distance matrix corresponding to E, $D = \{d_{a,b}\}$, in which $d_{a,b}$ represent the distance between city a and city b is given. Let $\prod$ represents all feasible permutations of set V. An optimal TSP solution is a permutation $\pi \in \prod$, which has the shortest possible round trip distance, as shown in Eq. (1),

$$C_{TSP}(\pi \in \prod) = \sum_{i=1}^{n-1}[d_{\pi(i),\pi(i+1)}] + d_{\pi(n),\pi(1)} \quad (1)$$

where $\pi(i) \in V$ indicates the i-th element in π.

III. Related Work

This section presents studies related to the BCO algorithm and ML-based parameter control/tuning methods. They are described in Section III.A and Section III.B respectively.

A. Bee Colony Optimization

Bees are insects with well-organized interactive behaviour. In a bee colony, each bee is assigned with different tasks, including breeding, foraging, and building hive. In order to maintain a seamless food supply chain, foraging is a one of the very crucial tasks. In a bee colony, bees explore different spots looking for new food sources. When a bee discover a new food source, it performs waggle dance as an interactive medium to inform others about the amount of nectar, distance, and direction of the new food source [25]. As a result, a number of bees which observe the dance are attracted towards the newly discovered food source. A number of algorithms have been inspired by this foraging behaviour for solving a large variety of COPs.

The initial BCO model [26] modelled the foraging behaviour of bees as its solution construction mechanism by conducting a stepwise state transition rule which comprises two components: heuristic distance and arc fitness. A bee begins its foraging process with probabilistically selecting a waggle dance as a preferred path. When the bee constructs a solution, the node which followed the choice of the preferred path is assigned with greater arc fitness and therefore it is more likely to be picked as the next node to be visited. On the other hand, nearer nodes from the current node is more likely to be chosen due to the impact of the heuristic distance component. In addition, the waggle dance duration is monitored using a linear function which consists of three parameters: individual profitability score of a bee, average profitability score of the bee colony, and the scaling factor [26].

This BCO algorithm is then enriched with a local search [27], a Fragmentation State Transition Rule (FSTR) [28], a fuzzy-based dance mechanism [15] and pruning strategies [29-31]. The developed BCO algorithm is a generic framework which is able to solve multiple COPs e.g. symmetric TSP [27, 29], asymmetric TSP [32], JSSP [14, 33, 34], QAP [13], and sequential ordering problem [35]. The pseudocode of the BCO algorithm is shown in Algorithm I.

Algorithm I. Bee Colony Optimization (BCO) algorithm

```
Procedure BCO
    initialization()
    while stop criteria are not fulfilled do
        for all bees in a population do
            observeDance()
            constructSolutionByFSTR()
            performLocalSearch()
            performDance()
        end while
    end while
end Procedure BCO
```

The BCO algorithm described in Algorithm I is equipped with a set of user defined parameters as follows. λ, α, and β are the three parameters defined in the bees' path construction mechanism. K is a scaling parameter that controls the

magnitude of a dance duration. N_{Bee} denotes the number of bees (i.e. population size) being used in the BCO algorithm. ϖ denotes the capacity of past memory associated with each bee. The ML-based parameter control mechanism proposed in the paper focuses on the parameters defined in the bees' path construction mechanism (i.e. λ and β). More details are presented in Section IV.

B. ML-based Parameter Tuning/Control Methods

There are a large variety of parameter tuning and control methods in the literature. In this section, some existing metaheuristic parameters tuning and control methods based on ML are highlighted. A summary is provided in Table I.

TABLE I. PREVIOUS WORK ON ML-BASED PARAMETER TUNING AND PARAMETER CONTROL FOR METAHEURISTICS

Approaches		**ML model used**	**Metaheuristic(s)**
Parameter Tuning	Silc, et al. [36]	Decision Tree	Differential Ant-Stigmergy Algorithm (DASA)
	Aoun, et al. [37]	Hidden Markov Model	Particle Swarm Optimization (PSO)
	Pereira, et al. [38]	Case-based Reasoning	Tabu Search (TS), Genetic Algorithm (GA), Simulated Annealing (SA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO)
Parameter Control	Lessmann, et al. [39]	Non-linear Regression	Particle Swarm Optimization (PSO)

For parameter tuning, Silc, et al. [36] tested 5000 different parameter settings of the Differential Ant-Stigmergy Algorithm (DASA) and modelled the relationship between the parameter settings and performance using a decision tree. Aoun, et al. [37] utilized a Hidden Markov Model to evaluate the performance of each tested parameter configuration of Particle Swarm Optimization (PSO) over multiple test instances. Pereira, et al. [38] employed a case-based reasoning model. For a given new problem (case), the best metaheuristic and respective parameters that have been successfully used to solve similar cases are retrieved from the case base.

Although parameter tuning approaches are able to determine a set of parameter setting for a metaheuristic, Eiben and Smith [16] suggest that there are no 'one size fits all' approach in setting parameter values because metaheuristics are comprised of dynamic and adaptive processes. Different stages of the optimization process require different balance of exploration and exploitation, and thus might need different choices of parameters to efficiently navigate through the search space [40, 41]. Therefore, relying only on parameter tuning strategy is insufficient for setting up the optimal search environment. In a recent work by Lessmann, et al. [39], an ML-based parameter control approach was proposed. Lessmann, et al. [39] suggested that the features of a solution instance are correlated to the parameter values and hence a number of models that fit the features of a solution instance to the parameter values are developed using several ML techniques, i.e. Support Vector Machine (SVM), Multiple Linear Regression (MLR), and Random Forest (RF). These techniques are evaluated using a water supply network planning problem which the decision variables are continuous in nature. The results conclude that non-linear relationship exists between features of solution instances and suitable parameter values.

Inspired by Lessmann, et al. [39], in this paper, the use of a non-linear regression approach, namely Random Forest in controlling BCO parameters is investigated. There are two major differences as compared with Lessmann, et al. [39] as follows. First, in terms of the metaheuristic optimization algorithm, Lessmann et al. control the PSO parameters using a set of ML techniques whereas the proposed method is used to control the BCO parameters. Second, in terms of the optimization datasets used in performance evaluation, Lessmann et al. apply the parameter control mechanism on a set of continuous optimization problems (i.e. water supply network planning problem) whereas the proposed method is tested using a number of TSP benchmark instances which involve a discrete search space. The following section describes the proposed ML-based parameter control method for BCO.

IV. PROPOSED WORK

As mentioned in Section III.A, the bee foraging behaviour is modelled as the solution construction mechanism of the BCO algorithm [26]. Before a bee starts the solution construction process, it probabilistically select a dance performed by others to determine a preferred path, denoted as θ. θ denotes a tour that was previously explored by another bee. By referring to θ as guidance, the bees iteratively construct solutions by traveling from one city to another city according to a state transition probability, $P_{ij,t}$, which represents the probability to travel from node i to node j at time t. $P_{ij,t}$ is a function consists of two elements, namely: heuristic distance d_{ij}, and arc fitness $\rho_{ij,t}$ of the connecting edge between the two cities, as shown in Eq. (2).

$$P_{ij,t} = \frac{[\rho_{ij,t}]^{\alpha} \cdot \left[\frac{1}{d_{ij}}\right]^{\beta}}{\sum_{j \in A_{i,t}} \left([\rho_{ij,t}]^{\alpha} \cdot \left[\frac{1}{d_{ij}}\right]^{\beta} \right)} \quad (2)$$

where $A_{i,t}$ represents a set of cities which are yet to be selected at time t, α is a binary variable that turns on or off the influence of arc fitness, β controls the significance level of heuristic distance, and $\rho_{ij,t}$ is defined in Equation (3):

$$\rho_{ij,t} = \left\{ \begin{array}{c} \lambda, j \in G_{i,t}, |A_{i,t}| > 1 \\ \frac{1-\lambda|A_{i,t} \cap G_{i,t}|}{|A_{i,t} - G_{i,t}|}, j \notin G_{i,t}, |A_{i,t}| > 1 \\ 1, |A_{i,t}| > 1 \end{array} \right\} \forall j \epsilon A_{i,t}, 0 \leq \lambda \leq 1 \quad (3)$$

$G_{i,t}$ represents the city which the bee prefers to move from city i at time t as recommended by θ and λ is a parameter determining the likelihood of selecting the city recommended by θ. The first two conditions indicate that the edge recommended by θ (if there is one) is given a likelihood score of λ, while equal likelihood score is assigned to the remaining edges. Based on the third condition, when there is only one node left in $A_{i,t}$, its likelihood score is set to 1.

This stepwise state transition rule allows bees to add one city in a given step until a complete TSP solution is constructed. This solution construction mechanism is a computationally expensive and it is not scalable when solving a TSP with large dimension. The stepwise state transition rule was modified in [28] such that it is less expensive in terms of computational time. In the modified solution construction mechanism, i.e. fragmentation state transition rule, a bee could add a batch/fragment of nodes in each step during the solution construction process, instead of adding only one node in each step.

ALGORITHM II. BEE COLONY OPTIMIZATION (BCO) ALGORITHM

```
1 Procedure BCO-Control
2   B = initialization()
3   while iteration < BC_MAX do
4     for ∀b_i ∈ B do
5       θ = b_i.observeDance()
6       if iteration < interval
7         λ, β = randomizeParam()
8       else
9         λ, β = model.predict(θ)
10      end if
11      b_i.constructSolutionByFSTR(λ, β, θ)
12      b_i.performLocalSearch()
13      if iteration ≠ 0 and Cost(b_i') < Cost(θ)
14        D ∪ (λ, β, θ)
15      end if
16      if iteration % 1000 == 0
17        model = RF.train(D)
18        D = ∅
19      end if
20      b_i.performWaggleDance()
21      Conduct dance elitism
22      Reset memory if stcuked in local optimum
23    end for
24    iteration++
25  end while
26 end Procedure BCO-Control
```

In this paper, it is aimed to control the parameters of the fragmentation state transition rule (i.e. λ and β) in the BCO algorithm. Algorithm II shows the pseudocode of BCO integrated with the proposed parameter control mechanism (denoted as BCO-Control).

The proposed machine-learning-based parameter control method consists of three main phases: Data Collection, Model Training, and Deployment. During the Data Collection phase, the training data is collected during the execution of the BCO algorithm. In the initial period of the execution process, the two parameters λ and β are randomized and sampled within specified range: $0.0 \leq \lambda \leq 1.0$ and $0.0 \leq \beta \leq 10.0$ (Line 6-7 in Algorithm II). The purpose of employing randomized parameters is to sample various parameter values and identify good values with regards of different situations of the BCO algorithm execution process. When a suitable parameter setting for a situation is identified (e.g. a bee improves its preferred path with the parameter setting), the preferred path and its corresponding parameter values are appended to a set D, which is a training set for the subsequent Model Training phase (Line 13-15 in Algorithm II). The accumulated training data (i.e. set D) are periodically sent for regression model training to predict suitable parameter values. In this study, the Model Training is performed after every 1000 iterations (Line 16-19 in Algorithm II).

TABLE II. A FEATURE REPRESENTATION EXAMPLE

Solutions	AB	AC	AD	BC	BD	CD
ACBDA	0	1	1	1	1	0
ADCBA	1	0	1	1	0	1

To ensure that the training time is affordable, the following pre-processing steps are performed to keep the size of the dataset (i.e. the number of columns and rows) in an acceptable range. Specifically, the top N (i.e. $N = 25$) most frequent edges presented in the collected dataset will be captured and used as Boolean features to represent bee solutions. When a solution contains/does not contain an edge, E, its value in the corresponding column representing E is set at one/zero. Consider a TSP with four cities, i.e. A, B, C, and D, Table II illustrates the feature representation of two solutions with $N = 6$. The main purpose of using such feature extraction approach is to capture the edge information of a TSP solution while maintaining a reasonable number of columns. Moreover, a deduplication process over the rows is performed on the collected dataset. For every row with the same features, only the one with lowest parameter value will be kept for subsequent model training. The rationale of introducing a bias towards the lowest parameter values is the assumption of parameter setting that is able to improve the solution with least influence of the arc fitness and heuristic distance, is the most suitable parameter setting.

In the proposed ML-based parameter control method, the Random Forest algorithm [42] is employed to model the relationship between the solution instance features and parameters. Random Forest is a decision-tree-based ensemble model. The pseudocode of the Random Forest algorithm is presented in Algorithm III [42]. In this study, the default parameter setting of Random Forest in Weka [43] is utilized.

ALGORITHM III. RANDOM FOREST

```
procedure randomForest
  forest = ∅
  for i ∈ 1, …, ntree do
    d_i = A bootstrap sample from the training dataset
    tree_i = treeConstruction(d_i)
    forest = forest ∪ tree_i
  end for
  return forest
end procedure

procedure treeConstruction(d)
  for each node in tree do
    Split based on the best feature in a subset of all features
  end for
  return tree
end procedure
```

This ensemble model is chosen because a previous study [39] has proved that Random Forest is able to map the non-linear relationships in the dataset for parameter control task. In the model training process, 70% of the dataset is employed for training, while 30% is utilized for validation. The model performance is validated using two main indicators: correlation coefficient which tells how many percent of the data variance is captured by the model, and root mean squared error which is the square root of total deviations between predicted values with the validation dataset. The set D is cleared after each Model Training phase.

The Deployment phase follows the Model Training phase. In the Deployment phase, the values of λ and β used by each bee are guided by the predictions from the trained regression models (Line 9 in Algorithm II). Simultaneously, another cycle of Data Collection phase is started to collect new datasets for the subsequent Model Training phase. The BCO algorithm and its machine-learning-based parameter control mechanism repeat until either the algorithm has found the known optimal value or reached the predefined iteration limit.

V. Experimental Results

This section presents the empirical results of this study. It includes the experimental setup, result indicator and the experimental results.

A. Experiment Setup

In order to assess the effectiveness and performance of the machine-learning-based parameter control for the BCO algorithm, a total of 16 symmetric TSP instances are chosen from TSPLIB [44] with their dimension ranging from 136 to 1291 cities. The dimension of the problem instance is reflected in the numerical figure of the instance name, e.g. PR136 is a 136-city TSP; D1291 is a 1291-city TSP.

The results were benchmarked against the work by Wong [13] (denoted as BCO-Fixed), which uses fixed parameter values determined by a manual tuning process. The parameter used by BCO-Fixed is as follows: $\alpha = 1$, $\beta = 1$, $\lambda = 0.1$, $N_{Bee} = 50$, and $BC_{Max} = 10000$.

B. Results Indicators

NetBeans IDE version 8.2 and Weka 3.8.3 were used to implement the BCO algorithm and its machine-learning-based parameter control mechanism. Weka 3.8.3 supports all ML-related functions, i.e. the Random Forest implementation. The experiments were conducted on a workstation running Ubuntu operating system with Intel Core™ i7-3930K 3.20 GHz hexa-core processor and 16 GB memory.

There are two main indicators to evaluate the performance of BCO solving TSP problems: percentage deviation from optimum tour length δ (measured in %), and computational time to achieve the best tour length possible, T (measured in seconds). The formula for the calculation of δ is as shown in Eq. (4):

$$\delta = \frac{\mu_C - C^*}{C^*} * 100\% \quad (4)$$

C^* is the known optimum for the problem instance, i.e. the shortest possible tour length found by the research community so far. Ten replications are conducted for the experiment. A set of results $C = [c_1, c_2, \ldots, c_{10}]$ and $T = [t_1, t_2, \ldots, t_{10}]$ are recorded, in which C is a set of generated shortest tour lengths while T is the time taken to achieve the corresponding shortest tour lengths. From the results, the average of set C and T are calculated and denoted as μ_C and μ_T respectively. The δ_{avg} indicator denotes the percentage difference of the best result from the known optimum on average.

C. Experimental Results

Table III shows the performance comparison between BCO-Fixed and BCO-Control. Table III reports the instance names, the known optimal tour length, average best tour length obtained by the algorithms (μ_C), deviation of μ_C from known optimum, and average time taken to obtain C (μ_T).

Based on Table III, the average δ_{avg} scores for BCO-Fixed and BCO-Control are 0.437% and 0.363%, respectively. The largest δ_{avg} scores for BCO-Fixed and BCO-Control (i.e. 0.964% and 0.862% respectively) are both obtained when solving the PCB1173 instance.

BCO-fixed solves the 16 TSP instances to 0.437% from the known optimum within 1498.6 seconds (≈25 minutes). On the other hand, BCO-Control solves the instances to 0.363% from the known optimum within 1358.3 seconds (≈22.6 minutes). In other words, BCO-Control averagely obtains 16.9% better solutions with 9.4% (or 2.4 minutes) shorter computational time as compared with BCO-Fixed.

TABLE III. Performance comparison between BCO-Fixed and BCO-Control algorithms based on 16 TSP benchmark problems

TSP Instances	Known Optimum	BCO-Fixed			BCO-Control		
		μC	δ_{avg}, %	μT, seconds	μC	δ_{avg}, %	μT, seconds
PR136	96772	96857.2	0.088	92.8	96794.0	0.024	105.9
LIN318	42029	42104.3	0.179	545.0	42081.4	0.125	714.8
RD400	15281	15322.9	0.274	592.7	15313.9	0.215	780.5
GR431	171414	172267.3	0.498	736.5	172207.4	0.463	619.5
PR439	107217	107301.9	0.079	983.7	107285.3	0.064	729.7
D493	35002	35188.2	0.532	1316.1	35171.0	0.483	1283.6
ATT532	27686	27790.3	0.377	1412.0	27783.7	0.353	842.4
ALI535	202339	202663.0	0.160	1339.3	202585.7	0.122	1032.2
U574	36905	37167.4	0.711	1297.8	37091.0	0.504	1435.1
D657	48912	49221.5	0.633	1529.5	49210.6	0.610	1083.3
GR666	294358	296134	0.603	1679.3	295821.8	0.497	2017.8
U724	41910	42212.7	0.722	1207.1	42177.7	0.639	2030.1
SI1032	92650	92650.0	0.000	463.0	92650.0	0.000	622.7
VM1084	239297	241027.1	0.723	3992.5	240526.5	0.514	2226.5
PCB1173	56892	57440.3	0.964	3590.9	57382.6	0.862	3027.3
D1291	50801	51026.1	0.443	3199.3	50969.6	0.332	3181.6
Average:			0.437	1498.6		0.363	1358.3

To compare the overall performance of BCO-Fixed and BCO-Control statistically, the Wilcoxon signed-rank test with a significance level of 0.05 is utilized. The μ_T values of all problem instances are scaled (i.e. all the values are between zero and one) such that the priorities of all instances are standardized. The differences between each pair of performance indicators (i.e. δ_{avg} and normalized μ_T) achieved by BCO-Fixed and BCO-Control are computed. After excluding those instances with zero differences, these differences are ranked in ascending order. In Table IV, N represents the effective sample size (i.e. number of TSP instances) after removing those with zero differences, R^- represents a summation of the ranks for the TSP instances where BCO-Control underperforms BCO-Fixed, and R^+ represents those for the TSP instances where BCO-Control are better than BCO-Fixed. A critical value, $W_{cri,N}$ acquired from a statistical table is employed to compare with the test statistic, W [45]. $W > W_{Cri,N}$ suggest that no statistical difference between the performance of BCO-Fixed and BCO-Control is observed, while $W \leq W_{Cri,N}$ implies a statistical difference between the performance of BCO-Fixed and BCO-Control. According to Table IV, BCO-Control has significantly better δ_{avg} scores as compared with BCO-Fixed ($W \leq W_{Cri,N}$ and $R^+ > R^-$). The zero R^- value implies that BCO-Control performs better than BCO-Fixed in terms of δ_{avg} in solving all 16 TSP instances. In terms of μ_T, BCO-Control and BCO-Fixed are on par with each other ($W > W_{Cri,N}$). In other words, the proposed ML-based parameter control method achieves better solutions while maintains similar computational time.

TABLE IV. Results of the Wilcoxon signed-rank test

	δ_{avg}	Normalized μ_T
N	15	16
R^+	120	77
R^-	0	59
W	0	59
$W_{Cri,N}$	19	23
Significant	yes	no

VI. Conclusions

In this study, a machine-learning-based parameter control mechanism is integrated as part of the BCO algorithm (BCO-Control). BCO-Control has been tested on a set of 16 TSP benchmark instances. The empirical results suggest that BCO-Control has successfully controlled the two BCO parameters (i.e. λ and β) and guides the BCO algorithm to find better TSP tour length compared to the BCO-Fixed which uses parameters with fixed values. The results coincide with the findings by Lessmann, et al. [39] and proves that such machine-learning-based parameter control implementation is applicable to BCO for solving a discrete COP, i.e. TSP. The proposed machine-learning-based parameter control mechanism is able to address the drawbacks of manual parameter tuning and adapt different parameter values at different stages throughout the optimization process.

For future work, application of similar approach to other metaheuristics can be investigated. As a TSP solution can be represented using a graph, the use of graph embedding [46] in the feature extraction process can be explored. Also, application of Reinforcement Learning [47] and AutoML [48] for parameter control in metaheuristic can be studied.

Ackownledgement

This work was supported by Tabung Persidangan Luar Negara (TPLN). The authors would like to thank the Universiti Sains Malaysia for the awarded grant.

References

[1] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671-680, 1983.

[2] H. Wang, K. Li, and W. Pedrycz, "A routing algorithm based on simulated annealing algorithm for maximising wireless sensor networks lifetime with a sink node," *International Journal of Bio-Inspired Computation*, vol. 15, pp. 264-275, 2020.

[3] Z. Wei and J.-K. Hao, "Multistart solution-based tabu search for the Set-Union Knapsack Problem," *Applied Soft Computing*, vol. 105, p. 107260, 2021.

[4] F. Glover and M. Laguna, "Tabu search," in *Handbook of Combinatorial Optimization*, D.-Z. D. P. M. Pardalos, Ed., ed: Springer, 1998, pp. 2093-2229.

[5] Ü. Çavuşoğlu and A. H. Kökçam, "A new approach to design S-box generation algorithm based on genetic algorithm," *International Journal of Bio-Inspired Computation*, vol. 17, pp. 52-62, 2021.

[6] S. Mirjalili, "Genetic algorithm," in *Evolutionary Algorithms and Neural Networks*, ed: Springer, 2019, pp. 43-55.

[7] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE computational intelligence magazine*, vol. 1, pp. 28-39, 2006.

[8] W. Deng, J. Xu, Y. Song, and H. Zhao, "An effective improved co-evolution ant colony optimisation algorithm with multi-strategies and its application," *International Journal of Bio-Inspired Computation*, vol. 16, pp. 158-170, 2020.

[9] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Applied soft computing*, vol. 8, pp. 687-697, 2008.

[10] S. S. Choong and L. P. Wong, "A Hyper-heuristic based Artificial Bee Colony Optimization for the Traveling Salesman Problem," in *Big Data Summit 2: HPC and AI Empowering Data Analytics (co-locate with PRAGMA 35 Meeting)*, 2018.

[11] M. Yazdani and N. J. Navimipour, "Join query optimisation in the distributed databases using a hybrid harmony search and artificial bee colony algorithm," *International Journal of Bio-Inspired Computation*, vol. 17, pp. 189-198, 2021.

[12] S. S. Choong, L. P. Wong, and C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the Traveling Salesman Problem," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 357-362.

[13] L. P. Wong, "A generic bee colony optimization framework for combinatorial optimization problems," PhD thesis, School of Computer Engineering, Nanyang Technological University, 2012.

[14] W. M. Choo, L. P. Wong, and A. T. Khader, "A modified bee colony optimization with local search approach for job shop scheduling problems relevant to bottleneck machines," *International Journal of Advanced Soft Computing Applications*, vol. 8, pp. 52-78, 2016.

[15] S. S. Choong, L. P. Wong, and C. P. Lim, "A dynamic fuzzy-based dance mechanism for the bee colony optimization algorithm," *Computational Intelligence*, vol. 34, pp. 999-1024, 2018.

[16] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing* vol. 53: Springer, 2015.

[17] M. Birattari and J. Kacprzyk, *Tuning metaheuristics: A machine learning perspective*, second edition ed. vol. 197: Springer Publishing Company, Incorporated, 2009.

[18] B. Akay and D. Karaboga, "Parameter Tuning for the Artificial Bee Colony Algorithm," in *International Conference on Computational Collective Intelligence*, Berlin, Heidelberg, 2009, pp. 608-619.

[19] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle, "F-Race and Iterated F-Race: An Overview," in *Experimental Methods for the Analysis of Optimization Algorithms*, T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 311-336.

[20] M. Fallahi, S. Amiri, and M. Yaghini, "A parameter tuning methodology for metaheuristics based on design of experiments," *International Journal of Engineering and Technology Sciences*, vol. 2, pp. 497-521, 2014.

[21] F. Dobslaw, "A parameter tuning framework for metaheuristics based on design of experiments and artificial neural networks," in *International Conference on Computer Mathematics and Natural Computing*, 2010.

[22] S. S. Choong, L. P. Wong, and C. P. Lim, "An artificial bee colony algorithm with a modified choice function for the traveling salesman problem," *Swarm and Evolutionary Computation*, vol. 44, pp. 622-635, 2019.

[23] X.-S. Yang, S. Deb, M. Loomes, and M. Karamanoglu, "A framework for self-tuning optimization algorithm," *Neural Computing and Applications*, vol. 23, pp. 2051-2057, 2013.

[24] M. Jünger, G. Reinelt, and G. Rinaldi, "The traveling salesman problem," *Handbooks in operations research and management science*, vol. 7, pp. 225-330, 1995.

[25] K. Von Frisch, "Decoding the language of the bee," *Science*, vol. 185, pp. 663-668, 1974.

[26] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm for traveling salesman problem," in *Proceedings of the Second Asia International Conference on Modeling & Simulation*, Kuala Lumpur, 2008, pp. 818-823.

[27] L. P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," in *Proceedings of the 6th IEEE International Conference on Industrial Informatics*, Daejeon, 2008, pp. 1019-1025.

[28] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm with the fragmentation state transition rule for traveling salesman problem," in

Proceedings of the Conference on Innovative Production Machines and Systems (IPROMS 2009), Cardiff, UK, 2009, pp. 399-404.

[29] S. S. Choong, L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimisation algorithm with a sequential-pattern-mining-based pruning strategy for the travelling salesman problem," *International Journal of Bio-Inspired Computation*, vol. 15, pp. 239-253, 2020.

[30] L. P. Wong and S. S. Choong, "A bee colony optimization algorithm with frequent-closed-pattern-based pruning strategy for traveling salesman problem," in *2015 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, 2015, pp. 308-314.

[31] L. P. Wong, M. Y. H. Low, and C. S. Chong, "An efficient bee colony optimization algorithm for traveling salesman problem using frequency-based pruning," in *2009 7th IEEE International Conference on Industrial Informatics*, 2009, pp. 775-782.

[32] L. P. Wong, A. T. Khader, M. A. Al-Betar, and T. P. Tan, "Solving Asymmetric Traveling Salesman Problems using a generic Bee Colony Optimization framework with insertion local search," in *Proceedings of the 13th International Conference on Intelligent Systems Design and Applications (ISDA 2013)*, Bangi, 2013, pp. 20-27.

[33] L. P. Wong, M. Y. H. Low, and C. S. Chong, "Solving job shop scheduling problems with a generic bee colony optimization framework," in *Proceedings or the International Conrerence on Industrial Engineering and Systems Management. International Institute for Innovation, Industrial Engineering and Entrepreneurship (I4e2)*, 2011, pp. 269-280.

[34] L.-P. Wong, C. Y. Puan, M. Y. H. Low, Y. W. Wong, and C. S. Chong, "Bee colony optimisation algorithm with big valley landscape exploitation for job shop scheduling problems," *International Journal of Bio-Inspired Computation*, vol. 2, pp. 85-99, 2010.

[35] M. H. Wun, L. P. Wong, A. T. Khader, and T. P. Tan, "A bee colony optimization with automated parameter tuning for sequential ordering problem," in *Proceedings of the Fourth World Congress on Information and Communication Technologies (WICT 2014)*, Bandar Hilir, 2014, pp. 314-319.

[36] J. Silc, K. Taškova, and P. Korošec, "Data mining-assisted parameter tuning of a search algorithm," *Informatica*, vol. 39, pp. 169-176, 2015.

[37] O. Aoun, M. Sarhani, and A. El Afia, "Investigation of hidden markov model for the tuning of metaheuristics in airline scheduling problems," *IFAC-PapersOnLine*, vol. 49, pp. 347-352, 2016.

[38] I. Pereira, A. Madureira, P. B. de Moura Oliveira, and A. Abraham, "Tuning Meta-Heuristics Using Multi-agent Learning in a Scheduling System," in *Transactions on Computational Science XXI: Special Issue on Innovations in Nature-Inspired Computing and Applications*, M. L. Gavrilova, C. J. K. Tan, and A. Abraham, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 190-210.

[39] S. Lessmann, M. Caserta, and I. Montalvo, "Tuning metaheuristics: A data mining based approach for particle swarm optimization," vol. 38, pp. 12826-12838, 2011.

[40] S. S. Choong, L. P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Information Sciences*, vol. 436, pp. 89-107, 2018.

[41] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, pp. 309-325, 2014.

[42] A. Liaw and M. Wiener, "Classification and regression by randomForest," *R news*, vol. 2, pp. 18-22, 2002.

[43] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, pp. 10-18, 2009.

[44] G. Reinelt, "TSPLIB—A traveling salesman problem library," *ORSA journal on computing*, vol. 3, pp. 376-384, 1991.

[45] F. Wilcoxon, S. K. Katti, and R. A. Wilcox, "Critical values and probability levels for the wilcoxon rank sum test and the wilcoxon signed rank test," in *Selected Tables in Mathematical Statistics*. vol. 1, H. L. Harter and D. B. Owen, Eds., ed Providence: American Mathematical Society, 1970, pp. 171-259.

[46] H. Cai, V. W. Zheng, and K. C. Chang, "A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 1616-1637, 2018.

[47] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*: MIT press, 2015.

[48] X. He, K. Zhao, and X. Chu, "AutoML: A Survey of the State-of-the-Art," *Knowledge-Based Systems*, vol. 212, p. 106622, 2021.