A Bee Colony Optimization Algorithm with Frequent-closed-pattern-based Pruning Strategy for Traveling Salesman Problem

Li-Pei Wong
School of Computer Sciences
Universiti Sains Malaysia
11800 USM, Pulau Pinang, MALAYSIA
Email: lpwong@usm.my

Shin Siang Choong
School of Computer Sciences
Universiti Sains Malaysia
11800 USM, Pulau Pinang, MALAYSIA
Email: css13_sk010@student.usm.my

***Abstract*—Bees perform waggle dance in order to communicate the information of food source to their hive mates. This unique foraging behaviour has been computationally realized as an algorithmic tool named the Bee Colony Optimization (BCO) algorithm to solve different types of Combinatorial Optimization Problems such as Traveling Salesman Problem (TSP). In order to enhance the performance of the BCO algorithm, it is integrated with a local optimization approach and a pruning strategy named as the Frequency-based Pruning Strategy (FBPS), which allows a subset of bees to undergo the local optimization and hence reduces the high processing overhead of the local optimization. Although the local optimization and the FBPS enhance the performance of the BCO algorithm, the FBPS becomes not scalable when building blocks in various sizes are considered in its pruning operation. This paper proposes a pruning strategy which employs the bi-directional extension (BIDE) based frequent closed pattern mining algorithm. It is named as the Frequent-closed-pattern-based Pruning Strategy (FCPBPS). The FCPBPS consists of two major operations: solutions accumulation and pruning operation. Solutions generated by bees are accumulated throughout the BCO algorithm execution. Based on the accumulated solutions, a set of frequent closed patterns in various sizes is mined using the BIDE algorithm. This set of frequent closed patterns is used in the FCPBPS pruning operation such that only relatively better bees (i.e. bees that produce solution which contains many of these frequent closed patterns) are allowed to undergo the local optimization. A total of 18 selected symmetric TSP benchmark problems range from 318 cities to 1291 cities are used as the testbed of this research. On average, the experimental results show that the FCPBPS requires 20.2% lesser computational time compared to the FBPS, to yield similar best-so-far TSP tour lengths.**

***Keywords*-combinatorial optimization problem; meta-heuristic; frequency-based pruning strategy; local search.**

I. Introduction

Combinatorial optimization is one of the areas that has been studied intensively in computer science and operations research. Solving a Combinatorial Optimization Problem (COP) involves finding a set of feasible solutions in a discrete search space such that only optimum solutions are identified. One of the COP examples is Traveling Salesman Problem (TSP). TSP is a NP-hard problem [1] which requires a salesman to make a round-trip tour (i.e. Hamiltonian tour) with the minimum cost based on a set of fully connected cities. The round-trip tour indicates that the salesman must visit every city once and return to the starting city.

If the dimension of a TSP being solved is large, exhaustive search (i.e. exact optimization method) is not feasible and hence a robust, approximate, and scalable technique is favoured. Many approximate optimization algorithms have been proposed to solve TSP. Swarm intelligence-based algorithms are among the approximate optimization algorithms which computationally realize insect's behaviour to solve complex optimization problems including TSP. For example, bee foraging behaviour is computationally realized as a useful algorithmic tool to solve various problems.

In a bee colony, bees travel to different sites to discover new food sources. When a new food source is discovered, upon flying back to the bee hive, waggle dances are performed by bees as a communication medium to exchange information about the newly discovered food sources (i.e. direction, distance, and amount of nectar) [2], [3]. Via waggle dances, more bees are recruited towards the newly discovered food source. Examples of the algorithms which computationally realize the bee foraging behaviour include the Artificial Bee Colony (ABC) algorithm [4], the Bee Colony Optimization (BCO) algorithm [5], [6], and their variants.

To further improve the performance, some of these algorithms are hybridized or integrated with a local search mechanism. For example, the BCO algorithm proposed by Wong et al. [7] is integrated with a local search approach. The local search is effective to generate better solutions but it incurs high computational overhead. In [8], a pruning strategy named the Frequency-based Pruning Strategy (FBPS) was introduced to allow a subset of bees to undergo the local optimization. The FBPS tracks the appearance frequency of the two-element building blocks (i.e. the smallest building blocks) and employs them in its pruning operation to disallow some bees undergo the local optimization. Further details of the FBPS can be found in Section II-B.

Although the FBPS is able to reduce the high computational overhead incurred by local search, it becomes not scalable

978-1-4673-9606-6/15/$31.00 ©2015 IEEE

when building blocks in various sizes are considered. This paper proposes a pruning strategy which is based on a frequent closed pattern mining algorithm named bi-directional extension (BIDE) algorithm [9]. The proposed pruning strategy is integrated in the BCO algorithm with a local search method named the Fixed-radius Near Neighbour 2-opt [10]. In the proposed algorithm, solutions generated by bees are accumulated throughout the BCO algorithm execution. The accumulated solutions will be undergoing a mining process using the BIDE algorithm such that a set of frequent closed patterns in various sizes is identified. Based on this set of frequent closed patterns, the pruning strategy allows a subset of relatively better bees to undergo the local optimization. The proposed pruning strategy is named as Frequent-closed-pattern-based Pruning Strategy (FCPBPS). Further details of the FCPBPS can be found in Section III.

The organization of the paper is as follows. Firstly, related work are presented in Section II. Section III describes the proposed pruning strategy based on the BIDE algorithm. Section IV presents the results. Finally, Section V ends this paper with a conclusion and highlights some future work.

II. Related Work

This section presents some related work. It first provides an overview of various bee related optimization algorithms, which are based on bee foraging behaviour. The integration of local search mechanism in these bee optimization algorithms to improve their performance will also be introduced. Then, it is followed by a discussion on how a pruning strategy works with the local search mechanism. In particularly, the FBPS will be explained in details. Since this paper proposes to replace the FBPS with a pruning strategy which is based on the sequential patten mining approach, hence, some related sequential pattern mining approaches will be discussed.

A. Bee Related Algorithms and Their Hybridization with Local Search

The Artificial Bee Colony (ABC) algorithm proposed by Karaboga is a meta-heuristic for optimization problems which is inspired by the bee foraging behaviour [4]. In [4], a solution is represented as a food source or a dance on the dance floor. There are three types of artificial bees in the ABC algorithm, namely: employed bees, onlooker bees, and scout bees. These three types of bees own different responsibilities. The responsibilities are performed in three different phases by each type of the bees. The three phases are sequentially executed for a number of iterations during the foraging process. The first phase is named as the Employed Bee Phase (EBP) where each bee is associated to a specific food source and exploitation is done. The second phase is named as the Onlooker Bee Phase (OBP) where each bee in the hive selects a dance to exploit based on a fitness proportional rule. The third phase is named as the Scout Bee Phase (SBP) where a scout bee randomly searches for a new food source (i.e. a new solution). Various variants of the ABC algorithm are employed to solve different COPs including the TSP. To increase quality of the solutions, the ABC algorithm is enriched with local search methods (e.g. 2-opt or 3-opt) as reported in [11], [12].

Teodorović and Dell'Orco proposed the Bee Colony Optimization (BCO) algorithm [13], [14]. The BCO algorithm is a multi-agent system and it can be considered as a bottom-up approach to modeling which the agents are created by analogy with bees. Each artificial bee represents one agent. Each agent produces one solution to the problem. The agents work collaboratively to solve a complex combinatorial optimization problem [14]. The algorithm consists of two alternating phases. The first phase is known as forward pass. During the forward pass, every agent explores the search space and applies a pre-defined number of moves which constructs and/or improves the solution, yielding to a new solution. They do this via a combination of individual exploration and collective experience from the past. After getting a new partial solution, every agent returns to the hive and the second phase (i.e. backward pass) begins. During the backward pass, all agents share information about their solution. Every bee can obtain the solution quality generated by all other bees. In this way, bees exchange information about the quality of the created partial solutions. These two phases are performed repetitively until they meet a stopping criterion.

Another variant of the BCO algorithm was proposed by Wong et al. This BCO variant was initially proposed to solve the symmetric TSP [15]. It computationally realizes the bee foraging behaviour by implementing a step-by-step state transition rule as its path/solution construction mechanism. The step-by-step transition rule is made up of two elements: arc fitness and heuristic distance. Before a bee starts foraging, it will probabilistically observe a waggle dance to follow. This dance becomes a preferred path of the foraging bee. When the bee constructs the path/solution, the node which appeared in preferred path is with higher arc fitness value and therefore it has higher chance to be selected by the bee as the next visiting node. On the other hand, under the influence of the heuristic distance, a bee tends to select the next nearest node from the current node. Besides, this BCO variant [15] also models the waggle dance performance as linear duration function which is formed by three parameters: dance scaling factor, profitability score of an individual bee, and bee colony's average profitability. The BCO variant is then incorporated with a local search [7], [16] and a pruning strategy (i.e. FBPS [8]) to improve the performance. The FBPS will be explained in Section II-B. Due to the high overhead, the step-by-step transition rule is replaced by a fragmentation state transition rule in order to help bees to construct a path/solution by combining a few fragments of city, rather than one city at a time [17]. The BCO algorithm is then compiled as a generic model using object-oriented approach [10] such that it is able to solve multiple combinatorial optimization problems (e.g. symmetric TSP [15], Quadratic Assignment Problem [10], Job Shop Scheduling Problem [18], asymmetric TSP [19], and Sequential Ordering Problem [20]).

B. Pruning Strategy in Local Search Mechanism

As mentioned in Section II-A, bee related optimization algorithms can be integrated with local search mechanisms when solving a TSP. In such algorithm execution, every bee will undergo a local search mechanism with the aim to improve the generated TSP solution. This is a computational expensive mechanism as it allows each and every bee to perform the local search mechanism. Therefore, a pruning strategy can be integrated to allow only a subset of promising bees to undergo the local search mechanism. Intuitively, if only a subset of bees is allowed to perform the local search, the high computational overhead can be reduced.

A pruning strategy named the Frequency-based Pruning Strategy (FBPS) is integrated in the BCO algorithm proposed by Wong et al. [8]. The working mechanism of the FBPS is as follows. A TSP solution is made up of a set of building blocks (i.e. substring of cities). Based on these accumulated building blocks obtained from the solutions generated by bees, the FBPS tracks the frequency of the smallest (i.e. two-element) building blocks using a frequency matrix. Over a period of time, hot spots can be identified. A hot spot is defined as a contiguous connection between two cities which is greater than the pre-defined (or user-defined) hot spot inclusion threshold ($\eta\%$). In other words, hot spots are common adjacent cities found in the set of the accumulated solutions. Based on the hot spots in the frequency matrix, if the dissimilarity of a solution generated by a bee is greater than or equal to a pre-defined value ($\kappa\%$) when compared to the identified hot sports, the solution will then be prohibited from undergoing local optimization. This pruning strategy will essentially allow a subset of solutions generated by bees (i.e. contain many hot spots) to undergo the local search mechanism and subsequently reduces the expensive overhead incurred by the local search. η and κ are two user-defined threshold values. They were tuned empirically trough a set of TSP experiments [8]. This static tuning is tedious and a dynamic control mechanism to regulate η and κ was proposed in [10].

The execution complexity of the FBPS relies on the building blocks tracked by the frequency matrix. Currently, it only considers the building blocks with direct adjacency (i.e. two-elements building blocks). When building blocks with more than two elements are considered, complexity of the pruning strategy in terms of computation time and data structure storage becomes difficult to handle. Hence, the FBPS is very sensitive to the number of building blocks considered in the pruning mechanism. It is not scalable when building blocks or patterns with more than two elements are considered. To make the pruning strategy to consider building blocks in different sizes, sequential pattern mining can be employed in such pruning strategy. This approach will be explained further in Section III.

C. Sequential Pattern Mining

This section provides an overview of sequential pattern mining. In particularly, the BIDE algorithm which is used to mine frequent sequential closed patterns will be discussed. The BIDE algorithm will be used in the proposed pruning strategy.

The sequential pattern mining is an essential task in data mining and it has broad applications such as market and customer analysis, web log analysis, and pattern discovery in protein sequences. For example, given a large database of customer transactions, where each transaction consists of customer identity number, transaction time, and the items bought in the transaction, sequential pattern mining can be performed over such database to understand customers' buying behaviours [21]. Sequential patterns (or frequent patterns) are subsequences that frequently appear in a sequence database, and its frequency is no less than a user-specified minimum support threshold [22].

In most of the frequent pattern mining cases, the mined frequent pattern set (i.e. mining output) is huge. The explosive number of frequent subsequences for long patterns might lead to inefficient processing in terms of computational time and space. Hence, the focus of the frequent pattern mining must not exhaustively mine all the frequent patterns but only mine the frequent closed patterns. The reason of focusing only the closed patterns mining is that the set of frequent closed pattern is a more compact yet complete set and this leads to better efficiency. A sequential pattern, S_a, is said to be closed if there is no other sequential pattern S_b, such that S_b is a supersequence of S_a. The set of the frequent closed patterns is a compact summarization of all frequent patterns [22].

One of the useful algorithms to mine a set of frequent closed patterns is the BIDE algorithm proposed by Wang and Han [9]. It was developed with the aim to answer the following three questions: (i) how to completely list the set of frequent patterns; (ii) upon getting a frequent pattern, how to check if it is closed; and (iii) how to design some search space pruning methods or other optimization techniques to accelerate the mining process. Hence, the BIDE algorithm consists of three important modules which each of these modules is used to address the questions listed above.

The frequent sequence patterns are constructed by using a lexicographical sequence tree which consists of root node that is placed at the top of the tree. This node will recursively add to the child node of the tree when each pattern is inserted. The frequent sequence enumeration mechanism is similar to the pseudo-projection-based PrefixSpan algorithm [23].

To determine if a pattern is a closed pattern, the bi-directional extension closure checking scheme is used. A pattern is considered closed when this pattern cannot be absorbed by its super-sequence with the same support. Via this checking scheme, the BIDE algorithm needs not maintain a set of historic closed patterns, thus it is a scalable algorithm in finding large number of frequent closed patterns.

To address the third question which is related to search space pruning and mining process optimization, the BIDE algorithm uses two techniques, namely: the BackScan pruning method and the ScanSkip optimization method. The BackScan pruning method is an aggressive way to mine the frequent closed patterns from the lexicographical sequence tree. This

pruning method is aggressive because it does not require the candidate-maintenance. The ScanSkip optimization technique is used after performing the BackScan pruning method to speed up the performance of the BackScan search space pruning. This technique further prunes the nodes that consist of non-closed sequential patterns which are performed by the BackScan search space pruning method.

III. Frequent-closed-pattern-based Pruning Strategy

This section explains the working mechanism of the proposed Frequent-closed-pattern-based Pruning Strategy (FCPBPS) and how it is integrated in the BCO algorithm. The pseudocode of the proposed algorithm is shown in Algorithm 1. The proposed algorithm starts with an initialization mechanism where a swarm of bees is initialized. Each bee, i.e. $B = \{b_1, b_2, \ldots, b_{N_{Bee}}\}$, is associated with a TSP solution. N_{Bee} denotes the bee population size.

The FCPBPS consists of two main operations, namely: a solutions accumulation for frequent closed patterns mining using the BIDE algorithm, and a pruning operation based on the output after the BIDE algorithm execution. The FCPBPS is integrated with the BCO algorithm proposed in [10] such that it works with a local search mechanism.

Throughout the BCO algorithm execution, the FCPBPS accumulates a set of top m TSP solutions. These top m TSP solutions are accumulated as a queue which illustrates the first-in-first-out characteristic (as denoted by Q_{top_m}). Whenever there is a better solution found, it will be inserted into a queue and the TSP tour length associated to this newly inserted solution will become the threshold value. As long as the subsequent newly found TSP solution has a shorter tour length compared to this constantly updated threshold value, it will be inserted into the queue. If the queue is full, it will discard the earliest inserted solution to make room for the better solution. The accumulation operation might be trapped in a local optima such that it is unable to accumulate any better solutions. If this happens, a reset policy which adjusts the threshold value is initiated.

This set of accumulated m TSP solutions will be periodically sent for frequent closed patterns mining using the BIDE algorithm [9]. This forms a periodic BIDE execution schedule which can be in a form of fixed (i.e. uniform) or varied interval. For example, if a fixed interval schedule is followed, the BIDE algorithm will be executed after every x iterations, where the value of x is fixed. If a varied interval schedule is followed, the value of x is changeable from time to time. The BIDE algorithm execution requires a minimum support threshold parameter (i.e. min_sup) to be determined. min_sup can range from 0.0 to 1.0 and it determines the output of the BIDE algorithm. For example, if m TSP solutions serve as the input for BIDE algorithm execution and $min\_sup = 0.2$, any frequent closed pattern mined by the BIDE algorithm which is having a support no less than the min_sup is considered as the output. The output of the BIDE algorithm is a set of frequent closed patterns and it is denoted by FCP in this paper. It is noticed that the size of the FCP depends on the min_sup. If the min_sup is set at high value, then the size of the set FCP is small (or empty) and it might require shorter time for its mining execution. On the other hand, if min_sup is set at low value, the size of the set FCP is large and it might require longer execution time. Hence, a strategy is needed such that the proposed FCPBPS is not overwhelmed with huge size of the FCP and long BIDE execution time. At the same time, the chance to obtain empty FCP (i.e. none of the frequent closed pattern is identified) must be minimized.

Algorithm 1 The pseudocode of the BCO algorithm with frequent-closed-pattern-based pruning strategy for the TSP.

```
procedure BCO_with_FCPBPS
  B ←Initialization()
  while stop criteria are not fulfilled do
    for ∀b_i ∈ B do
      b_i.constructSolution()
      if b_i ∈ top m solutions then
        Q_top_m ∪ b_i {solutions accumulation for pattern mining}
      end if
      if the periodic interval criterion is fulfilled then
        FCP ← performMining(Q_top_m) {BIDE algorithm execution}
      end if
      if FCP ≠ ∅ then
        FCP' ← performSampling(FCP)
        if b_i is similar to FCP' then
          b'_i ← b_i.performLocalSearch() {b_i is not pruned}
        end if
      else
        b'_i ← b_i.performLocalSearch()
      end if
      if Cost(b'_i) < Cost(b_i) then
        b_i ← b'_i
      end if
      b_i.performWaggleDance()
      Perform dance elitism
      Reset memory when getting trapped in local optimum
    end for
  end while
end procedure BCO_with_FCPBPS
```

Next, the pruning operation in the proposed FCPBPS is described. When a bee has constructed a TSP solution, before it undergoes a local search mechanism, it needs to pass a pruning operation. The pruning operation in the FCPBPS starts with checking the FCP. If FCP is a non-empty set (i.e. $FCP \neq \emptyset$), then it indicates a set of frequent closed patterns have been successfully identified by the BIDE algorithm and the dissimilarity detection begins. This set of FCP is then compared against every TSP solution (i.e. before local search). If the dissimilarity measure of a solution is $\geq \kappa\%$, where κ is a dissimilarity threshold, then it will be pruned from undergoing a local search. When it is pruned, it denotes the TSP solution being checked does not contain most of the frequent closed patterns in the FCP and thus it is considered as a non-promising TSP solution to undergo a local search. κ is a self-regulated parameter which takes the moving average of the dissimilarity of the last Ψ TSP solutions generated by bees at a particular moment during the algorithm execution.

There might be a situation where the size of the set FCP

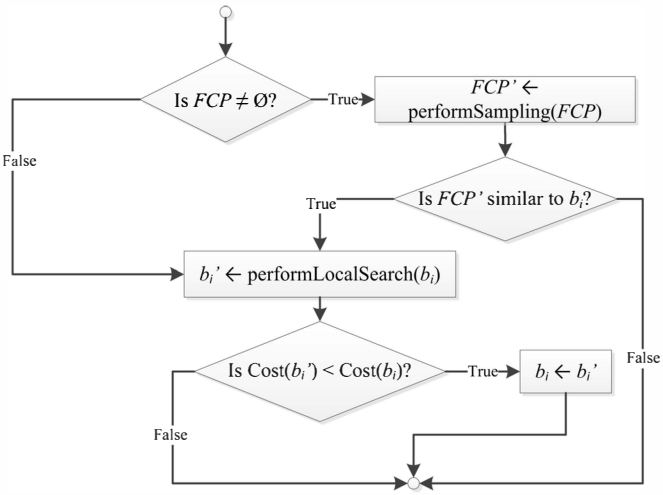

Fig. 1. The pruning operation of the proposed FCPBPS.

is large. Checking each pattern in the FCP against every TSP solution can be very time consuming. Hence, a random sampling technique which at most selects N_c patterns from the set FCP can be applied. This selected sample is named as FCP'. Size of FCP', denoted by $|FCP'|$, is at most N_c, i.e. $|FCP'| <= N_c$. N_c denotes the dimension of the TSP being solved. With this sampling technique, the pruning checking can be capped at a maximum of N_c patterns. Every time a new checking needs to be done, a new set of N_c patterns is sampled. This ensures that all the closed patterns in the FCP will essentially be considered in the pruning operation. Figure 1 illustrates the pruning operation of the proposed FCPBPS, which is also illustrated in Algorithm 1. b_i denotes the TSP solution before the local search and b_i' denotes the modified TSP solution after the local search. If the cost of b_i' is better, then b_i' will replace b_i (i.e. original TSP solution).

IV. Results and Discussions

This section presents the experimental results. It discusses the benchmark problems, experiments platform, and parameter settings. Then, a discussion on the results of this study and the performance comparison of the proposed FCPBPS against the FBPS are presented.

To conduct the experiments, 18 symmetric TSP benchmark instances obtained from the TSPLIB[1] are chosen as the testbed to evaluate the performance of the proposed FCPBPS. The dimension of the 18 selected symmetric TSP problem instances ranges from 318 to 1291 cities.

The BCO algorithm, the local search mechanism and the proposed FCPBPS are developed using the Java programming language. For the BIDE algorithm, the implementation obtained from the SPMF [24] is utilized. Since the BIDE implementation provided by the SPMF is also written in Java, the integration of all the components (i.e. the BCO algorithm, the local search, the proposed FCPBPS and the BIDE algorithm) can be done seamlessly. A workstation which is running the Ubuntu (i.e. a Linux distribution) operating system with Intel CoreTM i7-3930K 3.20 GHz processor and 16 GB Main memory (RAM) is used to run the experiments.

Unless it is stated otherwise, all the experiments were executed using the setting as follows (named as Setting φ). The BCO algorithm was executed with the following parameters: $N_{Bee} = 50$, $\beta = 1$, $\lambda = 0.1$, $K = 100$, $BC_{Max} = 10000$, and $\varpi = 25$. This shows that a total of 50 bees were employed and the BCO algorithm was executed for 10000 iterations. The local search used in the experiments was the Fixed-radius Near Neighbour 2-opt. As for the setting of the proposed FCPBPS, a total of $m = 1000$ top/best solutions were accumulated throughout the BCO algorithm execution. These accumulated solutions were sent for frequent closed patterns mining operation by the BIDE algorithm which followed an execution schedule {500, 2000, 4000, 6000, 8000}. In other words, the BIDE algorithm was executed periodically at 500-th, 2000-th, 4000-th, 6000-th, and 8000-th iteration respectively. To make sure that the size of the output for the BIDE algorithm (i.e. FCP) are within reasonable range, a declining control strategy to regulate the min_sup is applied. The min_sup for the first BIDE execution was set at 0.9. If it failed to produce any FCP, the min_sup will be lowered by 0.1. Based on the newly lowered min_sup, if FCP is successfully produced, the min_sup would remain for the subsequent BIDE execution. Also, the time to execute the BIDE algorithm was capped at a maximum of three minutes (can be less than three minutes). The pruning operation of the FCPBPS was performed in such a way that κ is a moving average of the dissimilarity of the last $\Psi = 100$ TSP solutions generated by bees.

Table I shows the performance of proposed FCPBPS on a set of 18 selected symmetric TSP benchmark problems obtained from the TSPLIB and its comparison against the FBPS. μ_L and δ_{μ_L} denote the average tour length and its deviation (in terms of percentage) from the known optimum of ten runs of experiments. For example, when the BCO+Local_Search+FCPBPS with Setting φ is applied on lin318, out of the ten runs, the average tour length is 42103.4, which is equivalent to 0.177% deviation from the known optimum, $l^* = 42029$. μ_T denotes the average computational time (measured in seconds) to yield the best-so-far tour length, based on the ten runs. To give a better picture on how these three performance indicators (i.e. μ_L, δ_{μ_L}, and μ_T) are derived, considered an experiment based on a benchmark problem with ten runs. A total of ten tour lengths $L = \{l_1, l_2, \ldots, l_{10}\}$ and ten corresponding computational times $T = \{t_1, t_2, \ldots, t_{10}\}$ are produced. Hence, $\mu_T = \frac{\sum_{i=1}^{|T|} t_i}{|T|}$, and $\mu_L = \frac{\sum_{i=1}^{|L|} l_i}{|L|}$ where $|L| = |T| = 10$. On the other hand, $\delta_{\mu_L} = \frac{\mu_L - l^*}{l^*}\%$. The LS column in Table I shows the successful local search execution (in terms of %) which are not pruned by the respective pruning strategy. Lastly, Table I also shows p-value of the Student's t-test which is used to determine if two sets of data are significantly different from each other. The two sets of data refer to the μ_L obtained from the FBPS and the proposed FCPBPS. The

[1] http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/

TABLE I
PERFORMANCE OF THE PROPOSED FCPBPS ON A SET OF 18 SELECTED SYMMETRIC TSP BENCHMARK PROBLEMS OBTAINED FROM THE TSPLIB AND ITS COMPARISON AGAINST THE FBPS.

Setting	Instance	Optimum, l^*	BCO+Local_Search+FBPS				BCO+Local_Search+FCPBPS				p-value
			μ_L	δ_{μ_L}(%)	μ_T(s)	LS (%)	μ_L	δ_{μ_L}(%)	μ_T(s)	LS (%)	
	LIN318	42029	42080.5	0.123	202.1	70.0	42103.4	0.177	191.4	50.4	0.175
	RD400	15281	15321.3	0.264	255.2	68.0	15332.9	0.340	232.9	50.2	0.102
	GR431	171414	172257.7	0.492	340.6	68.2	172336.5	0.538	409.9	49.6	0.503
	PR439	107217	107287.0	0.065	340.1	69.0	107307.9	0.085	370.4	49.8	0.263
	D493	35002	35144.7	0.408	474.7	66.8	35140.8	0.397	541.6	49.6	0.877
	ATT532	27686	27793.9	0.390	609.5	80.6	27797.2	0.402	353.2	49.6	0.870
	ALI535	202339	202607.4	0.133	848.8	80.0	202753.9	0.205	608.2	50.6	0.174
Setting φ	U574	36905	37178.3	0.741	1185.3	80.0	37185.1	0.759	512.6	50.0	0.836
	D657	48912	49163.8	0.515	1197.0	81.6	49188.0	0.564	975.9	50.2	0.390
	GR666	294358	296195.1	0.624	1228.6	80.0	296155.4	0.611	593.3	49.8	0.832
	U724	41910	42174.7	0.632	1341.6	80.6	42194.3	0.678	1148.0	49.8	0.553
	PR1002	259045	261600.4	0.986	3296.3	80.6	261761.3	1.049	1918.9	49.4	0.625
	SI1032	92650	92650.0	0.000	704.1	81.6	92650.0	0.000	393.2	71.6	-
	VM1084	239297	240784.2	0.621	3358.8	81.6	240914.0	0.676	1826.5	49.6	0.593
	D1291	50801	51027.2	0.445	2749.5	69.0	51099.2	0.587	3426.1	49.0	0.143
		Average:		0.429	1208.8			0.471	900.1		
	SI535	48450	48461.2	0.023	1144.2	78.2	48464.9	0.031	932.7	50.0	0.290
Setting ξ	RAT783	8806	8882.1	0.864	1724.8	81.4	8893.1	0.989	1712.8	50.0	0.108
	PCB1173	56892	57446.4	0.974	4030.1	81.6	57417.0	0.923	3825.7	50.0	0.644
		Average:		0.621	2299.7			0.648	2157.1		

null hypothesis (i.e. H_0) and the alternative hypothesis (i.e. H_A) are as follows. H_0: μ_L(FBPS) = μ_L(FCPBPS) and H_A: μ_L(FBPS) $\neq$ μ_L(FCPBPS). The confidence interval is set at 95%. The null hypothesis H_0 will be rejected when p-value is less than 0.05. Note that also in Table I, "-" is shown when both of the algorithms consistently obtain similar results for all the ten runs.

When Setting φ is applied on the FBPS and the FCPBPS, all the benchmark problems except SI535, RAT783, and PCB1173 show no significant difference in terms of μ_L. This can be observed via the p-value column in Table I where the p-values are bigger than 0.05. However, in terms of μ_T, the FBPS requires an average of 1208.8s whereas the FCPBPS needs an average of 900.1s, to yield the best-so-far tour length. This is equivalent to 25.5% of reduction. Two benchmark problem instances (i.e. U574 and GR666) yield a reduction of more than 50% in terms of μ_T, while maintaining the μ_L within a similar range. FBPS requires $\mu_T = 1185.3s$ and $\mu_T = 1228.6s$ whereas FCPBPS requires $\mu_T = 512.6s$ and $\mu_T = 593.3s$ for U574 and GR666 respectively. For SI535, RAT783, and PCB1173, a different setting named Setting ξ is applied. This setting follows a different BIDE execution schedule {500, 2500, 5000, 7500}. Also, the time to execute the BIDE algorithm is capped at a maximum of six minutes. In terms of μ_L, both the FBPS and the FCPBPS show no significant difference. μ_T experiences a slight drop from 2299.7s to 2157.1s, which is equivalent to 6.2% of reduction.

Although the proposed FCPBPS is able to reduce the average computational time to yield the best-so-far tour length, it has three problems as follows. First, the computational time to execute the BIDE algorithm to mine the frequent closed patterns from a set of TSP solutions is not negligible. The BIDE execution depends on the size of the input provided. If the input is large, then the BIDE execution takes longer time to complete the mining process. Second, the min_sup for the BIDE algorithm needs to be tuned. Finding a good range of values for min_sup is a challenging task as it ensures the appropriate BIDE output for the pruning process. Third, although the BIDE algorithm is able to mine frequent closed patterns in various sizes, it considers both contiguous and non-contiguous blocks. For TSP, the use of non-contiguous blocks in the pruning process might not be apparent.

V. CONCLUSIONS AND FUTURE WORK

A pruning strategy named the Frequent-closed-pattern-based Pruning Strategy (FCPBPS) is proposed to work with the Bee Colony Optimization (BCO) algorithm integrated with a local search mechanism on the symmetric Traveling Saleman Problem (TSP). The main aim of the proposed strategy is to allow a subset of bees, rather than each and every bee to undergo the local optimization. The FCPBPS accumulates a set of solutions throughput the BCO algorithm execution. This set of accumulated solutions are mined using the bi-directional extension (BIDE) algorithm such that a set of frequent closed patterns is identified. The frequent closed patterns can be considered as prevalent building blocks in the search space

and they can be used as a point of reference before sending a bee for local optimization. If a bee produces a TSP solution which contains many prevalent frequent closed patterns, it will undergo the local optimization. Otherwise, local optimization is disallowed to save computational time. A total of 18 symmetric TSP benchmark problem instances obtained from the TSPLIB are used as the testbed. On average, the results show that the FCPBPS requires 20.2% lesser computatiopal time compared to the FBPS (i.e. a pruning strategy based on frequency of two-element building blocks), to yield similar best-so-far TSP tour lengths.

Despite the proposed FCPBPS delivers some merit to reduce the average computational time to yield the best-so-far tour length, there are certain aspects which can be further enhanced. It is a challenging task to set the min_sup threshold to obtain a fixed number of patterns without running the BIDE algorithm several times and fine-tuning the min_sup. In this case, other algorithms such as algorithm for mining the top-k sequential patterns (TKS) [25] can be employed. The TKS algorithm requires a user-defined value of k such that k number of patterns can be directly output. Furthermore, the TKS is able to only mine the contiguous frequent patterns, which is more favoured in the pruning decision making for TSP solutions.

ACKNOWLEDGMENT

This work was supported by the Short-term Grant (Grant No: 304/PKOMP/6313024) and *Tabung Persidangan Luar Negara (TPLN)* at the Universiti Sains Malaysia. The authors would like to thank the Universiti Sains Malaysia for the awarded grants.

REFERENCES

[1] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W. H. Freeman Co., 1979.

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

[3] J. C. Biesmeijer and H. de Vries, "Exploration and exploitation of food sources by social insect colonies: A revision of the scout-recruit concept," *Behavioral Ecology and Sociobiology*, vol. 49, no. 2-3, pp. 89–99, 2007.

[4] D. Karaboga, "An idea based on honey bee swarm for numerical opimization," Computer Engineering Department, Engineering Faculty, Erciyes University, Tech. Rep. TR06, 2005.

[5] P. Lučić, "Modeling transportation problems using concepts of swarm intelligence and soft computing," Ph.D. Thesis, Virginia Polytechnic Institute and State University, 2002.

[6] 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.

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

[8] ——, "An efficient bee colony optimization algorithm for traveling salesman problem using frequency-based pruning," in *Proceedings of the 7th IEEE International Conference on Industrial Informatics (INDIN 2009)*. IEEE, 2009, pp. 775–782.

[9] J. Wang and J. Han, "BIDE: Efficient mining of frequent closed sequences," in *Proceedings of the 20th International Conference on Data Engineering (ICDE 2014)*. IEEE, 2004, pp. 79–90.

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

[11] M. S. Kiran, H. İşcan, and M. Gündüz, "The analysis of discrete artificial bee colony algorithm with neighborhood operator on traveling salesman problem," *Neural Computing and Applications*, vol. 23, no. 1, pp. 9–21, 2013.

[12] H. E. Kocer and M. A. Akca, "An improved artificial bee colony algorithm with local search for traveling salesman problem," *Cybernetics and Systems*, vol. 45, no. 8, pp. 635–649, 2014.

[13] D. Teodorović and M. Dell'Orco, "Bee colony optimization - A cooperative learning approach to complex transportation problems," in *Proceedings of the 16th Mini EURO Conference and 10th Meeting of EWGT*, 2005, pp. 51–60.

[14] D. Teodorović, "Bee Colony Optimization (BCO)," in *Innovations in Swarm Intelligence*, ser. Studies in Computational Intelligence, C. P. Lim, L. C. Jain, and S. Dehuri, Eds. Springer-Verlag, 2009, pp. 39–60.

[15] L. P. Wong, M. Y. H. Low, and C. S. Chong, "A bee colony optimization algorithm for traveling salesman problem," in *Proceedings of the Second Asia International Conference on Modelling & Simulation*. IEEE Computer Society, 2008, pp. 818–823.

[16] ——, "Bee colony optimization with local search for traveling salesman problem," *International Journal on Artificial Intelligence Tools*, vol. 19, no. 3, pp. 305–334, 2010.

[17] ——, "A bee colony optimization algorithm with the fragmentation state transition rule for traveling salesman problem," in *Proceedings of the 2009 Conference on Innovative Production Machines and Systems (IPROMS 2009)*, D. T. Pham, E. E. Eldukhri, and A. J. Soroka, Eds., Cardiff University, Cardiff, UK, 2009, pp. 399–404.

[18] ——, "Solving job shop scheduling problems with a generic bee colony optimization framework," in *Proceedings of the 2011 International Conference on Industrial Engineering and Systems Management (IESM 2011)*, 2011, pp. 269–280.

[19] L. P. Wong, A. T. Khader, M. Al-betar, and T. P. Tan, "Solving asymmetric traveling salesman problems using a generic bee colony optimization framework with insertion local search," in *Proceedings of the 13th International Conference onIntelligent Systems Design and Applications (ISDA2013)*, 2013, pp. 20–27.

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

[21] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Proceedings of the IEEE Eleventh International Conference on Data Engineering (ICDE 2995)*. IEEE, 1995, pp. 3–14.

[22] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast vertical mining of sequential patterns using co-occurrence information," *Advances in Knowledge Discovery and Data Mining*, vol. 8443, pp. 40–52, 2014.

[23] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 16, no. 11, pp. 1424–1440, 2004.

[24] P. Fournier-Viger, A. Gomariz, T. Gueniche, A. Soltani, C. Wu., and V. S. Tseng, "SPMF: a Java Open-Source Pattern Mining Library," *Journal of Machine Learning Research (JMLR)*, vol. 15, pp. 3389–3393, 2014. [Online]. Available: http://www.philippe-fournier-viger.com/spmf/

[25] P. Fournier-Viger, A. Gomariz, T. Gueniche, E. Mwamikazi, and R. Thomas, "Tks: Efficient mining of top-k sequential patterns," in *Advanced Data Mining and Applications*. Springer, 2013, pp. 109–120.