The IEEE International Conference on
Industrial Informatics (INDIN 2008)
DCC, Daejeon, Korea July 13-16, 2008

Bee Colony Optimization with Local Search for Traveling Salesman Problem

[i] Li-Pei Wong, [ii] Malcolm Yoke Hean Low, [iii] Chin Soon Chong
[i,ii] School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, SINGAPORE 639798.
[iii] Singapore Institute of Manufacturing Technology, 71 Nanyang Drive, SINGAPORE 638075.
[i] wonglipei@pmail.ntu.edu.sg, [ii] yhlow@ntu.edu.sg, [iii] cschong@SIMTech.a-star.edu.sg

Abstract- **Many real world industrial applications involve finding a Hamiltonian path with minimum cost. Some instances that belong to this category are transportation routing problem, scan chain optimization and drilling problem in integrated circuit testing and production. This paper presents a Bee Colony Optimization (BCO) algorithm for Traveling Salesman Problem (TSP). The BCO model is constructed algorithmically based on the collective intelligence shown in bee foraging behaviour. The model is integrated with 2-opt heuristic to further improve prior solutions generated by the BCO model. Experimental results comparing the proposed BCO model with existing approaches on a set of benchmark problems are presented.**

I. INTRODUCTION

Traveling Salesman Problem (TSP) is about finding a Hamiltonian path with minimum cost. It is common in areas such as logistics, transportation and semiconductor industries. For instance, finding an optimized scan chains route in integrated chips testing, parcels collection and sending in logistics companies, are some of the potential applications of TSP. Efficient solution to such problems will ensure the tasks are carried out effectively and thus increase productivity. Due to its importance in many industries, TSP is still being studied by researchers from various disciplines.

Some behaviors in animals have the potential to be adapted to solve TSP. An example will be the highly coordinated patterns shown in bee [1] and ant [2] foraging behaviour. Ants deposit a chemical substance known as pheromone along the way between their colony and food source. Bees perform waggle dance upon returning to their hive when they found a food source. Pheromone trail and waggle dance are used as a communication medium among individuals in ant or bee colony. These individuals usually perform their actions (depositing pheromone or performing waggle dance) locally with limited knowledge about the entire system. However, when their local actions are combined, they will emerge to produce global effects such as directing more individuals to the new discovery of food source.

We have previously described a Bee Colony Optimization (BCO) algorithm for TSP in [3]. The model is based on bee foraging behaviour where bees are used to generate feasible solutions for TSP benchmark problems found in TSPLIB[1]. Our initial work [3] showed that the original BCO algorithm to TSP could be improved further. In this paper we report our work in integrating the BCO model with 2-opt local search heuristic so that the BCO algorithm is able to obtain better solutions for selected TSP problems.

This paper starts with a discussion on TSP in Section II. Section III explains the 2-opt heuristic. A discussion on the BCO model is presented in Section IV. Section V discusses the implementation details of the BCO algorithm for TSP. Experiments and results are presented in Section VI. Finally, this paper ends with conclusion and future works.

II. TRAVELING SALESMAN PROBLEM (TSP)

Suppose a salesman is given a set of cities associated with traveling distances (or costs) from any city to any other city. The salesman is required to make a round-trip tour (the salesman must visit every city only once and then return to the starting city) with minimum distances (or costs). The TSP is therefore to determine a Hamiltonian tour with minimum cost. TSP is one of the discrete optimization problems which is classified as NP-hard [4].

Generally, TSP can be denoted via graph notations as follows:

- Let $G = (V, E)$ be a graph.
- V is a set of m cities, $V = \{v_1, \ldots, v_m\}$.
- E is a set of arcs or edges, $E = \{(r, s): r, s \in V\}$.

E is normally associated with a distance (or cost) matrix, which is defined as $D = (d_{r,s})$. If $d_{r,s} = d_{s,r}$, the problem is a symmetric TSP (STSP). Otherwise, it becomes an asymmetric TSP (ATSP).

In TSP, the objective is to determine the permutation π of set V that minimizes the total roundtrip distance as shown in Equation (1). Overview and fundamental concepts about TSP can be found in [5,6].

$$C(\pi) = \sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \qquad (1)$$

Various techniques have been used to solve TSP. In general, these techniques are classified into two broad categories: exact and approximation algorithms [4]. Exact algorithms are methods which utilize mathematical models whereas approximation algorithms make use of heuristics and iterative improvements as the problem solving process. Some instances

[1] www.informatik.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

in the exact methods category are Branch and Bound, Lagrangian Relaxation and Integer Linear Programming.

Approximation algorithms can be further classified into two groups: constructive heuristics and improvement heuristics. Instances in constructive heuristics include Nearest Neighbourhood, Greedy Heuristics, Insertion Heuristics [7], Christofides Heuristics [8] etc. Instances in improvement heuristic include *k*-opt [9], Lin-Kernighan Heuristics [10,11], Simulated Annealing [12], Tabu Search [13], Evolutionary Algorithms [14-16], Ant Colony Optimization [17,18] and Bee System [19]. In the next section, a local improvement heuristic for TSP based on simple tour modifications is presented.

III. Local Search: 2-opt Heuristic

Improvement heuristics have been used widely in solving many combinatorial optimization problems. Generally, the heuristics under this category consist of the following steps:

1. Produce a pseudorandom feasible solution, *R*.
2. Perform a transformation on *R* to produce *R'*.
3. Replace *R* with *R'* if *R'* is found to be better than *R*. Repeat step 2 until no improvement is observed. At this stage, *R* is said to be locally optimal.
4. Repeat step 1 to step 3 until a pre-defined computation time is exceeded or when a satisfactory result is gained.

In the context of TSP, *R* can be viewed as a permutation of cities where $R \in V_\pi$. A feasible *R* has to fulfill the objective function *C* which is described in Section II. A transformation mechanism is conducted in order to obtain a better *R*. Once $C(R')<C(R)$ is found, *R* is replaced with *R'*. The transformation is repeated until *R* is locally optimal.

A similar framework has been applied in this paper. However, the BCO model is used to generate a set of feasible solutions rather than using a pseudorandom approach. These generated solutions are further improved by using a local search approach, namely the 2-opt heuristic. Both approaches are integrated to solve TSP. In this section, the 2-opt heuristic is presented. The BCO algorithm that generates a set of feasible solutions will be discussed in the subsequent section.

2-opt heuristic is a method that is frequently applied to solve TSP. Some advantages of applying 2-opt heuristic in TSP are the simplicity in its implementation and its ability to obtain near optimal results. The basic idea of 2-opt heuristic is to eliminate two arcs in *R* in order to obtain two different paths. These two paths are then reconnected in the other possible way. Let's consider a feasible solution, *R*, with the permutation of "A, B, C, D, E, F, A" with total tour length of 8 units as shown in Fig. 1(a). This closed tour is then further transformed by firstly eliminating two arcs (B, C) and (E, F) and thus producing two separate paths "F, A, B" and "C, D, E". Next, these two paths are then reconnected to produce another closed tour, *R'* = "A, B, E, D, C, F, A", as shown in Fig. 1(b). Note that there is only one possible way to reconnect these two paths in order to preserve the length of the other four arcs. By performing a two-arc transformation, the total length of the closed tour is reduced from 8 units to 6 units. The 2-opt transformation mechanism can be generalized to cater for more arcs in its eliminate-and-reconnect process. *k-opt* [9] is the general terminology for this approach.

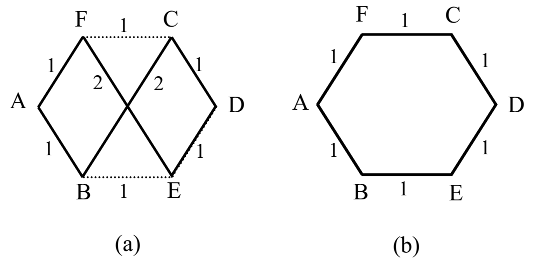

Fig. 1. (a) Original closed tour, R. (b) R' after 2-opt transformation

IV. The BCO Model with Local Search

This section first describes the natural foraging model of a typical bee colony. Next, an overview of how the foraging model is used in constructing feasible paths for TSP is included. Focus will be given to the arc fitness mechanism in the state transition rule. At the end of this section, a BCO algorithm for TSP is presented.

A. Bee Colony

The foraging behaviour in a bee colony remains mysterious for many years until von Frisch translated the language embedded in bee waggle dances [1]. Waggle dance operates as a communication tool among bees. Suppose a bee found a rich food source. Upon its return to the hive, it starts to dance in a figure-eight pattern. This figure-eight dance consists of a straight waggle run followed by a turn to the right back to the starting point, and then another straight waggle run followed by a turn to the left and back to the starting point again. The bee usually repeats these for a few times. Remarkably, via this informative dance, the bee has actually informed its hive mates about the direction and distance of the food source. The direction is expressed via the angle of dance relative to the sun position whereas the distance is expressed via the length of the straight waggle run. This coding and decoding process will eventually bring more bees towards the new food discovery. Some discussions about waggle dance can be found in [20,21].

B. Path Construction in Artificial Bee

In our proposed model, a bee is allowed to explore and search for a complete tour path. Before leaving the hive, the bee will randomly observe dances performed by other bees. The bee is then equipped with an ordered set of moves which are observed from the dance. This set of moves, named as "preferred path" and denoted as θ, will then serve as guidance in its foraging process. θ contains a complete tour that had been explored previously by its mate and it will direct the bee towards the destination. It is one of the permutation π of set V as described in Section II, $\theta \in V_\pi$.

During the foraging process, a bee will travel from one city to another city until it reaches the destination. In the bee model, a heuristic transition rule is employed to aid the bee in its decision making on which city to visit next. This rule consists of two factors: arc fitness and heuristic distance. The arc fitness is computed for all possible paths to cities that can be visited by a bee from a particular city at a particular time. A higher fitness value is assigned to the arc which is part of the preferred path. By doing this, a bee tends to choose the next visiting city based on the preferred path. On the other hand, under the heuristic distance influence, a bee tends to choose the next visiting city which is nearest to its current city.

An example of a bee visiting five cities under the influence of arc fitness will be presented. Let's assume five fully connected cities, denoted by $V = \{1, 2, 3, 4, 5\}$, are to be visited. The bee is assumed to start foraging from its hive, which is denoted as city H. The bee is equipped with a preferred path, θ = "3, 2, 1, 4, 5, 3", which is observed via a dance by another bee before leaving the hive. The hive and the rest of the cities are equidistance from each other. By having this equidistance assumption, a bee is biased towards visiting the first city in the θ. This simplifies the example as heuristic distance will have no effect in the decision making in this example. The influence of heuristic distance in our bee model will be discussed in the next section.

When a bee is in city i after n transitions, two sets of next visiting cities can be derived, namely the set of "allowed next cities", $A_{i,n}$ and the set of "preferred next city", $F_{i,n}$. $A_{i,n}$ is defined as a set of cities that can be reached from city i at transition n. $F_{i,n}$ is a set that contains one city which the bee prefers to move to from city i at transition n as recommended by θ. Let $\theta(m)$ denotes the m-th element in θ. If a bee has just started its exploration from the hive, $F_{\mathrm{H},0}=\{\theta(1)\}$. If the current visiting city is $\theta(q)$ after n transitions, then $F_{\theta(q),n}=\{\theta(q+1)\}$.

When $n = 0$, a bee leaves its hive and decides the first city to visit. Based on θ = "3, 2, 1, 4, 5, 3", two sets of cities are identified: $A_{\mathrm{H},0} = \{1, 2, 3, 4, 5\}$ and $F_{\mathrm{H},0} = \{3\}$. All the arcs that connect the hive and the cities in $A_{\mathrm{H},0}$ are assigned a fitness value: λ is assigned to arc (H, 3) and $(1-\lambda)/4$ is assigned to arcs (H, 1), (H, 2), (H, 4) and (H, 5).

The fitness values at transition n from city i can be summarized in an arc fitness matrix, $\Phi_{i,n}$ with dimension 1 x $|A_{i,n}|$. Each entry $\Phi_{i,n}$ is denoted by $\rho_{ij,n}$, which indicates the fitness value assigned to the arc from city i to city j where $j \in A_{i,n}$. The following matrix represents arc fitness values from the hive to city $j \in A_{H,0}$ when $n = 0$:

$$\Phi_{H,0} = [\rho_{H1,0} \quad \rho_{H2,0} \quad \rho_{H3,0} \quad \rho_{H4,0} \quad \rho_{H5,0}]$$
$$= \begin{bmatrix} \frac{1-\lambda}{4} & \frac{1-\lambda}{4} & \lambda & \frac{1-\lambda}{4} & \frac{1-\lambda}{4} \end{bmatrix}$$

λ represents the probability of following a city in the preferred path. Notice that the arc fitness matrix is associated with the following property as shown in Equation (2).

$$\sum_{j \in A_{i,n}} \rho_{ij,n} = 1 \qquad (2)$$

When $n = 1$ and let's assume city 3 is chosen in the previous stage: $A_{3,1} = \{1, 2, 4, 5\}$ and $F_{3,1} = \{2\}$. Likewise, the arc fitness of all cities in $A_{3,1}$ is shown as below:

$$\Phi_{3,1} = [\rho_{31,1} \quad \rho_{32,1} \quad \rho_{34,1} \quad \rho_{35,1}]$$
$$= \begin{bmatrix} \frac{1-\lambda}{3} & \lambda & \frac{1-\lambda}{3} & \frac{1-\lambda}{3} \end{bmatrix}$$

The idea of the mechanism is to ensure that the city suggested by the preferred path has a higher probability (λ in this case) to be chosen while the rest of the arcs share the probability of 1-λ.

Let's assume when $n = 1$, city 5 is chosen. When proceeding to $n = 2$, $A_{5,2} = \{1, 2, 4\}$ and $F_{5,2} = \{3\}$. Notice that city 3 suggested by the preferred path is no longer feasible as it has been visited at $n = 0$. Therefore, all the arcs that connect city 5 with the cities in $A_{5,2}$ will be assigned the probability of 1/3 as shown in the following arc fitness matrix:

$$\Phi_{5,2} = [\rho_{51,2} \quad \rho_{52,2} \quad \rho_{54,2}]$$
$$= \begin{bmatrix} 1/3 & 1/3 & 1/3 \end{bmatrix}$$

C. BCO Algorithm with 2-opt for TSP

Various bee colony optimization models have been attempted in different domains ranging from dynamic server allocation for internet hosting center [22], numerical function optimization [23], hex game playing program [24], job shop scheduling [25,26] to TSP [3,19]. In [19], Lucic and Teodorovic proposed a Bee System (BS) to solve TSP. In their model, bees are used to construct partial tours in every iteration. The position of the hive will also be reallocated to a new city after every iteration. Interaction among bees in their model is implemented in two ways: memory and waggle dance. Bees are able to memorize how many bees visited certain links in the past and are able to advertise partial tours to other hive mates. Upon initiating a new iteration, the solution is improved by 2-opt, 3-opt and modified 3-opt heuristics.

```
procedure BCO
   Initialize_Population( )
      while stop criteria are not fulfilled do
         while all bees have not built a complete path do
            Observe_Dance( )
            Forage_ByTransRule( )
            Perform_2-Opt( )
            Perform_Waggle_Dance ( )
         end while
      end while
end procedure BCO
```

Fig. 2. BCO with 2-opt local search for TSP

In this section, we described our proposed BCO algorithm with local search algorithm for TSP. The outline of our BCO algorithm is shown in Fig. 2.

We note the following distinctions between our proposed model and [19]:

- Bees in our proposed model do not have the ability to remember the number of bees that have visited an arc.

- Bees in our proposed model show the entire feasible path rather than partial tours via the waggle dance.
- The bee hive in our proposed system is assumed to have equal distance to all cities.
- Bees are influenced by both arc fitness and distance between cities when constructing solutions.

In our proposed model, a group of bees is created during the initial stage. The number of bee, N_{Bee}, is equal to the total number of cities in the TSP problem instance. Subsequently, foraging process is initiated. Bees are allowed to explore and exploit the search space to construct a feasible path for TSP. When all the bees have completed building a path, it is considered a complete bee cycle. As bees have no dance to follow during the first bee cycle, they are allowed to explore the search space in random. This will allow diversification in the search. The initialization phase can be modified to restrict bees to construct paths based on certain constructive heuristics listed in Section II such as Nearest Neighbourhood, Greedy Heuristics and Insertion Heuristics.

As mentioned in Section IV.B, a bee is aided by a state transition rule in making its decision to choose the next visiting city. The state transition probability, $P_{ij,n}$, gives the likelihood to move from city i to city j after n transitions. It is a function of the distance between cities and of the arc fitness present on the connecting edge. Formally, it is defined in Equation (3):

$$P_{ij,n} = \frac{[\rho_{ij,n}]^{\alpha} \cdot [\frac{1}{d_{ij}}]^{\beta}}{\sum_{j \in A_{i,n}} [\rho_{ij,n}]^{\alpha} \cdot [\frac{1}{d_{ij}}]^{\beta}} \qquad (3)$$

where $\rho_{ij,n}$ is the arc fitness from city i to city j after n transitions and d_{ij} represents the distance between city i and city j. Note that the $P_{ij,n}$ is inversely proportional to the city distance. In other words, the shorter the distance, the higher is the likelihood of that city to be selected. α is a binary variable that turns on or off the arc fitness influence in the model. β is to control the significant level of heuristic distance.

Arc fitness, $\rho_{ij,n}$, is defined as in Equation (4). The mechanism of arc fitness in path construction has been discussed in Section IV.B. $|A_{i,n} \cap F_{i,n}|$ is 1 when there is a common instance in both $A_{i,n}$ and $F_{i,n}$, or 0 otherwise. $A_{i,n} - F_{i,n}$ denotes the difference between sets $A_{i,n}$ and $F_{i,n}$. It contains all elements of $A_{i,n}$ that are not present in $F_{i,n}$. When there is only one city left in $A_{i,n}$, $\rho_{ij,n}$ is set to 1 to indicate that the city is the only choice. This happens at the last transition before a bee re-visits the start city in order to complete the tour.

$$\rho_{ij,n} = \begin{cases} \lambda & , j \in F_{i,n}, |A_{i,n}| > 1 \\ \frac{1-\lambda|A_{i,n} \cap F_{i,n}|}{|A_{i,n} - F_{i,n}|} & , j \notin F_{i,n}, |A_{i,n}| > 1 \\ 1 & , |A_{i,n}| = 1 \end{cases} \left. \begin{matrix} \forall j \in A_{i,n}, \\ 0 \le \lambda \le 1 \end{matrix} \right. \qquad (4)$$

After a bee has built a complete path according to the transition rule, the path will then be improved by the 2-opt heuristic as mentioned in Section III. The 2-opt heuristic is applied repeatedly until it reaches 2-opt optimal where the tour length shows no further improvement.

After the 2-opt operation, a dance will be performed by the bee to other hive mates according to the following policy: bees that construct shorter path compared to its previous trials are allowed to dance. In other words, not all bees are allowed to dance upon returning to hive. A bee has to find a shorter tour length compared to its previous best path in order to dance. Thus, the bees in our proposed model have the ability to remember their best path obtained previously. By using this policy, it is possible that no bee will be dancing eventually. This might happen when no bee could find a better path compared to its previous trials. To avoid this, another policy applies: if there is no bee dancing for ten continuous bee cycles, bees' memory will be "refreshed" to hold a value that is ten percent higher than its best path length. This policy can be considered as a negative feedback mechanism in homeostasis, an instance of which is hormones regulation in human body [27]. The aim of having this self-regulated mechanism in the BCO model is to maintain a balance and stable situation such that waggle dances are performed constantly.

A waggle dance will last for certain duration, determined by a linear function. The waggle dance duration of bee i, D_i, is defined in Equation (5):

$$D_i = K \cdot \frac{Pf_i}{Pf_{colony}} \qquad (5)$$

$$Pf_i = \frac{1}{L_i} \quad , \quad L_i = tour\ \ length \qquad (6)$$

$$Pf_{colony} = \frac{1}{N_{Bee}} \sum_{i=1}^{N_{Bee}} Pf_i \qquad (7)$$

where K denotes the waggle dance scaling factor, Pf_i denotes the profitability score of bee i as defined in Equation (6) and Pf_{colony} denotes the bee colony's average profitability as in Equation (7) and is updated after each bee completes its tour.

Pf_i can be interpreted as the quantity of the nectar collected by bee i, where a bee is assumed to be able to collect as much nectar as possible. Higher quantity of nectar will be collected if a bee travels along a shorter route. Therefore, Pf_i is defined to be inversely proportional to the tour length.

TABLE I
LOOKUP TABLE FOR ADJUSTMENT OF P_{follow}

Profitability Scores	P_{follow}
$Pf_i < 0.95Pf_{colony}$	0.80
$0.95Pf_{colony} \le Pf_i < 0.975Pf_{colony}$	0.20
$0.975Pf_{colony} \le Pf_i < 0.99Pf_{colony}$	0.02
$0.99Pf_{colony} \le Pf_i$	0.00

Before a bee leaves the hive, it will decide if it will observe and follow the dance shown by previous dancer with a probability of P_{follow}. The probability P_{follow} is adjusted dynamically according to the profitability score of the bee and the colony based on the lookup table as shown in Table I, which is adopted from [22,26]. In the extreme case where P_{follow} is zero, the bee will keep to its own path.

A bee is more likely to randomly observe and follow a waggle dance if its profitability rating is low when compared to the average profitability of the colony. Although a bee tends to be influenced either by its own or other bee's experience, it may still wander off from its preferred path occasionally as indicated in Equation (3).

V. Implementation Details

The BCO algorithm described in this paper is developed using JAVA with NetBeans IDE 5.5 as the development tool. A list of paths is created to represent the dances by bees. Each waggle dance is associated with a duration given by Equation (5) which is measured in terms of bee cycles. The duration will determine how long the dance will be kept in the list. The list will be refreshed and updated after every bee cycle so that expired dances will be discarded.

Our implementation utilizes the combination of random exploration and the Nearest Neighbourhood heuristic to generate initial solutions during the first bee cycle. These two approaches have equal likelihood to be employed by a bee. Random exploration is implemented in such a way that a bee starts from a random city, and chooses the next city to visit by using the state transition rule discussed in Section IV.C. Using Nearest Neighbourhood, a bee starts from an arbitrary city and then chooses the nearest city to move ahead. Random selection is applied if there is a tie in distance.

VI. Experiments and Results

This section describes the benchmark problems, the benchmark algorithms and experimental results in our comparison study.

A. Benchmark Problems

The performance of the BCO model is investigated by applying the algorithm to benchmark problems taken from TSPLIB. 20 problem instances are chosen from A, ATT, BERLIN, EIL, KRO, LIN, PR, ST and TSP series. The dimension of the problems ranges from 48 to 318 cities. The numerical figure appears in the name of the problem instance denotes the dimension of the problem. For example, ATT48 is a 48-city problem; LIN318 is a 318-city problem and so forth. Table II lists all the 20 problem instances which are considered in this paper.

B. Benchmark Algorithms

Seven other benchmark algorithms are included in our comparison study. They are: Ant Colony System (ACS) [17], MAX-MIN Ant System (MMAS) [18], Genetic Algorithms (GA) [14,15], Hybrid of GA with distance preserving crossover (HGA) [16], Lin-Kernighan heuristic (Lin-Ker) [11,28], Bee System (BS) [19] and Bee Colony Optimization (BCO) [3]. The results reported in these papers will be discussed in Section VI.C. Note that three variations of bee system were reported in [19]: bee system with 2-opt, 3-opt and modified 3-opt. However, only bee system with 2-opt and modified 3-opt are included in the table as the results for 3-opt and modified 3-opt are identical.

We also carried out a comparison between our proposed BCO model and a random approach. In the random approach, bees are allowed to construct a path randomly. When a bee needs to decide the next city to visit, it uses a random rule with the transition probability defined in Equation (8). Based on this random rule, the probability $P_{ij,n}$ of visiting each city in $A_{i,n}$ is equal.

$$P_{ij,n} = \frac{1}{|A_{i,n}|}, \quad \forall j \in A_{i,n} \tag{8}$$

C. Results

This section shows the results based on the BCO algorithm discussed in Section IV. Since the BCO algorithm described in this paper is a stochastic algorithm, the results reported in this study are the averages of five replications. We also report the best result of all five replications. All results are stated in terms of percentage deviation from known optimal value (O_i) for each problem instance. For best case scenario, the percentage deviation is defined as $(Y_i-O_i)/O_i$ whereas for average case scenario, it is defined as $(Z_i-O_i)/O_i$. Y_i denotes the best value and Z_i denotes the average value over five runs for each problem instance. Table II, III, IV, V summarize the experiment results in this paper.

All the experiments were run on a Windows cluster with four units of IBM BladeCenter HS21. Each unit is equipped with two Intel Xeon Quad Core E5335 processors (2.0GHz). Although the cluster has a total of 32 Intel Xeon Core processors, the experiments were distributed such that each run was handled by one processor only. After performing a series of parameter tunings, the parameter settings throughout the experiments are: BC_{Max} = 10000, N_{Bee} = total number of cities, α = 1, β = 10, λ = 0.95, K = 100. To illustrate the effect of the 2-opt operation, another set of experiments which is based on the original BCO model report in [3] is conducted with identical parameter settings.

Table II shows the comparison between the original BCO and BCO+2-opt on the benchmark problems. The table is sorted in ascending order according to the dimension of the problems. We note that significant improvement is obtained after incorporating 2-opt heuristic in BCO model. The 2-opt heuristic is performed for at most 50 passes after each path construction by a bee. In other words, the heuristic is not performed until 2-optimal. For the best case scenario, the BCO+2-opt algorithm achieves optimal result for all problem instances. For the average case of five replications, the algorithm obtains optimal result for all problem instances except for LIN318. The time to obtain the best solution, t_{Best}, is also showed in the table.

To investigate the performance of both algorithms on these 20 problem instances, the mean of the percentage deviations for the best case (μ_{best}) and average case ($\mu_{average}$) is showed in Table II. For the original BCO algorithm, on average, μ_{best} is

0.40% and $\mu_{average}$ is 0.92% from optimal. In contrast, the μ_{best} and $\mu_{average}$ for BCO+2-opt [3] is 0.00% and 0.005%.

TABLE II
PERFORMANCE COMPARISON OF BCO AND BCO+2-OPT ON SELECTED BENCHMARK PROBLEMS IN TSPLIB

Problem Instances	Optimal	BCO [3]			BCO+2-opt		
		Avg. (%)	Best (%)	t_{Best} (sec)	Avg. (%)	Best (%)	t_{Best} (sec)
ATT48	10628	0.28	0.00	37	0.00	0.00	0
EIL51	426	0.42	0.23	179	0.00	0.00	0
BERLIN52	7542	0.00	0.00	16	0.00	0.00	0
ST70	675	0.80	0.59	336	0.00	0.00	2
EIL76	538	0.63	0.00	310	0.00	0.00	2
PR76	108159	2.45	1.37	395	0.00	0.00	55
KROA100	21282	0.65	0.23	687	0.00	0.00	8
KROB100	22141	0.71	0.51	686	0.00	0.00	26
KROC100	20749	0.56	0.00	308	0.00	0.00	16
KROD100	21294	0.56	0.00	524	0.00	0.00	4
KROE100	22068	0.89	0.24	687	0.00	0.00	31
EIL101	629	1.34	0.16	697	0.00	0.00	67
LIN105	14379	0.21	0.00	472	0.00	0.00	2
KROA150	26524	1.70	0.61	1543	0.00	0.00	502
KROB150	26130	2.22	1.66	1542	0.00	0.00	184
KROA200	29368	0.25	0.05	20941	0.00	0.00	1986
KROB200	29437	1.44	0.41	18069	0.00	0.00	14981
TSP225	3916	0.79	0.23	22868	0.00	0.00	2987
A280	2579	0.22	0.08	48947	0.00	0.00	2974
LIN318	42029	2.24	1.72	73771	0.09	0.00	149064
Average:		0.92	0.40	9650.75	0.005	0.00	8644.55

Table III summarizes the results of random approach on the 20 problems. The random approach is discussed in Section V.B. As shown in Table III, the μ_{best} and $\mu_{average}$ is 523.68% and 533.65% after 50000 iterations, which is far from optimal. When the iteration is increased to 100000, the random approach shows little improvement where μ_{best} drops 4.48% and $\mu_{average}$ drops 3.88%. When the random approach is integrated with 2-opt, μ_{best} and $\mu_{average}$ drop significantly. However, when the random approach is compared to BCO+2-opt as shown in Table II, BCO+2-opt still outperforms Random+2-opt especially for problems with large dimension.

Table IV highlights the comparison of BCO+2-opt with the other six approaches. Three benchmark problems are selected for the comparison study: EIL51, KROA100 and LIN318. The dimensions of these problems are 51, 100 and 318 cities respectively. The results show that BCO+2-opt is able to out-perform BS+2-opt and is comparable to other approaches.

Table V shows a comparison between BCO+2-opt and BS. This comparison is conducted as both approaches show high similarity in their nature. They both utilize foraging behaviours of bee with local search for TSP. From the table, we see that BCO+2-opt outperforms both BS variants by obtaining optimal solutions for all the problems. As shown in the improvement achieved by BS with 3-opt over 2-opt, we predict that integrating our BCO algorithm with 3-opt will allow optimal solutions to be obtained for problems with larger dimension.

TABLE III
PERFORMANCE OF RANDOM APPROACH ON SELECTED BENCHMARK PROBLEMS IN TSPLIB

Problem Instances	Random (%) BCMAX=50000		Random (%) BCMAX=100000		Random+2-opt (%) BCMAX=10000	
	Avg.	Best	Avg.	Best	Avg.	Best
ATT48	204.97	197.23	198.17	193.55	0.00	0.00
EIL51	177.42	168.54	170.94	161.74	0.00	0.00
BERLIN52	181.74	174.85	178.66	172.89	0.00	0.00
ST70	301.33	288.59	296.59	289.63	0.00	0.00
EIL76	255.06	249.26	254.42	247.96	0.30	0.19
PR76	301.03	293.61	292.73	283.14	0.00	0.00
KROA100	503.16	491.47	494.52	483.62	0.00	0.00
KROB100	472.25	462.18	469.64	457.61	0.01	0.00
KROC100	515.36	512.64	500.40	482.27	0.00	0.00
KROD100	478.42	471.39	474.00	463.01	0.24	0.07
KROE100	481.32	457.32	484.65	476.98	0.12	0.02
EIL101	327.60	324.01	329.83	325.28	1.34	0.79
LIN105	546.68	535.04	540.26	522.17	0.00	0.00
KROA150	672.44	657.18	667.30	663.06	0.80	0.67
KROB150	676.47	664.76	668.36	646.46	0.65	0.51
KROA200	846.29	836.90	840.16	833.49	1.16	1.02
KROB200	817.89	808.58	823.87	810.84	1.85	1.66
TSP225	789.10	778.98	787.93	772.80	3.19	2.94
A280	1012.16	1002.40	1012.93	999.57	3.11	2.87
LIN318	1112.23	1098.58	1109.98	1098.90	2.46	1.98
Average:	533.65	523.68	529.77	519.25	0.76	0.64

TABLE IV
PERFORMANCE COMPARISON OF ACS, MMAS, GA, HGA, LIN-KER, BCO AND BCO+2-OPT

Instance	EIL51	KROA100	LIN318
ACS [17]	0.48/0.00	0.65/0.00	n/a
MMAS [18]	0.16/0.00	0.10/0.00	n/a
GA [14,15]	0.00/0.00	0.00/0.00	0.00/0.00
HGA [16]	0.59/0.47	0.01/0.01	1.37/0.73
Lin-Ker [11,28]	0.00/0.00	0.00/0.00	0.08/0.00
BS+2-opt [19]	1.14/0.53	1.36/0.73	n/a
BS+Modified 3-opt [19]	0.00/0.00	0.00/0.00	n/a
BCO+2-opt	0.00/0.00	0.00/0.00	0.09/0.00

x/*y*: *x* and *y* represent percentage difference of average tour length and best tour length from optimal value respectively. "n/a" stands for "not available".

TABLE V
PERFORMANCE COMPARISON OF BCO+2-OPT AND BS

Problem Instances	BCO+2-opt (%)		BS+2-opt [19] (%)		BS+Modified 3-opt [19] (%)	
	Avg.	Best	Avg.	Best	Avg.	Best
EIL51	0.00	0.00	1.14	0.53	0.00	0.00
BERLIN52	0.00	0.00	1.19	0.00	0.00	0.00
ST70	0.00	0.00	1.06	0.22	0.00	0.00
PR76	0.00	0.00	1.19	0.58	0.00	0.00
KROA100	0.00	0.00	1.36	0.73	0.00	0.00
EIL101	0.00	0.00	3.97	0.35	0.45	0.00
TSP225	0.00	0.00	6.60	5.35	1.31	1.06
A280	0.00	0.00	7.66	5.95	1.76	0.83

VII. Conclusion

A BCO algorithm based on the bees' collective foraging behaviour has been introduced to solve TSP. The integration of the 2-opt heuristic in our BCO algorithm has improved its performance significantly. The algorithm has been tested on a set of benchmark problems and is able to solve these problems optimally except for problems with larger dimension. We will explore the further use of higher dimension *k*-opt heuristics (e.g. 3-opt) to improve the BCO algorithm to obtain optimal solution for problems with larger dimension.

A linear function in calculating the waggle dance duration is employed in the current BCO and BCO+2-opt model. We will carry out investigations to see if it can be replaced by other type of functions such as sigmoid or quadratic functions. This is because waggle dance plays a significant role in directing other bees to achieve optimality. By sustaining and exploiting a relatively good solution with a reasonable duration, it helps in achieving optimality for these benchmark problems.

Acknowledgment

The authors would like to thank Science University of Malaysia and Ministry of Higher Education of Malaysia for the scholarship awarded to Li-Pei Wong to pursue his Ph.D. in Nanyang Technological University, Singapore.

References

[1] K. von Frisch, "Decoding the language of the bee," *Science*, vol. 185, no. 4152, pp. 663-668, 1974.

[2] M. Dorigo and T. Stützle, "The ant colony optimization metaheuristic: algorithms, applications, and advances." in *Handbook of Metaheuristics*, vol. 57, New York: Springer, 2002. pp. 250-285.

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

[4] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231-247, 1992.

[5] R. E. Burkard, V. G. Deineko, R. Van Dal, J. A. A. Van der Veen, and G. J. Woeginger, "Well-solvable special cases of the traveling salesman problem: A survey ," *SIAM Review*, vol. 40, no. 3, pp. 496-546, 1998.

[6] G. Gutin, "Traveling salesman problems." in *Handbook of Graph Theory* , J. L. Gross and J. Yellen, Ed. New York: CRC Press, 2007. pp. 279-299.

[7] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, "An analysis of several heuristics for the traveling salesman problem," *SIAM Journal on Computing*, vol. 6 , no. 3, pp. 563-581, 1977.

[8] A. M. Frieze, "An extension of Christofides heuristic to the k-person travelling salesman problem," *Discrete Applied Mathematics*, vol. 6, no. 1, pp. 79-83, 1983.

[9] B. Chandra, H. Karloff, and C. Tovey, "New results on the old k-opt algorithm for the traveling salesman problem," *SIAM Journal on Computing*, vol. 28, no. 6, pp. 1998-2029, 1999.

[10] S. Lin and B. W. Kerninghan, "An effective heuristic algorithm for the traveling salesman problem," *Operations Research*, vol. 21, no. 2, pp. 498-516, 1973.

[11] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," *European Journal of Operational Research*, vol. 126, no. 1, pp. 106-130, 2000.

[12] E. H. L. Aarts, J. H. M. Korst, and P. J. M. Vanlaarhoven, "A quantitative analysis of the simulated annealing algorithm - A case study for the traveling salesman problem," *Journal of Statistical Physics*, vol. 50, no. 1-2, pp. 187-206, 1988.

[13] J. Knox, "Tabu search performance on the symmetric traveling salesman problem," *Computers & Operations Research*, vol. 21, no. 8, pp. 867-876, 1994.

[14] B. Freisleben and P. Merz, "A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems," in *Proceedings of International Conference on Evolutionary Computation*, 1996. pp. 616-621.

[15] P. Merz and B. Freisleben, "Genetic local search for the TSP: New results," in *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, 1997. pp. 159-164.

[16] C. M. White and G. G. Yen, "A hybrid evolutionary algorithm for traveling salesman problem," in *Proceedings of Congress on Evolutionary Computation, 2004, CEC2004.*, 2004. pp. 1473 -1478.

[17] L. M. Gambardella and M. Dorigo, "Solving symmetric and asymmetric TSPs by ant colonies," in *Proceedings of IEEE International Conference on Evolutionary Computation, 1996.* 1996. pp. 622-627.

[18] T. Stützle and H. Hoos, "MAX-MIN ant system and local search for the traveling salesman problem," in *Proceedings of ICEC'97 - 1997 IEEE 4th International Conference on Evolutionary Computation*, 1997. pp. 308-313.

[19] P. Lucic and D. Teodorovic, "Computing with Bees: Attacking Complex Transportation Engineering Problems," *International Journal on Artificial Intelligence Tools*, vol. 12, no. 3, pp. 375-394, 2003.

[20] F. C. Dyer, "The biology of the dance language," *Annual Review of Entomology*, vol. 47, pp. 917-949, 2002.

[21] J. C. Biesmeijer and T. D. Seeley, "The use of waggle dance information by honey bees throughout their foraging careers," *Behavioral Ecology and Sociobiology*, vol. 59, no. 1, pp. 133-142, 2005.

[22] S. Nakrani and C. Tovey, "On honey bees and dynamic server allocation in Internet hosting centers," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 223-240, 2004.

[23] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459-471, 2007.

[24] J. van Rijswijck, "Are bees better than fruitflies? Experiments with a hex playing program." In *Advances in Artificial Intelligence*, vol. 1822/2000, Berlin / Heidelberg: Springer, 2007. pp. 13-25.

[25] C. S. Chong, M. Y. H. Low, A. I. Sivakumar, and K. L. Gay, "A bee colony optimization algorithm to job shop scheduling," in *Proceedings of the 2006 Winter Simulation Conference*, 2006. pp. 1954-1961.

[26] C. S. Chong, M. Y. H. Low, A. I. Sivakumar, and K. L. Gay, "Using a bee colony Algorithm for neighborhood search in job shop scheduling problems," in *Proceedings of 21st European Conference on Modeling and Simulation (ECMS 2007)*, 2007.

[27] S. C. Greenway, "Hormones in human metabolism and disease." in *Functional Metabolism: Regulation and Adaptation*, K. B. Storey, Ed. John Wiley & Sons, 2005. pp. 271-294.

[28] K. Helsgaun, "An effective implementation of the Lin-Kernighan traveling salesman heuristic," Roskilde University, 81, 1999.