A bee colony optimization algorithm with the fragmentation state transition rule for traveling salesman problem

L.P. Wong\textsuperscript{a}, M.Y.H. Low\textsuperscript{a}, C.S. Chong\textsuperscript{b}

\textsuperscript{a} School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798.
\textsuperscript{b} Singapore Institute of Manufacturing Technology, 71 Nanyang Drive, Singapore 638075.

Abstract

The Traveling Salesman Problem (TSP) is an NP-hard problem that has been used as a significant testing ground for many algorithms. It is also a problem that appears in many real industrial problems. This paper describes a Bee Colony Optimization (BCO) algorithm with a fragmentation state transition rule (FSTR) for TSP. The FSTR in the BCO algorithm helps bees to produce a tour by combining a few fragments of cities. Previously in [1,2], the authors suggest that bees have to travel from one city to another city in order to build a complete tour. This is a time consuming process. With the help of the FSTR, the BCO algorithm is tested on 84 benchmark problems with the dimension of [14, 1379]. The results show that while the average solution quality is 0.11\% from known optimum, the computational time is reduced by 50.45\%.

Keywords: Bee colony optimization, fragmentation strategy, traveling salesman problem, combinatorial optimization.

1. Introduction

Traveling Salesman Problem (TSP) is one of the areas that has been studied intensively in computer science and operations research. 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 (sometimes named as route or circuit) with minimum distances (or costs). The TSP is therefore to determine a permutation of a set of cities that minimizes the total round-trip distance. TSP is one of the discrete optimization problems which is classified as NP-hard. The difficulty of solving a TSP becomes obvious when the dimension of the problem increases (curse of dimensionality). The significance of TSP is shown in many domains such as logistics, transportation and semiconductor industries. Thus, TSP is still being studied by researchers from various disciplines and it remains as an important test bed for many newly developed algorithms.

Various techniques have been used to solve TSP. One of them is a method that is based on bee foraging behaviour. In a bee colony, the foraging behaviour permits bees to look for food source outside the hive and shares the information of newly discovered food source to other hive mates via an informative waggle dance. Upon flying back to its hive, a bee with food will start dancing in order to attract more bees towards the area where it foraged previously. Significant information about the food source such as its direction and distance from the hive is shown in the waggle dance. Upon flying back to its hive, a bee with food will start dancing in order to attract more bees towards the area where it foraged previously. Significant information about the food source such as its direction and distance from the hive is shown in the waggle dance. This unique behaviour is then mimicked as a useful algorithm to solve TSP. More information on waggle dance can be obtained in [3].

The bee foraging behaviour has been adapted as an algorithm to solve problems in different domains. Among them are dynamic server allocation for Internet hosting center [4], numerical function optimization [5],
telecommunication network routing [6], wood defect detection [7], stochastic vehicle routing problem [8] and job shop scheduling problem [9].

This paper describes a Bee Colony Optimization (BCO) algorithm with a fragmentation state transition rule (FSTR) for TSP. Previously in [1,2], a step-by-step state transition rule was proposed to aid bees in constructing a path by adding one city at each transition. With the FSTR, rather than adding one city, bees add a fragment of cities at each transition. While the FSTR factors in arc fitness and heuristic distance in its mechanism, it is able to reduce the computational time by limiting the neighbourhood size in the selection process of city to be visited next at each transition.

This paper starts with a discussion on some related works (Section 2). This followed by a discussion on the BCO algorithm where the FSTR is explained via an example (Section 3). Section 4 presents the experiment results and comparison study against seven benchmark algorithms. Finally, this paper ends with a conclusion.

2. Related works

There are many existing approaches for TSP. Among them are Branch and Bound, Integer Linear Programming, Iterated Lin-Kernighan heuristic, Tabu Search, Genetic Algorithm, Simulated Annealing etc. In this section, three approaches based on animal collective behaviours will be discussed, namely, Ant Colony Optimization (ACO), Bee Colony Optimization (BCO) and Particle Swarm Optimization (PSO).

Ants are able to find the shortest route from the nest to a food source. They share the information of the food source with other ants by depositing pheromone along the way between the nest and the food source. In a bee colony, bees convey the information of the newly discovered food source to other hive mates via the waggle dance (as described in Section 1). For animals such as fish and bird, their unique collective movement pattern has enabled them to avoid predators. To create such movement pattern, every individual in a group follows three rules: stay away from nearby neighbours, adopt the same direction as nearby neighbours and avoid being isolated by nearby neighbours.

Table 1 lists some existing algorithms in the literatures that will be used in our comparison study in Section 4. Please refer to their publication for details.

3. The Bee Colony Optimization algorithm

This section explains the BCO algorithm. They include an algorithm overview, fragmentation state transition rule, local search, pruning technique and waggle dance.

3.1. Overview

The outline of the BCO algorithm is shown in Fig. 1. In the algorithm, a group of bees is created during the initial stage. The number of bees, \( N_{bees} \) is set to a pre-defined number. During the first iteration, as no dance is observed, bees are initialized with a tour via an initialization procedure.

Table 1

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Abbrev.</th>
<th>Year</th>
<th>Initiator(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hybrid Discrete PSO [14]</td>
<td>HDPSO</td>
<td>2006</td>
<td>Li et al.</td>
</tr>
<tr>
<td>Bee System [16]</td>
<td>BS</td>
<td>2003</td>
<td>Lucic &amp; Teodorovic</td>
</tr>
</tbody>
</table>

Fig. 1. Overview of the BCO algorithm.

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) will check if it needs a transformation by the fixed-radius near...
neighbour 2-opt (FRNN 2-opt). The FBPS and FRNN 2-opt is further explained in Section 3.3. 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.

3.2. The fragmentation state transition rule

In [1,2], a step-by-step state transition rule is proposed to model the foraging process of bees. Every bee has to travel from one city to another city until it makes a full round trip. Before a bee starts the path construction process, it randomly observes dances performed by other bees at its hive. Once the bee has started its exploration from the hive, ($\theta$), it will be equipped with an ordered set of moves. This set of moves, named as a preferred path and denoted as $\theta$, will then serve as guidance in its foraging process. $\theta$ contains a complete path that has been explored previously by its mate. It is one of the permutations on a set of cities.

The step-by-step state transition rule, as shown in Eq. 1, is a probability function with two factors: arc fitness and heuristic distance. This probability function, $P_{\theta,n}$, determines the likelihood to move from city $i$ to city $j$ after $n$ transitions.

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

$P_{\theta,n}$ denotes the arc fitness from city $i$ to city $j$ after $n$ transitions and $d_{ij}$ denotes the distance between city $i$ and city $j$. $P_{\theta,n}$ is inversely proportional to the $d_{ij}$. In other words, under the heuristic distance influence, a bee tends to choose the next visiting city which is nearest to its current city. $\alpha$ is a binary variable that turns on or off the arc fitness influence in the algorithm. $\beta$ controls the significant level of heuristic distance.

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

The $\rho_{\theta,i,j}$ is computed for all possible paths to cities that are in $A_{\theta}$. A higher fitness value $\lambda$ is assigned to the arc which is part of $\theta$ so that the bee tends to pick the next visiting city based on its mate’s past experience. Formally, it is defined as in Eq. 2, with the following constraints: $\forall j \in A_{\theta,n}, 0 \leq \lambda \leq 1$:

$$\rho_{\theta,i,j} = \begin{cases} \frac{\lambda}{|A_{\theta,n}|}, & j \in F_{\theta,n} \wedge |A_{\theta,n}| > 1 \\ \frac{1 - \lambda}{|A_{\theta,n} - F_{\theta,n}|}, & j \notin F_{\theta,n} \wedge |A_{\theta,n}| > 1 \\ 1, & |A_{\theta,n}| = 1 \end{cases}$$  \hspace{1cm} (2)

In Eq. 2, $|A_{\theta,n} \cap F_{\theta,n}|$ is 1 when there is a common instance in both $A_{\theta,n}$ and $F_{\theta,n}$, or 0 otherwise. $A_{\theta,n} - F_{\theta,n}$ denotes the difference between sets $A_{\theta,n}$ and $F_{\theta,n}$. It contains all elements of $A_{\theta,n}$ that are not present in $F_{\theta,n}$. The first two conditions ensure that the arc that is suggested by the preferred path (if there is one) is assigned with a value of $\lambda$, whereas the rest of arcs are assigned with an equal likelihood value.

In this paper, we propose a modified version of the state transition rule, so that it becomes more scalable for problem instances with high dimension. In the proposed state transition rule, a bee will construct its path by adding a few cities (fragments of cities) at a time rather than adding one city at a time. This method is inspired by the Nearest Fragment (NF) operator demonstrated together with the Genetic Algorithm in solving TSP and microarray gene ordering [17]. The NF operator suggests that a chromosome is randomly sliced to multiple fragments. The first and the last element of each fragment are connection points. In the reconnection process of these fragments, the nearest neighbourhood heuristic is applied.

Consider an example that shows the working mechanism of the FSTR. A bee is assumed to own a complete path that is either obtained from its dance observation or from the path that it explored in previous iterations. Assume that bee owns a path through the
second scenario where $\epsilon = "A, B, C, D, E, F, G, H, I, J, A"$ is a path explored previously. This bee is also associated with a preferred path, $\theta = "I, J, H, C, E, F, G, B, A, D, I"$. To start the FSTR process, the last city in $\epsilon$ is discarded. $\epsilon$ then undergoes a slicing operation at any position so that multiple fragments of cities are produced. There are a few slicing strategies which we can choose from. Among them are variable length or fixed length fragmentation. The example in this section assumes that $\epsilon$ will undergo a variable length fragmentation. If $\epsilon$ is sliced at the positions three and eight, three different fragments are generated, “A, B, C | D, E, F, G, H | I, J”. On the other hand, if the fixed length fragmentation takes place, every resulting fragment has equal length except the last fragment, i.e. “A, B, C | D, E, F, G, H | I, J”.

At the transition $n = 0$, the bee will randomly choose any of these three fragments as part of its path. Assuming that the second fragment, “D, E, F, G, H” is chosen. Each city in the fragment will be marked as visited cities. The bee is now assumed to stop at city H and will decide which fragment to connect next to city H in the subsequent transition.

At the transition $n = 1$, in order to connect one of the remaining two fragments next to city H, cities A, C, I and J act as connection points. Cities A and C are the first and the last city of the fragment “A, B, C”. Cities I and J are the first and the last city of the fragment “I, J”. Thus, $A_{H,1} = \{A, C, I, J\}$ and $F_{H,1} = \{C\}$. The edges of (H, A), (H, C), (H, I) and (H, J) are assigned with a value of as indicated in Fig. 2 and Eq. 3.

Based on these values, Eq. 1 will determine the probability value to travel from city H to each of the city in $A_{H,1}$. Assuming that city C is chosen, the fragment of “A, B, C” will be linked next to city H in reverse order. The resulting tour after such linkage is “D, E, F, G, H, C, B, A”. If city A is chosen, the reverse connection is unnecessary and thus producing “D, E, F, G, H, A, B, C”. As cities A, B and C are now part of the resulting tour, they are marked as visited cities. The process continues until all the cities are visited.

FSTR applies the main idea (i.e. connecting fragments rather connecting a city at a time) of NF [17] to the state transition rule of BCO in [2]. The advantages of the FSTR are two-fold. Firstly, the FSTR is able to reduce the neighbourhood size in the path construction process and hence leads to lower computational cost. Secondly, instead of always picking the nearest city in its fragment reconnection process, which is a very greedy approach, other cities 

Fig. 2. Values that are assigned to edges of (H, A), (H, C), (H, I) and (H, J)

\[
\begin{pmatrix}
\frac{\rho_{HA}}{d_{HA}} & \frac{\rho_{HC}}{d_{HC}} & \frac{\rho_{HI}}{d_{HI}} & \frac{\rho_{HJ}}{d_{HJ}} \\
\frac{1 - \lambda}{\beta} & \frac{\lambda}{\beta} & \frac{(1 - \lambda)^2}{\beta} & \frac{(1 - \lambda)^3}{\beta}
\end{pmatrix}
\]

(3)

can still be selected probabilistically under the influence of arc fitness and heuristic distance in its probability calculation.

3.3. Local search and a filtering technique

After a bee has built a complete path according to the transition rule (see Section 3.2), the path will then be locally optimized by a 2-opt heuristic. In this paper, the fixed radius nearest neighbour 2-opt (FRNN 2-opt) [18] is applied for such purpose. This 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 decreases the tour length. Thus, the exploration of the second edge can be restricted to a circle centered at a node of the first edge with a radius equals to the edge length. Allowing each and every solution generated by bees to undergo the FRNN 2-opt is a strategy that leads to high computational time. 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. This strategy is named as frequency-based pruning strategy (FBPS). Whenever a solution is generated, the number of occurrences of each smallest building block is accumulated in a frequency matrix. This matrix is used to identify building blocks with high frequency and these blocks are marked as hot spots. If a tour is found to have $\kappa\%$ of building blocks that are not hot spots, the tour will then be prohibited to undergo the FRNN 2-opt. Further explanation of how the FRNN 2-opt and the FBPS are integrated in the BCO algorithm can be found in [2].
3.4. Waggle dance

Before a bee leaves its hive, it will decide if it will observe and follow a dance shown by previous dancers with a probability of \( P_{\text{follow}} \). \( P_{\text{follow}} \) is adjusted dynamically according to the profitability score of the bee and the colony based on the lookup table in Table 2, which is adopted from [4]. In the extreme case where \( P_{\text{follow}} \) is zero, the bee will keep to its own path.

\[
P_f^i = \frac{1}{L_i}, \quad L_i = \text{tour length of bee } i
\]

\[
P_f^\text{colony} = \frac{1}{N \text{Bee}} \sum_{i=1}^{N \text{Bee}} P_f^i
\]

Table 2: Lookup table for adjustment of \( P_{\text{follow}} \)

<table>
<thead>
<tr>
<th>Profitability Scores</th>
<th>( P_{\text{follow}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P_f &lt; 0.95P_f^\text{colony} )</td>
<td>0.80</td>
</tr>
<tr>
<td>( 0.95P_f^\text{colony} \leq P_f &lt; 0.975P_f^\text{colony} )</td>
<td>0.20</td>
</tr>
<tr>
<td>( 0.975P_f^\text{colony} \leq P_f &lt; 0.99P_f^\text{colony} )</td>
<td>0.02</td>
</tr>
<tr>
<td>( 0.99P_f^\text{colony} \leq P_f )</td>
<td>0.00</td>
</tr>
</tbody>
</table>

\( P_f^i \) denotes the profitability score of bee \( i \) as defined in Eq. 4. \( P_f^\text{colony} \) denotes the bee colony's average profitability as in Eq. 5 and is updated after each bee completes its tour.

After the 2-opt operation, a dance will be performed by the bee to other hive mates according to the policy as described in Section 3.1.

If a bee dances, the waggle dance will last for a certain duration. The dance duration of bee \( i \), \( D_i \), is determined by a linear function, as defined in Eq. 6.

\[
D_i = K \cdot \frac{P_f^i}{P_f^\text{colony}}
\]

According to Eq. 6, if a bee has a higher \( P_f^i \), it will dance longer. \( K \) is a user defined scaling factor that controls the magnitude of the dance duration.

4. Experiments and results

This section presents some experimental results in this study. The results reported in this study are the averages of five replications. Four performance indicators are used in this paper: \( \delta_{\text{Best}} \), \( \delta_{\text{Avg}} \), \( \delta_{\text{Worst}} \) and \( t_{\text{Avg}} \). The first three are measured in percentage. Each of them indicates the deviation from known optimum for best, average and worst case scenario. \( t_{\text{Avg}} \) is the average computational time (measured in second(s)).

The parameters of both the experiments shown in Table 3 are as follows: \( N_{\text{Bee}} = 50, \alpha = 1, \beta = 10, i = 0.95, K = 100, BC_{\text{Max}} = 10000 \) and \( \kappa = 0.10 \). We apply the fixed length fragmentation strategy in the FSTR.

Table 3 highlights the comparison of the BCO algorithm with the other seven approaches. Note that the results shown in the table are based on the experiment where the FSTR is part of the BCO algorithm. The average \( \delta \) for all the seven benchmark algorithms are computed based on the results published in the literature. The BCO algorithm with the FSTR outperforms most of the approaches except PSO+LS.

Table 3: Performance comparison of the BCO+FSTR with seven existing approaches.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Range of instances</th>
<th>( \delta_{\text{Best}} )</th>
<th>( \delta_{\text{Avg}} )</th>
<th>BCO [2] + FSTR</th>
<th>( \delta_{\text{Max}} )</th>
<th>( \delta_{\text{Avg}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>GACS+2-opt [10]</td>
<td>[51,783]</td>
<td>45</td>
<td>0.348</td>
<td>0.394</td>
<td>0.047</td>
<td>0.069</td>
</tr>
<tr>
<td>TSACS+2-opt [11]</td>
<td>[29,783]</td>
<td>11</td>
<td>2.900</td>
<td>0.087</td>
<td>0.115</td>
<td></td>
</tr>
<tr>
<td>ACO+GA [12]</td>
<td>[51,532]</td>
<td>9</td>
<td>0.101</td>
<td>0.197</td>
<td>0.038</td>
<td>0.075</td>
</tr>
<tr>
<td>PSO+LS [13]</td>
<td>[51,1379]</td>
<td>20</td>
<td>0.008</td>
<td>0.120</td>
<td>0.235</td>
<td>0.332</td>
</tr>
<tr>
<td>HDPSO [14]</td>
<td>[48,100]</td>
<td>6</td>
<td>4.540</td>
<td>5.377</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>C3DPSO [15]</td>
<td>[14,200]</td>
<td>6</td>
<td>0.223</td>
<td>1.745</td>
<td>0.000</td>
<td>0.000</td>
</tr>
<tr>
<td>BS+3-opt [16]</td>
<td>[51,1002]</td>
<td>10</td>
<td>0.623</td>
<td>0.926</td>
<td>0.072</td>
<td>0.090</td>
</tr>
</tbody>
</table>

Table 4 summarizes the average performance of the BCO algorithm on a set of 84 benchmark problems. They are obtained from TSPLIB (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/index.html). For the BCO algorithm with the step-by-step STR, the average of \( \delta_{\text{Avg}} \) is 0.31%. For the BCO algorithm with the FSTR, 0.11% is recorded. To further analyze the results, the 84 problem instances are grouped into three clusters: the first, second and third cluster consist of problems with dimension of [14, 318], [400, 783] and [1002, 1379] respectively. As shown in Table 4, while the average \( \delta \) is maintained, the average computational time for each cluster is improved when the FSTR is part of the BCO algorithm.

The BCO algorithm with the FSTR is also tested on another 11 problems with the dimension of [1400, 2392]. On average of 211391.8s (approximate of 2.5 days), \( \delta_{\text{Best}} = 0.65\% \), \( \delta_{\text{Avg}} = 0.88\% \) and \( \delta_{\text{Worst}} = 1.09\% \) are recorded.

5. Conclusion

A fragmentation state transition rule (FSTR) for the BCO algorithm has been proposed in this paper. This approach is tested on a set of 84 TSPLIB benchmark problems. With the help of the FSTR, bees no longer construct their path by adding one city at each transition. Instead, they build their path by adding a fragment of cities (multiple cities) at each transition. The results show that when the FSTR works together...
with the fixed-radius nearest neighbour 2-opt and the frequency-based pruning strategy in the BCO algorithm, it maintains an overall solution quality of 0.11\% from known optimum. At the same time, the execution performance is improved significantly. The BCO algorithm will be extended to solve other combinatorial optimization problems such as Sequential Ordering Problem and Quadratic Assignment Problem. To achieve this, a generic framework for the BCO algorithm will be developed.

Acknowledgements

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


Table 4

Performance of the BCO algorithm on a set of 84 problem instances taken from TSPLIB.

<table>
<thead>
<tr>
<th>Approaches</th>
<th>Average [14, 318]</th>
<th>58 instances</th>
<th>δAvg of 84 instances (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>δBest</td>
<td>δWorst</td>
<td>δAvg</td>
</tr>
<tr>
<td>BCO [2]</td>
<td>0.008</td>
<td>0.017</td>
<td>0.026</td>
</tr>
<tr>
<td>BCO [2] + FSTR</td>
<td>0.000</td>
<td>0.004</td>
<td>0.007</td>
</tr>
</tbody>
</table>