

An Efficient Bee Colony Optimization Algorithm for Traveling Salesman Problem using Frequency-based Pruning

Li-Pei Wong[†]

Malcolm Yoke Hean Low[‡]

School of Computer Engineering, Nanyang Technological University
Nanyang Avenue, Singapore 639798.

Email: [†]wonglpei@pmail.ntu.edu.sg, [‡]yhlow@ntu.edu.sg

Chin Soon Chong

Singapore Institute of Manufacturing Technology
71 Nanyang Drive, Singapore 638075.

Email: cschong@simtech.a-star.edu.sg

Abstract—In a bee colony, bees perform waggle dance in order to communicate the information of food source to their hive mates. This foraging behaviour has been adapted in a Bee Colony Optimization (BCO) algorithm together with 2-opt local search to solve the Traveling Salesman Problem (TSP) [1]. To reduce the high overhead incurred by 2-opt in the BCO algorithm proposed previously, two mechanisms named frequency-based pruning strategy (FBPS) and fixed-radius near neighbour (FRNN) 2-opt are presented. FBPS suggests that only a subset of promising solutions are allowed to perform 2-opt based on the accumulated frequency of its building blocks recorded in a matrix. FRNN 2-opt is an efficient implementation of 2-opt which exploits the geometric structure in a permutation of TSP sequence. Both mechanisms are tested on a set of TSP benchmark problems and the results show that they are able to achieve a 58.42% improvement while maintaining the solution quality at 0.02% from known optimal.

I. INTRODUCTION

Bees are highly organized social insects. The survival of the entire colony depends on every individual bee. Bees use a systematic task segregation among them to ensure a continued existence of its colony. They perform various tasks such as foraging, reproduction, taking care of young, patrolling, housekeeping and constructing hive. Of these, foraging is a major activity as bees have to ensure an undisrupted supply of food source to the colony.

The foraging behaviour of bees remains mysterious for many years until von Frisch translated the language embedded in bee waggle dances [2]. 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. Via this informative dance, the bee has actually informed its hive mates about the direction and distance to the new food discovery. This will eventually attract more bees towards the new food source. Discussions about waggle dance can be found in [3], [4].

The foraging behaviour of bees has been adapted as a useful computational algorithm to solve complex problems in different domains. Among them are dynamic server allocation for Internet hosting center [5], numerical function optimization [6], [7], telecommunication network routing [8], stochastic

vehicle routing problem [9], [10] and Job Shop Scheduling Problem [11], [12].

Another problem that has been attempted is the Traveling Salesman Problem (TSP) [1], [13]. TSP is a problem that requires a salesman to visit a set of fully connected cities where each connection between two cities is associated with a cost. The traveling sequence has to comply with a constraint, that is the salesman will start at a city, visit each city exactly once, and back to the start city. The resulting route should incur a minimum cost. TSP is common in areas such as logistics, transportation and semiconductor industries. Finding an optimized scan chains route in integrated chips testing, parcels collection and delivery 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 and it remains as an important testbed for many newly developed algorithms.

This paper depicts an efficient Bee Colony Optimization (BCO) algorithm that significantly improves the algorithm proposed previously by the authors [1] in terms of computational speed. In [1], a BCO algorithm with 2-opt was tested on a set of TSP benchmark instances. Each solution generated by bees was locally optimized by an exhaustive 2-opt. The 2-opt heuristic is implemented according to the basic idea which eliminates two arcs in order to obtain two different paths. These two paths are then reconnected in the other possible way if the new path results in a shorter tour length. Although the integration of 2-opt gave promising results, the results presented showed that the execution time could be improved further. To achieve this, two mechanisms, namely frequency-based pruning strategy (FBPS) and fixed-radius near neighbour (FRNN) 2-opt, are presented in this paper. FBPS is a strategy that allows only a subset of promising solutions to perform 2-opt and FRNN 2-opt is an efficient implementation of 2-opt.

This paper starts with a discussion on the BCO algorithm (Section II). It is followed by a discussion on FBPS (Section III) which describes the building block concept and its working

mechanism. Section IV describes the FRNN 2-opt adapted from [14]. Section V describes the implementation details, experiment platform and source of the benchmark problems utilized in this study. It is then followed by Section VI which presents the findings of this paper. Finally, this paper ends with a conclusion.

II. BCO FOR TSP

This section explains the BCO algorithm for TSP which is inspired by the foraging behaviour of bees. An overview of the proposed BCO algorithm integrated with FBPS and FRNN 2-opt is presented. Details on path construction and waggle dance by bees will also be presented.

A. Overview of BCO with 2-opt

The outline of the BCO algorithm is shown in Algorithm 1. In the algorithm, a group of bees is created during the initial stage. The number of bees, N_{Bee} , is equal to the total number of cities in the TSP problem instance. During the first iteration, as no dance is observed, bees use state transition heuristic and Nearest Neighbourhood heuristic to generate initial solutions. These two approaches share equal likelihood to be chosen by a bee. State transition heuristic is implemented such that a bee starts from a random city, and chooses the next city to visit by using the state transition rule discussed in Section II-B. Under the Nearest Neighbourhood heuristic, 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.

Algorithm 1 BCO with 2-opt local search for TSP.

```

procedure BCO
  global best tour length,  $TL_{Best} \leftarrow \infty$ 
  Initialize_Population()
  while stop criteria are not fulfilled do
    for each forager bee  $f_i$  do
      tour  $T \leftarrow \{\}$ 
       $f_i$ .Observe_and_Select_Dance()
       $T \leftarrow f_i$ .Forage_ByTransRule()
      if  $T$  is not pruned by FBPS then
         $T \leftarrow f_i$ .Perform_FRNN_2-Opt()
      end if
      if  $T$ .length <  $f_i$ .best_tour_length then
         $f_i$ .Perform_Waggle_Dance()
      end if
      if  $T$ .length <  $TL_{Best}$  then
         $TL_{Best} \leftarrow T$ .length
      end if
    end for
  end while
end procedure BCO

```

For subsequent iterations, foraging process is initiated. A bee will decide if it needs to follow a dance before leaving the hive. After it completes a tour, the frequency-based pruning strategy (FBPS, refer Section III) will check if it needs a transformation by the fixed-radius near neighbour 2-opt (FRNN 2-opt, refer Section IV). The bee will start performing waggle dance if the resulting tour is shorter than its own personal best tour length. These steps will be repeated for

a certain number of iterations until the stopping criteria are fulfilled. The differences between the proposed algorithm as given in Algorithm 1 and the one in [1] will be explained in Sections III-B and IV.

B. Path Construction by Artificial Bees

The foraging behaviour of bees in BCO works as follows. During the foraging process, a bee is required to travel from a city to another city until it makes a full round trip based on a transition rule. This rule defines the transition probability to move from city i to city j after n transitions, $P_{ij, n}$, as given in Eq. 1. When the path construction ends, a permutation of cities is obtained and becomes one of the solutions for the TSP being solved. d_{ij} represents the distance between two cities and $\rho_{ij, n}$ denotes the arc fitness on the connecting edge of that two cities. $A_{i, n}$ is a set of cities (not yet visited) that can be reached from i at transition n . α and β are parameters that determine the relative significance of arc fitness versus heuristic distance.

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

d_{ij} is designed such that it has an inversely proportional relation with $P_{ij, n}$. The shorter the distance between i and j , the higher is the probability of j to be chosen as the next city to be visited.

The arc fitness, $\rho_{ij, n}$, measures the probability according to the existence of a two-city building block (between city i and city j) in a preferred path. The concept of building block will be further discussed in Section III-A. In natural metaphor, when a bee discovers a new food source, it will perform waggle dance. If another bee gets attracted to the dance, it tends to forage in that area. In BCO, this is modeled via the implementation of a preferred path, denoted as θ . θ is a permutation of moves a bee observed from another hive mate. It serves as a guidance in the foraging process. $\rho_{ij, n}$ is defined in Eq. 2, $\forall j \in A_{i, n}$, $0 \leq \lambda \leq 1$:

$$\rho_{ij, n} = \begin{cases} \lambda & , j \in F_{i, n}, |A_{i, n}| > 1 \\ \frac{1 - \lambda |A_{i, n} \cap F_{i, n}|}{|A_{i, n} - F_{i, n}|} & , j \notin F_{i, n}, |A_{i, n}| > 1 \\ 1 & , |A_{i, n}| = 1 \end{cases} \quad (2)$$

λ represents the probability of following a city in θ . $F_{i, n}$ is a set that contains one city which the bee prefers to move from 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, $F_{H, 0} = \{\theta(1)\}$. If the current visiting city i is at the m -th position in the preferred path after n transitions, then $F_{\theta(m), n} = \{\theta(m+1)\}$. $F_{i, n}$ contains one element as only $\theta(m+1)$ (forward adjacent city of i in θ) is considered. $F_{i, n}$ can have two cities if $\theta(m-1)$ (backward adjacent city) is also considered.

In Eq. 2, the first two rules ensure that the arc recommended by the θ (if there is one) is assigned with a probability λ whereas the rest of the arcs are assigned with the same

probability. If there are l arcs to consider, one arc (exist in θ) will be set to λ and the rest of the arcs will be set to $(1 - \lambda)/(l - 1)$. If none of its arcs is found to be alike as compared to the θ , all the arcs being considered (in $A_{i,n}$) will be set to $1/l$. The third rule will assign 1 to $\rho_{ij, n}$ when there is only one city left in $A_{i,n}$. This is observed at the last transition before a bee re-visits the start city to complete the tour.

C. Waggle Dance

When a bee finds a new food source, upon returning to its hive, it will perform a waggle dance. In the BCO algorithm, not all bees that completed a tour will dance. Only bees that produce shorter tour length compared to its own previous best tour length are allowed to dance. Thus, bees in the BCO algorithm are equipped with memory to remember their “personal best tour length” obtained during the algorithm execution.

If a bee dances, the waggle dance will last for a certain duration. The dance duration D_i of bee i is determined by a linear function as given in Eq. 3. The duration is measured in terms of iterations in algorithm execution. According to the linear function, if bee i has a higher Pf_i , it will be given a chance to dance longer (dance appears in more iterations). Otherwise, it dances for a shorter period. Pf_i denotes the profitability score of a bee i as defined in Eq. 4. Pf_{colony} denotes the bee colony’s average profitability as in Eq. 5 and is updated after each bee completes its tour. K is a user defined scaling factor that controls the magnitude of the duration.

$$D_i = K \cdot \frac{Pf_i}{Pf_{colony}} \quad (3)$$

$$Pf_i = \frac{1}{L_i}, \quad L_i = \text{tour length} \quad (4)$$

$$Pf_{colony} = \frac{1}{N_{Bee}} \sum_{i=1}^{N_{Bee}} Pf_i \quad (5)$$

Pf_i can be interpreted as the quantity of the nectar collected by bee i . Higher quantity of nectar will be collected if a bee travel along a shorter route. Thus, Pf_i is defined to be inversely proportional to the tour length.

The waggle dances performed by bees are accumulated and made available for observation by other hive mates that are about to go out to forage. Although one might think of when a bee contributes a dance, it does not forage and hence the number of foraging bees is reduced. This is not the case as the BCO algorithm uses the same number of bees to forage in each iteration. When a dance duration expires (as determined by Eq. 3), it will be discarded. After many iterations, this might create a situation where no bee is performing waggle dance, as all bees fail to get a better tour length and all accumulated dances have expired. This situation should be avoided as waggle dance is an important communication tool among bees that is able to guide bees in their foraging process. A memory adjustment policy is used in this case. When no bee has danced for several consecutive cycles (e.g. ten consecutive iterations), the personal best tour length of each bee in the population will

be adjusted. It will be increased by a certain percentage as defined by user, so that bees have a higher chance to get a shorter tour and hence contribute a dance.

Before a bee leaves its hive, it will decide if it needs to follow a dance shown by its hive mates with a probability of P_{follow} . P_{follow} is adjusted dynamically according to the profitability score of a bee and the colony based on the lookup table in Table I, which is adapted from the [5], [12]. Essentially, a bee is more likely to follow a waggle dance if its Pf_i is low compared to Pf_{colony} . In the extreme case where P_{follow} is zero, a bee will not observe any dance shown by its mates and keep to its own path which it explored previously. When a bee has decided to follow a dance, it will choose a dance from a pool of dances with equal likelihood.

TABLE I
LOOKUP TABLE FOR P_{follow} ADJUSTMENT.

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

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 Eq. 1.

The use of the memory adjustment policy and the lookup table is to avoid local optima. These two mechanisms encourage exploration and hence bees are able to search for feasible solutions in the solution space more efficiently.

III. FREQUENCY-BASED PRUNING STRATEGY

This section describes the FBPS, which is introduced to permit only a subset of promising results to undergo 2-opt. The concept of building block is discussed first followed by the FBPS working mechanism.

A. The Building Block Concept

Previous research into building blocks have been found in many disciplines. One of them is the Genetic Algorithms (GA) in computer science. GA uses binary encoding scheme as its chromosomes. However, other schemes such as gray code, fixed or floating numbers are still in practice. All these schemes encode a chromosome as a block of numbers. These chromosomes are then further transformed by genetic operators such as crossover and mutation, according to the concept of evolution: survival of the fittest.

Gero et al. conducted to study on the building blocks in GA chromosomes [15], [16]. A population of chromosomes is segregated into two groups, one above a threshold (better fitness), and the other one below the threshold (lower fitness). They proposed a mechanism to identify prevalent building blocks in the better fitness group and absent building blocks

in the lower fitness group. The identified building blocks are then combined as a single gene and they are prevented from further modifications by crossover and mutation. The proposed approach was tested on space layout planning and results showed that if these blocks are frequently reused in the evolutionary process, a significant reduction of computational time can be achieved.

Wu and Lindsay [17] investigated the effects of location-independent building blocks in GA. Urquhart et al. [18] proposed a building block mutation for GA to solve the street based routing problem. Both techniques also ensure a frequent reusability of prevalent building blocks in the evolutionary process of GA.

Gao et al. proposed an Ant Colony Optimization algorithm with Association Rule discovery for TSP [19]. This method is able to identify a set of important sequential patterns (building blocks) found in the permutations of TSP solutions with certain minimum support. These identified blocks will have a higher odds to be utilized in its optimization operations. A 30-city TSP with known optimal was tested and the best result showed a deviation of 0.95% from the optimal.

The FBPS proposed in this paper is able to identify some common building blocks in the TSP solutions generated by bees. The identified building blocks will be used to decide which solutions need to be further improved by 2-opt. The mechanism of the strategy is presented in the next section.

B. Mechanism of FBPS

In [1], each and every solution generated by bees will be locally optimized by 2-opt. This leads to high computational time for obtaining good results. Hence, a pruning strategy based on the accumulated frequency of building blocks is proposed to prohibit 2-opt operations from being performed on some solutions.

Consider the following example that explains the working mechanism of the pruning strategy. Firstly, an $n \times n$ matrix is created with each entry of the matrix assigned to zero. n is the number of cities of a TSP instance. If a 6-city TSP is being solved, a matrix H with size of 6×6 , is created:

$$H = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 0 & 0 & 0 & 0 & 0 \\ B & 0 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 0 & 0 & 0 & 0 \\ E & 0 & 0 & 0 & 0 & 0 & 0 \\ F & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

The purpose of creating H is to record the accumulated frequency of the smallest building blocks of every solutions generated by bees. In this case, the smallest building block contains two elements. If a solution “A, B, C, D, E, F, A” is considered, H will be updated as follow:

$$H = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 1 & 0 & 0 & 0 & 1 \\ B & 1 & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 1 & 0 & 1 & 0 & 0 \\ D & 0 & 0 & 1 & 0 & 1 & 0 \\ E & 0 & 0 & 0 & 1 & 0 & 1 \\ F & 1 & 0 & 0 & 0 & 1 & 0 \end{array}$$

The strategy considers only the smallest building blocks in a solution, but it can be extended to consider building blocks in different sizes e.g. three-element block and so on. For every building block found in the permutation, two entries of H will be updated. For instance, when the building block AB is encountered, the entries of AB and BA of H are both incremented by 1. If another solution with a permutation of “A, B, E, D, C, F, A” is added, H becomes:

$$H = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 2 & 0 & 0 & 0 & 2 \\ B & 2 & 0 & 1 & 0 & 1 & 0 \\ C & 0 & 1 & 0 & 2 & 0 & 1 \\ D & 0 & 0 & 2 & 0 & 2 & 0 \\ E & 0 & 1 & 0 & 2 & 0 & 1 \\ F & 2 & 0 & 1 & 0 & 1 & 0 \end{array}$$

After several cycles of update, H might end up with the following entries:

$$H = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0 & 88 & 929 & 22 & 113 & 23 \\ B & 88 & 0 & 754 & 355 & 105 & 4 \\ C & 929 & 754 & 0 & 11 & 826 & 2 \\ D & 22 & 355 & 11 & 0 & 176 & 933 \\ E & 113 & 105 & 826 & 176 & 0 & 56 \\ F & 23 & 4 & 2 & 933 & 56 & 0 \end{array}$$

By examining H , building blocks with high frequency of access can be identified. Taking the first row of H as an example, where it records the frequency of the building blocks with A as the first element {AB, AC, AD, AE, AF}. It has a total frequency of 1175 (summation of 88, 929, 22, 113 and 23). The building block of AA is disregarded as the system does not allow any recursive visit. Of these five building blocks, AB contributes 7.49% of the total occurrences ($\frac{88}{1175} \times 100\%$). AC, AD, AE and AF contribute 79.06%, 1.87%, 9.62% and 1.96% respectively. Similar calculation is carried out on the rest of entries in H , and the following matrix is obtained:

$$H_{\%} = \begin{array}{c|cccccc} & A & B & C & D & E & F \\ \hline A & 0.00 & 7.49 & 79.06 & 1.87 & 9.62 & 1.96 \\ B & 6.74 & 0.00 & 57.73 & 27.18 & 8.04 & 0.31 \\ C & 36.84 & 29.90 & 0.00 & 0.44 & 32.75 & 0.08 \\ D & 1.47 & 23.71 & 0.73 & 0.00 & 11.76 & 62.32 \\ E & 8.86 & 8.23 & 64.73 & 13.79 & 0.00 & 4.39 \\ F & 2.26 & 0.39 & 0.20 & 91.65 & 5.50 & 0.00 \end{array}$$

Although a two-entry update system is described in this paper, the accumulated frequency can also be updated to only the entries in the upper diagonal of the matrix. This is because the TSP instances being solved are symmetrical TSPs. Based on $H_{\%}$, building blocks that are $q\%$ and above are identified as hot spots. q is a user defined threshold value. For instance, if $q = 5\%$, AB, AC and AE are the hot spots with A as its first element (as shown in Fig. 1). The identified hot spots will then be employed in the decision making process. The pruning criterion is set as follow: if any solution contains $\kappa\%$ or more dissimilar building blocks when compared to Fig. 1, it will be prohibited from performing the FRNN 2-opt. Otherwise, it

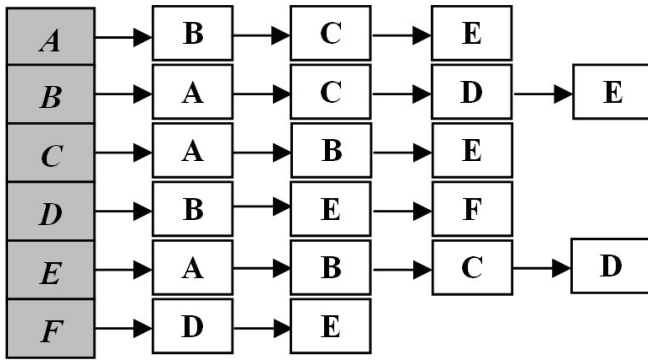


Fig. 1. An adjacency list that shows the building blocks that are greater than 5% of the total row-based frequency. These blocks are marked as hot spots.

will be further optimized by the FRNN 2-opt. If a permutation of “A, D, C, F, B, E, A” is compared against Fig. 1, building blocks of AD, DC, CF and FB are not hot spots. Only BE and EA are hot spots. If κ is configured at 20%, this particular permutation will be pruned from performing 2-opt. If another permutation of “A, B, C, E, D, F, A” is encountered, it will be accepted for 2-opt optimization as only one dissimilar building blocks is observed ($16.67\% < \kappa$ where κ is set at 20%). κ is a user defined pruning threshold and experiments of tuning it will be described in Section VI-A.

IV. FIXED-RADIUS NEAR NEIGHBOUR 2-OPT

In [1], the 2-opt local search applied an exhaustive comparison and swap method in finding two potential edges for its operation and hence it is a computationally expensive process. In this paper, an efficient way of implementing 2-opt by Bentley [14] is adapted, namely Fixed-radius Near Neighbour (FRNN) 2-opt. The FRNN 2-opt method is supported by an observation found in any successful swap in 2-opt, that is, for any successful swap, at least one node has an edge that will decrease the tour length. Otherwise, the 2-opt swap could not decrease the tour length as both edges increase in length. Because of this observation, the exploration of the second edge can be restricted to a circle centered at a node of the first edge with a radius equal to the edge length.

Consider a tour which is drawn on a two dimensional surface as in Fig. 2. The tour with a permutation of “E, F, I, H, C, G, A, B, D, E” will be optimized by 2-opt. While the edge that links E and F, E_{EF} , serves as the first edge, a search of the second edge based on FRNN method centered at E is performed with a radius of d_{EF} . All the nodes within the vicinity will be considered to form the second edge together with its single appropriate neighbour. In Fig. 2, node B is found within the vicinity. E_{BA} and E_{BD} will then be picked as potential candidates for the second edge in 2-opt swap.

If the exhaustive 2-opt local search is applied on the same permutation of “E, F, I, H, C, G, A, B, D, E” and E_{EF} is chosen as the first edge, the potential candidates for second edge are E_{IH} , E_{HC} , E_{CG} , E_{GA} , E_{AB} and E_{BD} . In the worst case scenario, the exhaustive 2-opt has to perform six

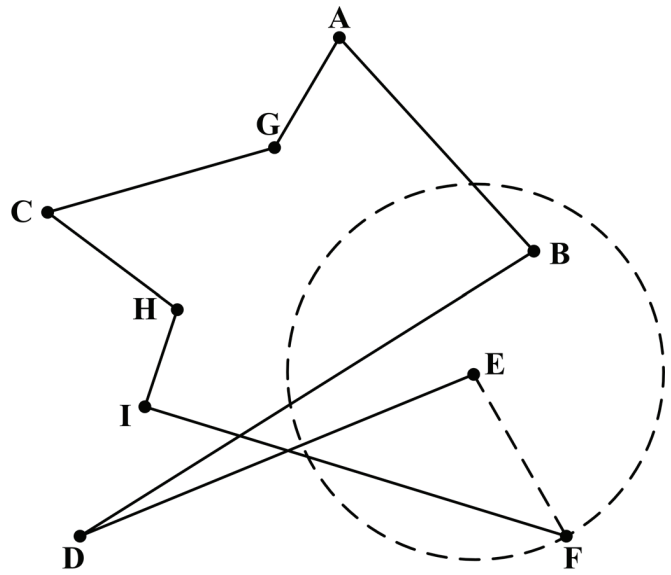


Fig. 2. Searching for the second edge based on FRNN method where E_{EF} is the first edge.

comparisons before a successful swap. However, only two comparisons are needed in FRNN 2-opt. It is equivalent to a 67% reduction in terms of the number of comparisons.

V. IMPLEMENTATION DETAILS

The BCO algorithm described in this paper is developed using JAVA with NetBeans IDE 5.5 as the development tool. Experiments were carried out 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).

The algorithm described in this paper is tested on a set of benchmark problems taken from TSPLIB¹. 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 instances taken from TSPLIB follow a naming notation that starts with alphabetical characters (denote the origin of the problem) and followed by a numerical figure (denote the dimension of the problem). For example, ATT48 is a 48-city problem from the ATT series; LIN318 is a 318-city problem from the LIN series and so forth.

VI. RESULTS AND DISCUSSIONS

This section presents some experimental results in this study. The results for each benchmark instance are computed as a average tour length of five replications. Before discussing the results, a notation of δ is introduced. δ denotes the tour length percentage deviation from the known optimum for a particular problem instance.

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

TABLE II
RESULTS FOR EXPERIMENTS OF ADJUSTING κ .

Exp.	κ (%)	Attr.	Rep. 1	Rep. 2	Rep. 3	Rep. 4	Rep. 5	Avg.	δ_{Avg}	σ
I	5	Length	42029	42163	42157	42148	42155	42130.40	0.24	56.94
		Time (s)	10856	43690	43624	42969	43606	36949.00	-	-
		Cycle_best	1241	944	1233	1263	669	1069.00	-	-
		#2-opt	394638	1590000	1590000	1590000	1590000	-	-	-
		%pruned	61.11	66.70	70.18	88.12	67.12	-	-	-
II	10	Length	42029	42029	42029	42029	42091	42041.40	0.03	27.73
		Time (s)	11305	19525	14396	16595	39167	20197.60	-	-
		Cycle_best	1231	2495	1854	2118	895	1714.00	-	-
		#2-opt	391458	793410	583848	673524	1590000	-	-	-
		%pruned	19.07	19.71	19.62	19.66	20.71	-	-	-
III	30	Length	42029	42091	42029	42029	42143	42064.20	0.08	51.59
		Time (s)	26045	47232	30277	21112	47424	34418.00	-	-
		Cycle_best	2752	2671	3204	3229	2566	2883.40	-	-
		#2-opt	875136	1590000	1018872	1026822	1590000	-	-	-
		%pruned	0.00	0.00	0.00	0.00	0.00	-	-	-
IV	50	Length	42155	42107	42091	42091	42155	42119.80	0.22	32.79
		Time (s)	47244	47400	47456	47304	47351	47351.00	-	-
		Cycle_best	1938	1200	2720	3010	3551	2482.80	-	-
		#2-opt	1590000	1590000	1590000	1590000	1590000	-	-	-
		%pruned	0.00	0.00	0.00	0.00	0.00	-	-	-
V	75	Length	42162	42102	42029	42082	42091	42093.20	0.15	47.60
		Time (s)	47275	47185	26347	46994	46957	42951.60	-	-
		Cycle_best	2714	1520	2780	2724	2995	2545.60	-	-
		#2-opt	1590000	1590000	884040	1590000	1590000	-	-	-
		%pruned	0.00	0.00	0.00	0.00	0.00	-	-	-
VI	90	Length	42155	42083	42155	42163	42029	42117.00	0.21	58.96
		Time (s)	46769	46796	46812	46792	33269	44087.60	-	-
		Cycle_best	2246	2638	3026	3607	3549	3012.20	-	-
		#-opt	1590000	1590000	1590000	1590000	1128582	-	-	-
		%pruned	0.00	0.00	0.00	0.00	0.00	-	-	-

A. κ Adjustment

Firstly, results for the experiments of tuning κ are presented. Six experiments were carried out with the settings as follows: $\alpha = 1$, $\beta = 10$, $\lambda = 0.95$, $K = 100$, $N_{Bee} = 318$ and $BC_{Max} = 5000$. FBPS and FRNN 2-opt is integrated and each experiment consists of five replications. LIN318 (42029 as its known optimal) is used as the test dataset for this set of experiments. The results of the experiments are summarized in Table II. Whenever a replication achieves the optimum tour length, the execution will be halted and the results will be collected. Otherwise, it will be executed till termination for 5000 iterations (cycles).

For each replication, the table shows the best tour length, execution time and at which cycle the best tour length is achieved. #2-opt denotes the total number of 2-opt operations if the FRNN 2-opt is performed on every solution. For cases where the optimum tour length is obtained before 5000 iterations, #2-opt is equal to $N_{Bee} \times \text{Cycle_best}$. For instance, #2-opt for replication 1 in experiment I equals to $318 \times 1241 = 394638$. For cases where the execution is run for 5000 iterations till termination, #2-opt is equal to $N_{Bee} \times 5000 = 1590000$. #pruned and %pruned denotes the total number of pruned solutions and its percentage.

Based on the results shown in Table II, experiment II produces the most satisfactory results. It shows the lowest values for the average tour length and the average computational time, that are 42041.40 ($\delta_{Avg} = 0.03$) and 20197.60s respectively.

The lowest standard deviation for tour length, $\sigma = 27.73$, is achieved via the same experiment too.

In terms of the total number of pruned solutions, Table II shows that experiments III, IV, V and VI fail to prune any solution. This makes their computational times much longer than experiment II (nearly twice the time of experiment II), although their average tour lengths are not very much different from experiment II. In experiment I, by setting κ to a low value, it is expected that the execution time will be reduced. However, the results show otherwise. The FBPS prunes all solutions towards the end of the algorithm execution and this does not contribute to finding the optimal solutions. Hence, more time is needed to search in the space. This makes the performance of experiment I not comparable to the rest. All these observations show that κ should not be configured either too high or too low. To achieve a reasonable trade off between solution quality and computational cost, κ should be set within a bound of [10%, 30%].

B. Experimental Results

The experimental results that show the performance of the BCO algorithm integrated with FBPS and FRNN 2-opt in terms of solution quality and computational timing on a set of 20 selected problem instances from TSPLIB are presented in Tables III and IV. The tables illustrate results of three different sets of experiment. Parameter settings for these three experiments are the same as what were used in Section VI-A except $BC_{Max} = 10000$. Experiment A uses the 2-opt which

TABLE III
PERFORMANCE OF THE BCO ALGORITHM IN TERMS OF SOLUTION QUALITY ON 20 TSPLIB BENCHMARK PROBLEMS.

Inst.	Opt.	Exp. A BCO+2-opt						Exp. B BCO+FRNN 2-opt						Exp. C BCO+FRNN 2-opt+FBPS					
		Tour Length			δ (%)			Tour Length			δ (%)			Tour Length			δ (%)		
		Best	Avg.	σ	Best	Avg.	σ	Best	Avg.	σ	Best	Avg.	σ	Best	Avg.	σ	Best	Avg.	σ
ATT48	10628	10628	10628.00	0.00	0.00	0.00	10628	10628.00	0.00	0.00	0.00	10628	10628.00	0.00	0.00	0.00	0.00	0.00	0.00
EIL51	426	426	426.00	0.00	0.00	0.00	426	426.00	0.00	0.00	0.00	426	426.00	0.00	0.00	0.00	0.00	0.00	0.00
BERLIN52	7542	7542	7542.00	0.00	0.00	0.00	7542	7542.00	0.00	0.00	0.00	7542	7542.00	0.00	0.00	0.00	0.00	0.00	0.00
ST70	675	675	675.00	0.00	0.00	0.00	675	675.00	0.00	0.00	0.00	675	675.00	0.00	0.00	0.00	0.00	0.00	0.00
EIL76	538	538	538.00	0.00	0.00	0.00	538	538.00	0.00	0.00	0.00	538	538.00	0.00	0.00	0.00	0.00	0.00	0.00
PR76	108159	108159	108159.00	0.00	0.00	0.00	108159	108159.00	0.00	0.00	0.00	108159	108159.00	0.00	0.00	0.00	0.00	0.00	0.00
KROA100	21282	21282	21282.00	0.00	0.00	0.00	21282	21282.00	0.00	0.00	0.00	21282	21282.00	0.00	0.00	0.00	0.00	0.00	0.00
KROB100	22141	22141	22141.00	0.00	0.00	0.00	22141	22141.00	0.00	0.00	0.00	22141	22141.00	0.00	0.00	0.00	0.00	0.00	0.00
KROC100	20749	20749	20749.00	0.00	0.00	0.00	20749	20749.00	0.00	0.00	0.00	20749	20749.00	0.00	0.00	0.00	0.00	0.00	0.00
KROD100	21294	21294	21294.00	0.00	0.00	0.00	21294	21294.00	0.00	0.00	0.00	21294	21294.00	0.00	0.00	0.00	0.00	0.00	0.00
KROE100	22068	22068	22068.00	0.00	0.00	0.00	22068	22068.00	0.00	0.00	0.00	22068	22068.00	0.00	0.00	0.00	0.00	0.00	0.00
EIL101	629	629	629.00	0.00	0.00	0.00	629	629.00	0.00	0.00	0.00	629	629.00	0.00	0.00	0.00	0.00	0.00	0.00
LIN105	14379	14379	14379.00	0.00	0.00	0.00	14379	14379.00	0.00	0.00	0.00	14379	14379.00	0.00	0.00	0.00	0.00	0.00	0.00
KROA150	26524	26524	26524.00	0.00	0.00	0.00	26524	26524.00	0.00	0.00	0.00	26524	26524.00	0.00	0.00	0.00	0.00	0.00	0.00
KROB150	26130	26130	26130.00	0.00	0.00	0.00	26130	26130.00	0.00	0.00	0.00	26130	26130.00	0.00	0.00	0.00	0.00	0.00	0.00
KROA200	29368	29368	29368.00	0.00	0.00	0.00	29368	29368.00	0.00	0.00	0.00	29368	29368.00	0.00	0.00	0.00	0.00	0.00	0.00
KROB200	29437	29437	29437.00	0.00	0.00	0.00	29437	29437.00	0.00	0.00	0.00	29437	29437.00	0.00	0.00	0.00	0.00	0.00	0.00
TSP225	3916	3916	3916.00	0.00	0.00	0.00	3916	3916.00	0.00	0.00	0.00	3916	3916.00	0.00	0.00	0.00	0.00	0.00	0.00
A280	2579	2579	2579.00	0.00	0.00	0.00	2579	2579.00	0.00	0.00	0.00	2579	2579.00	0.00	0.00	0.00	0.00	0.00	0.00
LIN318	42029	42029	42068.40	42.00	0.00	0.09	42029	42056.00	32.26	0.00	0.06	42029	42043.80	23.56	0.00	0.04	0.00	0.04	0.04
Average:					0.00	0.005					0.00	0.003						0.00	0.002

TABLE IV
COMPUTATIONAL TIMING OF THE BCO ALGORITHM AND ITS VARIANTS ON 20 TSPLIB BENCHMARK PROBLEMS.

Inst.	Exp. A BCO+2-opt			Exp. B BCO+FRNN 2-opt				Exp. C BCO+FRNN 2-opt+FBPS					
	Time (s)			Time (s)			PI. (%)	Time (s)			PI. (%)		
	Best	Avg. (x)	σ	Best	Avg. (y)	σ	$\frac{(y-x)}{x}$	Best	Avg. (z)	σ	$\frac{(z-y)}{y}$	$\frac{(z-x)}{x}$	
ATT48	1.00	2.20	2.17	1.00	1.00	0.00	-54.55	1.00	1.20	0.45	20.00	-45.45	
EIL51	1.00	6.20	3.56	1.00	1.60	1.34	-74.19	3.00	4.80	1.30	200.00	-22.58	
BERLIN52	0.03	0.08	0.08	0.03	0.08	0.11	0.00	0.03	0.08	0.11	0.00	0.00	
ST70	2.00	12.20	7.56	4.00	8.40	4.56	-31.15	7.00	11.80	5.45	40.48	-3.28	
EIL76	2.00	16.80	12.48	1.00	13.00	9.35	-22.62	5.00	8.00	2.55	-38.46	-52.38	
PR76	55.00	86.00	38.97	20.00	59.00	67.37	-31.40	44.00	88.60	53.54	50.17	3.02	
KROA100	8.00	19.00	7.45	1.00	3.20	3.83	-83.16	1.00	2.20	1.64	-31.25	-88.42	
KROB100	26.00	40.60	14.33	1.00	9.20	6.61	-77.34	3.00	9.80	6.42	6.52	-75.86	
KROC100	16.00	24.20	6.42	2.00	4.20	1.92	-82.64	2.00	5.40	3.97	28.57	-77.69	
KROD100	4.00	115.80	120.55	32.00	66.00	30.65	-43.01	17.00	74.60	77.17	13.03	-35.58	
KROE100	31.00	97.00	58.60	77.00	142.60	75.44	47.01	44.00	72.20	24.18	-49.37	-25.57	
EIL101	67.00	199.40	137.53	52.00	148.60	91.57	-25.48	49.00	72.00	34.01	-51.55	-63.89	
LIN105	2.00	7.00	7.42	1.00	1.20	0.45	-82.86	1.00	2.60	2.19	116.67	-62.86	
KROA150	502.00	878.80	447.66	273.00	925.20	1019.70	5.28	98.00	165.20	56.99	-82.14	-81.20	
KROB150	184.00	393.20	304.42	108.00	234.40	94.88	-40.39	2.00	60.40	46.23	-74.23	-84.64	
KROA200	1986.00	2477.60	468.63	949.00	1330.80	400.13	-46.29	88.00	180.40	90.00	-86.44	-92.72	
KROB200	14981.00	22026.80	6896.08	1796.00	2803.60	1221.99	-87.27	337.00	615.20	219.01	-78.06	-97.21	
TSP225	2987.00	4755.60	1083.17	3645.00	11328.60	10345.40	138.22	721.00	999.60	319.27	-91.18	-78.98	
A280	2974.00	3631.20	414.18	1519.00	1969.00	280.49	-45.78	248.00	373.20	83.02	-81.05	-89.72	
LIN318	144286.00	147973.40	2191.47	30951.00	71463.60	33253.71	-51.71	7162.00	9709.60	2613.72	-86.41	-93.44	
Average:							-34.47				-13.74	-58.42	

requires an exhaustive comparison and swap in its operation. Experiment B uses the FRNN 2-opt explained in Section IV but not the FBPS. Experiment C uses both FRNN 2-opt and FBPS and the κ is set at 10%.

The results for experiment A in Table III has been previously reported in [1] and they were comparable with other seven existing approaches for solving TSP. As the solution quality is maintained after the FRNN 2-opt and FBPS integration, such comparison study as in [1] is not included in

this paper. The focus of this paper is to investigate the effect on computational timing when such the integration is made. Readers may refer to [1] in order to obtain further details on the comparison against the seven approaches.

As shown in Table III, in terms of δ_{Avg} , the three approaches are able to obtain the optimal tour length for all the problems except LIN318. δ_{Avg} for LIN318 in experiments A, B and C are 0.09%, 0.06% and 0.04% respectively.

Table IV shows the computational timing of the BCO

algorithm and its variants on the 20 TSPLIB benchmark problems. For the experiments B and C, besides the best, the average and the standard deviation of the computational timing, the percentage of improvement (P.I.) for each instance is also shown. If a P.I. is a negative value, a reduction in terms of computational time is observed. A positive P.I. value means an increase in computational time. The experiment B is compared against experiment A. The experiment C is compared against the experiments A and B.

Considering the largest instance in Table IV, that is LIN318, the average computational time for LIN318 is reduced from 147973.4s (experiment A) to 71463.6s (experiment B). A significant reduction in terms of computational time is observed when the FRNN 2-opt is applied. When the FBPS and FRNN 2-opt are integrated with the BCO algorithm (experiment C), the average time is further reduced to 9709.6s, which is equivalent to a total of 93.44% reduction.

Considering all 20 problems together, Tables III and IV show that the solution quality is sustained while a major reduction in computational time is observed. When the experiment B is compared against experiment A, the average P.I. is -34.47%. When the experiment C is compared against experiments A and B, the average P.I. are -13.74% and -58.42% respectively.

VII. CONCLUSION

A BCO algorithm with a pruning strategy has been proposed in this paper. It is combined with an efficient implementation of 2-opt adapted from [14]. The experimental results show that the addition of both FBPS and FRNN 2-opt in the BCO algorithm is able to significantly improve its execution performance compared to the BCO algorithm proposed in [1]. The proposed algorithm will ensure that only a set of promising solutions are locally optimized by FRNN 2-opt, and hence avoid performing an exhaustive 2-opt on each and every solution generated by bees. The algorithm will be tested on other larger benchmark problems (e.g. thousands or millions cities). To achieve this, a study to develop a parallel version of the algorithm will be conducted.

ACKNOWLEDGMENT

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

REFERENCES

- [1] L. P. Wong, M. Y. H. Low, and C. S. Chong, "Bee colony optimization with local search for traveling salesman problem," in *Proc. of 6th IEEE International Conference on Industrial Informatics (INDIN 2008)*. IEEE, 2008, pp. 1019–1025.
- [2] K. von Frisch, "Decoding the language of the bee," *Science*, vol. 185, no. 4152, pp. 663–668, 1974.

- [3] 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.
- [4] F. C. Dyer, "The biology of the dance language," *Annual Review of Entomology*, vol. 47, pp. 917–949, 2002.
- [5] 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.
- [6] 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.
- [7] —, "On the performance of artificial bee colony (abc) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008.
- [8] F. H. Wedde, M. Farooq, and Y. Zhang, "Beehive: An efficient fault-tolerant routing algorithm inspired by honey bee behavior," in *Ant Colony, Optimization and Swarm Intelligence*, ser. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer, 2004, pp. 83–94.
- [9] P. Lučić and D. Teodorović, "Vehicle routing problem with uncertain demand at nodes: The bee system and fuzzy logic approach," in *Fuzzy Sets in Optimization*, J. L. Verdegay, Ed. Berlin / Heidelberg: Springer-Verlag, 2003, pp. 67–82.
- [10] D. Teodorović, "Swarm intelligence systems for transportation engineering: Principles and applications," *Transportation Research Part C: Emerging Technologies*, vol. 16, no. 6, pp. 651–657, 2008.
- [11] C. S. Chong, M. Y. H. Low, A. I. Sivakumar, and K. L. Gay, "A bee colony optimization algorithm to job shop scheduling," in *Proc. of the 2006 Winter Simulation Conference*, 2006, pp. 1954–1961.
- [12] —, "Using a bee colony algorithm for neighborhood search in job shop scheduling problems," in *Proc. of 21st European Conference on Modeling and Simulation (ECMS2007)*, 2007.
- [13] P. Lučić and D. Teodorović, "Computing with bees: Attacking complex transportation engineering problems," *International Journal on Artificial Intelligence Tools*, vol. 12, no. 3, pp. 375–394, 2003.
- [14] J. L. Bentley, "Fast algorithms for geometric traveling salesman problems," *ORSA Journal on Computing*, vol. 4, no. 4, pp. 387–441, 1992.
- [15] J. S. Gero and V. A. Kazakov, "Evolving design genes in space layout planning problems," *Artificial Intelligence in Engineering*, vol. 12, no. 3, pp. 163–176, 1998.
- [16] J. S. Gero, V. A. Kazakov, and T. Schnier, "Genetic engineering and design problems," in *Evolutionary Algorithms in Engineering Applications*, D. Dasgupta and Z. Michalewicz, Eds. Berlin: Springer, 1997, pp. 47–68.
- [17] A. S. Wu and R. K. Lindsay, "A comparison of the fixed and floating building block representation in the genetic algorithm," *Evolutionary Computation*, vol. 4, no. 2, pp. 169–193, 1996.
- [18] N. Urquhart, P. Ross, B. Paechter, and K. Chisholm, "Improving street based routing using building block mutations," in *Applications of Evolutionary Computing*, ser. Lecture Notes in Computer Science. Berlin / Heidelberg: Springer, 2002, pp. 334–341.
- [19] S. Gao, L. Zhang, F. Zhuang, and C. Zhang, "Solving traveling salesman problem by ant colony optimization algorithm with association rule," in *Proc. of 3rd International Conference on Natural Computation, 2007 (ICNC 2007)*, vol. 3, 2007, pp. 693–698.