Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 34 (2007) 2949–2963
computers & operations research
www.elsevier.com/locate/cor

A hybrid scatter search for the probabilistic traveling salesman problem

Yu-Hsin Liu*

Department of Civil Engineering, National Chi Nan University No. 1, University Road, Puli, Nantou Hsien 545, Taiwan, ROC

Available online 20 December 2005

Abstract

The probabilistic traveling salesman problem (PTSP) is an important theoretical and practical topic in the study of stochastic network problems. It provides researchers with a modeling framework for exploring the stochastic effects in routing problems. This paper focuses on developing the hybrid scatter search (HSS) by incorporating the nearest neighbor rule (NNR), threshold accepting (TA) and edge recombination (ER) crossover into a scatter search conceptual framework to solve the PTSP. A set of numerical experiments were conducted to test the validity of the HSS based on the test problems from Tang and Miller-Hooks' study. The numerical results showed that the HSS can effectively solve the PTSP in most of the tested cases in terms of objective function value. Moreover, the results also indicated that incorporating threshold accepting into the scatter search framework can further increase the computation efficiency while maintaining solution quality. These findings show the potential of the proposed HSS in solving the large-scale PTSP.

Keywords: Stochastic combinatorial optimization; Probabilistic traveling salesman problem; Scatter search; Threshold accepting; Edge recombination crossover

1. Introduction

The probabilistic traveling salesman problem (PTSP), a type of NP-hard problem, is a basic stochastic optimization problem [1–3]. Due to the fact that the element of uncertainty not only exists, but also significantly affects, the system performance in many real-world transportation and logistic applications, the results from the PTSP can provide insights into research into other stochastic combinatorial optimization problems. The PTSP can be used to model many real-world applications in logistical and transportation planning, such as daily pickup–delivery services with stochastic demand, job sequencing involving changeover cost [4], design of retrieval sequences in a warehouse or in a cargo terminal operations [5], meals on wheels in senior citizen services [6], trip-chaining activities [7], and vehicle routing problem with stochastic demand [8].

The PTSP is an extension of the well-known traveling salesman problem (TSP), which has been extensively studied in the field of combinatorial optimization. The goal of the TSP is to find the minimum length of a tour to all customers,

* Tel.: +886 49 291 0960x4978; fax: +886 49 291 8679.
E-mail address: yuhsin@ncnu.edu.tw.

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.11.008

given the distances between all pairs of customers. The objective of the PTSP is to minimize the expected length of the a priori tour where each customer requires a visit only with a given probability. The a priori tour can be seen as a template for the visiting sequence of all customers. In a given instance, the customers should be visited based on the sequence of the a priori tour while the customers who do not need to be visited will simply be skipped. The TSP can be treated as a special case of the PTSP. The main difference between PTSP and TSP is that in PTSP the probability of each node being visited is between 0.0 and 1.0 while in TSP the probability of each node being visited is 1.0.

The closed form expressions and asymptotic analysis as well as combinatorial properties for computing the a priori expected length of any given PTSP tour were first developed by Jaillet [1,4,9]. Computational studies of several heuristic approaches modified from the TSP (e.g., nearest neighbor, savings approach, spacefilling curve, radical sorting, 1-shift, 2-opt and 3-opt exchanges) were analyzed by Bertsimas et al. [8] , Bertsimas and Howell [10], Bianchi et al. [11], and Rossi and Gavioli [12]. By using stochastic integer programming formulation, an exact algorithm based on an integer L-shaped method has been used to solve 50-node instances [13]. To efficiently and effectively solve the large-scale PTSP, recent studies focus on adopting new algorithmic approaches based on meta-heuristics such as ant colony optimization (ACO) [14–16], genetic algorithm [17], simulated annealing [18], and threshold accepting (TA) [7]. Lately, scatter search, a conceptual framework of the population-based evolutionary meta-heuristics optimization method, has been shown to yield promising outcomes for solving various complicated optimization problems [19–21]. Moreover, several studies have found that the scatter search performed better than some widely used meta-heuristics such as genetic algorithm [22–24], simulated annealing [23] and tabu search [25,26]. Therefore, based on a scatter search framework, this study devises and tests a hybrid optimization procedure for solving the PTSP.

Even though numerous studies have attested to the potential efficacy of scatter searches [19–27], an in-depth review of the literature revealed that no attempts have yet been made to solve the TSP and PTSP by scatter searches. Furthermore, despite the fact pioneers in the area have developed the conceptual framework of a scatter search for complicated optimization problems, the algorithmic procedure involved in each of the proposed components of the scatter search cannot be generalized to other problem types and needs to be carefully redesigned, refined and redefined so as to better fit the specific problem at hand. Therefore, this study designs a hybrid scatter search (HSS) procedure for solving the PTSP by incorporating the nearest neighbor rule (NNR), TA, and edge recombination (ER) crossover operator to further expand the conceptual framework and implementation of the scatter search.

To validate the effectiveness and efficiency of the proposed HSS, a set of test problems with various numbers of nodes (50, 75, 100), each with different presence probability intervals (0.0–0.2, 0.0–0.5, 0.0–1.0) as used in Tang and Miller-Hooks [7] are adopted for the purpose of comparison. The comparative results obtained can substantiate the potential of the HSS in solving the PTSP, one of the most frequently encountered problem types in real-world applications.

The remainder of this paper is organized as follows. Section 2 introduces the expressions for evaluating the a priori tour for the PTSP. Section 3 presents the details of the HSS for the PTSP. Section 4 describes the design of the numerical experiment. The results of the numerical experiments are presented and discussed in Section 5, followed by concluding comments.

2. Definition and evaluation of the PTSP

The PTSP is defined on a directed graph $G := (V, E)$, where $V := \{0, v_1, v_2, \ldots, v_n\}$ is the set of nodes or vertices, $E \subseteq V \times V$ is the set of directed edges. Node 0 represents the depot with the presence probability of 1.0. Each non-depot node v_i $(i = 1, 2, \ldots, n)$ is associated with a presence probability p_i that represents the possibility that node v_i will be present in a given realization. Given a directed graph G, the PTSP is to find an a priori tour with minimal expected length in G.

Solving the PTSP mainly relies on computing the expected length of an a priori tour. The computation of the expected length of a specific a priori PTSP tour τ, denoted as $E[\tau]$, depends on the relative location of nodes on that tour and the presence probability of each node in a given instance. By explicitly considering all realizations based on the presence of each individual node, the expected length of tour τ can be calculated. For an n-node PTSP instance, a tour τ has 2^n possible realizations. The probability of realization r_j, $p(r_j)$, can be calculated based on the presence probability of each individual node. Let $L[r_j(\tau)]$ describe the tour length of τ for realization r_j under the assumption that nodes not

in r_j are simply skipped in the tour. The expected tour length can then be formally described as

$$E[\tau] = \sum_{j=1}^{2^n} p(r_j) L[r_j(\tau)]. \tag{1}$$

The computation of expected length based on Eq. (1) is inefficient, because the computational complexity increases exponentially with an increasing number of nodes. Jaillet and Odoni [28] proposed an efficient approach to calculate $E[\tau]$ in the complexity of $O(n^3)$ for the PTSP

$$E[\tau] = \sum_{i=0}^{n} \sum_{j=i+1}^{n+1} \left\{ d_{\tau(i)\tau(j)} p_{\tau(i)} p_{\tau(j)} \prod_{k=i+1}^{j-1} (1 - p_{\tau(k)}) \right\}, \tag{2}$$

$\tau(i)$ denotes the node that has been assigned the ith stop in tour τ and $p_{\tau(i)}$ is the presence probability of node $\tau(i)$. $\tau(0)$ and $\tau(n+1)$ represent node 0, which is the depot. $d_{\tau(i)\tau(j)}$ represents the distance between nodes $\tau(i)$ and $\tau(j)$. Eq. (2) is used to calculate the expected length of an a priori tour throughout this study.

3. The hybrid scatter search (HSS) for the PTSP

As mentioned above, a scatter search is an evolutionary method that has recently been applied and shown potential for solving various complicated optimization problems [19–27]. Unlike genetic algorithm, a scatter search operates on a small set of solutions (called reference set) and makes only limited use of randomization as a proxy for diversification when searching for a globally optimal solution. In 1998 Glover [19] proposed a template to serve as the guideline of implementing scatter search. The template consists of five components. They are the diversification generation method (DGM), improvement method (IM), reference set update method (RSUM), subset generation method (SGM), and solution combination method (SCM).

As most available scatter search applications are limited to non-routing problems, the algorithmic procedures involved in most of its associated components need to be redesigned for the PTSP. Moreover, a previous study [21] incorporating a screening mechanism (i.e., TA) into the scatter search framework showed the potential of increasing the computational efficiency while maintaining solution quality. Therefore, this study proposed the HSS by adding TA to the five-component scatter search template proposed by Glover [19] for the PTSP. As shown in Fig. 1, the HSS for the PTSP consists of six components. These are: initialization (INIT), improvement method (IM), reference set update method (RSUM), subset generation method (SGM), solution combination method (SCM), and TA. When starting to solve PTSP (Iteration 0, $G = 0$), initial solutions are generated based on the NNR, which are improved by the IM. Then, RSUM is called into place to further select solutions to form the reference set, based on solution quality (objective function value) and their diversification (degree of similarity (DOS)). From the reference set the SGM is used to generate the subsets, each containing two solutions. These subsets are later used to generate the new solutions via ER crossover in the SCM. The newly generated solutions from the SCM are improved using the IM if the objective function can satisfy the criteria set in the TA. The reference set is then updated based on the objective function value and DOS to the new solutions. The solutions are allowed to evolve through successive iterations until the preset maximum number of iterations ($G_{\max}$) is met.

As can be seen, even though the proposed HSS adopted Glover's general template, the proposed HSS added the sixth component (TA) and redesign the algorithmic procedures involved in most of its associated components. Specifically, first, the NNR was adopted in the INIT to generate initial solutions (rather than the DGM as used in the original scatter search). Second, a local search procedure was used in the IM to improve the solution generated. Third, the DOS was proposed to represent the degree of diversification of the solution in the RSUM. Fourth, the ER crossover operator from genetic algorithms was used in the SCM for generating new solutions. Finally, TA was introduced as the screening mechanism for the newly generated solutions to increase the efficiency of the HSS. In the following sections, the detailed descriptions of each of the embedded components are given.

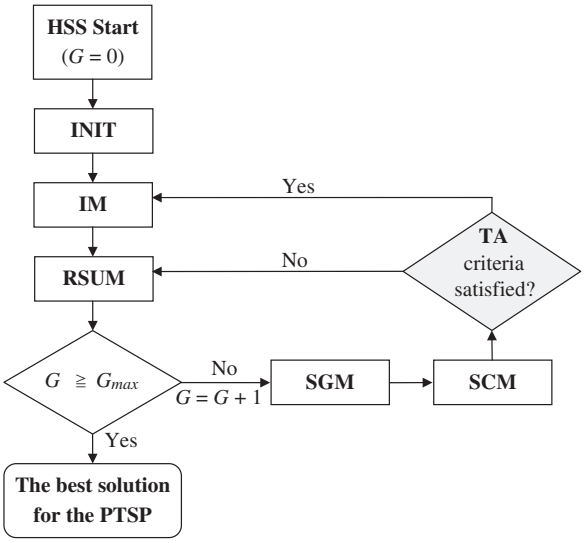

Fig. 1. The general procedure of the HSS for the PTSP.

3.1. Initialization (INIT)

The original scatter search framework used the diversification generation method to generate a collection of diverse trial solutions by employing controlled randomization and frequency memory. The diversification generation method tries to generate the solutions randomly while keeping these solutions uniformly distributed in the feasible region [19]. However, preliminary results done by the author showed that a systematic generation procedure based on the NNR yielded better outcomes than one using random generated solutions. Hence, the NNR-based procedure is adopted to systematically generate initial solutions and is described as follows.

The "Initialization (INIT)" is designed to generate m initial solutions ($m = 20$ in this study). Considering a PTSP with n nodes (excluding the depot, node 0), the farthest node, a_0, from node 0 is selected first and randomly inserted into a location between $(\lfloor (n+1)/2 \rfloor - q)$ and $(\lfloor (n+1)/2 \rfloor + q)$. The reason why $\lfloor (n+1)/2 \rfloor - q$ and $\lfloor (n+1)/2 \rfloor + q$ are used rather than the middle point of the tour $(\lfloor (n+1)/2 \rfloor)$ is to generate a set of different initial solutions. The value of q should be within the range of 1 and $\lfloor (n-1)/2 \rfloor$ to guarantee a valid tour. Based on a preliminary test conducted by the author, different values of q yielded similar objective function values with the same level of computational efforts. In this study, q is set to be 4 since it yielded a slightly better objective function value as compared to those yielded by other values of q. The NNR [29,30] is used to systematically build up the sequence of the tour. After selecting node a_0, the nearest node (a_1) from a_0 is selected and inserted in front of a_0. The second nearest node (a_2) from a_0 is selected and inserted behind a_0. Then, among the remaining nodes, the nearest node (a_3) from a_1 is selected and inserted in front of a_1, while the nearest node (a_4) from a_2 is selected and inserted behind a_2. The first initial solution (tour) is thus built by following the above rule and expressed as follows:

$$0 \cdots a_5 \; a_3 \; a_1 \; a_0 \; a_2 \; a_4 \; a_6 \cdots 0$$

To create different initial solutions, the remaining initial solutions are generated using the above rule with slight modifications. The only difference lies in whenever $l = 6, 12, 18, \ldots$, instead of using the nearest node from a_{l-2}, a_l is randomly chosen from the first or second nearest node from a_{l-2}.

3.2. Improvement method (IM)

The improvement method (IM) is used to transform the solution generated into an enhanced solution via a local search procedure. A simple local search method was used in the HSS to effectively and efficiently solve the PTSP. Later, a set of numerical experiments were conducted to test the performance of the commonly used 2-opt, 3-opt exchanges.

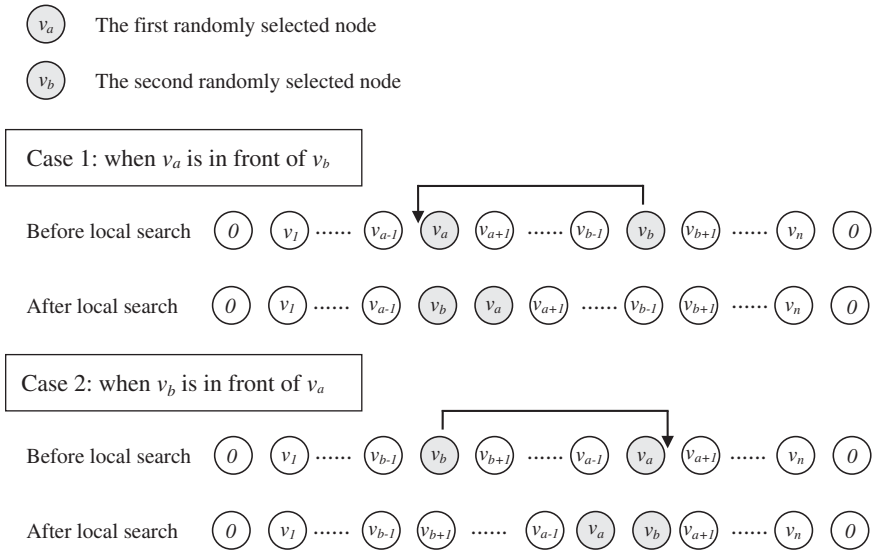

Fig. 2. The general procedure of the local search.

The local search procedure is summarized in Fig. 2 and described as follows. Assuming that one solution and its a priori tour is $(0, v_1, v_2, v_3, \ldots, v_n, 0)$ where v_i $(i = 1, 2, \ldots, n)$ represents the ith stop of the sequence in this tour. First, a node v_a is randomly selected from v_i $(i = 1, 2, \ldots, n)$, followed by randomly choosing another node v_b. If v_b is placed in sequence after v_a, as shown in case 1 of Fig. 2, the new solution will be generated by placing v_b immediately before v_a after the local search. If, on the contrary, v_b is placed in sequence before v_a, as shown in case 2 of Fig. 2, the new solution will be generated by placing v_b immediately after v_a after the local search. Generally speaking, this local search procedure mainly keeps the same sequence of the original tour by only relocating one randomly selected node to the neighborhood of the other randomly selected node. If the local search yields a better $E[\tau]$ value than the one from the original solution, the new solution will replace the original solution. If no improvement has been found after the local search, no replacement will be made. The procedure is repeated N_{IM} times for each solution ($N_{\mathrm{IM}} = 15$ in this study).

3.3. Reference set update method (RSUM)

The reference set update method (RSUM) is used to build and maintain a reference set depending on solutions' quality and diversity. The reference set, *RefSet*, is a collection of high quality ($RefSet_1$) and diverse ($RefSet_2$) solutions which will be further used to generate new solutions. When starting to solve PTSP ($G = 0$), the reference set should be built based on the quality and diversity of the "improved" initial solution from the IM. The procedure of establishing the initial reference set is described in Section 3.3.1. After iteration 0, the reference set is updated based on the existing reference set and the newly generated solution from the SCM. The procedure of updating the reference set is illustrated in Section 3.3.2.

3.3.1. Initial reference set generation

The initial reference set is generated by first selecting the best b_1 solutions ($b_1 = 5$, in this study) from the solutions improved by the IM in terms of their objective function value, $E[\tau]$. After including the best b_1 solutions in *RefSet*, the diverse solutions which are determined by DOS are included in the *RefSet* one by one via repeating the following procedure b_2 times ($b_2 = 5$, in this study). The DOS developed in this study is described first, followed by the procedure of including b_2 diverse solutions in the *RefSet*.

Assuming that one solution of the *RefSet*, $S^*(v_i^*)$, has the node sequence $(0, v_1^*, v_2^*, v_3^*, \ldots, v_n^*, 0)$, and a solution from non-*RefSet*, $S_j(v_i)$, has the node sequence $(0, v_1, v_2, v_3, \ldots, v_n, 0)$, each solution can be decomposed into $(n+1)$ edges, that is, $(0, v_1^*)$, (v_1^*, v_2^*), (v_2^*, v_3^*), $\ldots$, (v_{n-1}^*, v_n^*), $(v_n^*, 0)$ and $(0, v_1)$, (v_1, v_2), (v_2, v_3), $\ldots$, (v_{n-1}, v_n), $(v_n, 0)$.

The DOS for these two solutions, $DOS(S_j, S^*)$, can be obtained by counting the number of the same edges between these two solutions. The bigger the value of the DOS is, the more similar these two solutions are. For each solution in non-*RefSet*, the maximum *DOS*, $DOS_{\max}(S_j)$, to the solutions in *RefSet* is computed

$$DOS_{\max}(S_j) = \max_{S^* \in RefSet} \{DOS(S_j, S^*)\}, \quad S^* \in RefSet; \quad S_j \in \text{non-}RefSet. \tag{3}$$

The solution with the minimum $DOS_{\max}(S_j)$, (i.e., the most diverse solution) is added to the *RefSet* and deleted from the non-*RefSet*. Once this case arises, the values of the $DOS_{\max}(S_j)$ will need to be recomputed for all the solutions of the non-*RefSet*. This procedure is repeated b_2 times to conclude the initial reference set generation.

3.3.2. Reference set updating

After constructing the initial reference set, the SCM is used to generate the new solutions based on the subsets created by the SGM. The reference set is dynamically updated based on the quality and DOS of the new solutions generated by the SCM. A newly generated solution may become a member of $RefSet_1$ only if the new solution has a better objective function value than the solution with the worst objective value in $RefSet_1$. To become a member of $RefSet_2$, the new solution should have a smaller $DOS_{\max}(S_j)$ value than the solution with the largest $DOS_{\max}(S_j)$ value in $RefSet_2$. If none of the members in the reference set are replaced by the newly generated solutions, the INIT is used to regenerate new solutions to continue the evolution process.

3.4. Subset generation method (SGM)

Based on the solutions in the reference set, the SGM generates a 2-solution subset as a basis for creating combined solutions in the SCM. There are $\binom{b_1}{2}$ subsets by only considering b_1 high quality solutions, while there are $b_1 \times b_2$ subsets by considering both b_1 high quality solutions and b_2 diversified solutions. Based on preliminary tests, better results are obtained by using all $\binom{b_1}{2}$ subsets and randomly choosing half of $b_1 \times b_2$ subsets to generate new solutions in the SCM; this procedure is thus used in this paper.

3.5. Solution combination method (SCM)

The main purpose of the SCM is to create new solutions using a given subset of solutions generated by the SGM. Based on the results from previous studies [17,31], the ER crossover from genetic algorithms performed best when compared to other crossover strategies for both in TSP and PTSP. Therefore, ER crossover was adopted in this study for the SCM.

ER crossover was proposed by Whitley et al. [32] to solve the traditional TSP. A 5-node PTSP is used as an example to describe the procedure of ER crossover. Assuming that two solutions (tours) are chosen from the SGM—(0, 4, 3, 1, 2, 0) and (0, 1, 2, 3, 4, 0), the edges connected to each node are as follows. For node 0, the first solution indicates that node 0 connects to nodes 2 and 4 and the second solution shows that node 0 connects to nodes 1 and 4. Therefore, node 0 connects to nodes 1, 2, and 4 by considering these two solutions. Similarly, node 1 connects to nodes 0, 2, 3; node 2 connects to nodes 0, 1, 3; node 3 connects to nodes 1, 2, 4; node 4 connects to nodes 0, 3. These are the initial edge lists for each node.

The operation of the ER crossover is described below. To clearly illustrate the procedure, the *newly added node* in each step for the new solution is denoted as v_{new}. Assuming that node 0 is selected as the starting node (v_{new}) for the new solution, all edges incident to node 0 must be deleted from the initial edge list. As described, from node 0 we can go to nodes 1, 2, or 4, while nodes 1 and 2 have two active edges and node 4 has only one active edge by deleting node 0 from the initial edge list. The node with the fewest active edges, node 4, is picked as the node (v_{new}) next to node 0 in the new solution. Then, the edge list for the remaining nodes (nodes 1, 2, and 3) is further updated by deleting node 4. The updated edge list is node 1 (2, 3), node 2 (1, 3), and node 3 (1, 2). From node 4, we can only go to node 3 (as node 0 is deleted from the list already). Therefore, node 3 is chosen to be the node (v_{new}) next to node 4 in the new solution. The edge list for the remaining nodes (nodes 1 and 2) is updated by deleting node 3. Since both nodes 1 and 2 have one active edge, a random choice is made between nodes 1 and 2. If node 1 (v_{new}) is selected, the new solution is completed by adding node 2 next to node 1. Noted that if no nodes can be chosen to be added to the new

solution during the process, the remaining node with the fewest active edges is selected to be the v_{new} to continue the procedure. The procedure of ER crossover is summarized as follows:

Step 1: List the edge lists for all nodes and choose a starting node (node 0 in this study), which is the first v_{new} in the new tour.
Step 2: Update the edge lists by deleting all edges incident to the v_{new}.
Step 3: From the edge list of the v_{new}, the node with the fewest active edges is selected to be added to the new solution. If more than one node with the same active edge is present, randomly choose one of these nodes. If the edge list of the v_{new} is empty, choose the node with the fewest active edges from the remaining nodes.
Step 4: Update the v_{new}, which is the node selected in step 3.
Step 5: Repeat steps 2, 3 and 4 until all nodes are included in the tour.

The new solution is generated following the above procedure, which is then checked by the criteria of the TA mechanism before feeding into the IM.

3.6. Screening mechanisms—threshold accepting

In the original scatter search [19], all newly generated solutions were improved using IM. However, it was time-consuming to improve all the solutions using the local search. A previous study [21] found that computational efficiency could be enhanced and solution quality be maintained by improving qualified solutions that have satisfied the specific threshold value. Moreover, TA proposed by Dueck and Scheuer [33] was successfully implemented in several studies related to TSP [34–37], vehicle routing problems [37,38], and PTSP [7]. Therefore, the concept of incorporating TA as a screening mechanism into the scatter search is adopted in the HSS to solve the PTSP. A set of numerical tests were conducted to test the performance of the proposed algorithm.

To establish criteria to screen out qualified solutions, an index, $\mathscr{D}$, is defined in Eq. (4), which is based on the expected length of the a priori tour of the new solution generated from the SCM and the expected length of the a priori tour of the best solution in the $RefSet_1$

$$\mathscr{D} = \frac{E[\tau^{New}] - E[\tau^*]}{E[\tau^*]}, \tag{4}$$

where τ^* denotes the best solution in the $RefSet_1$ in terms of objective function value; τ^{New} denotes the solution generated from the SCM. $E[\tau^*]$ and $E[\tau^{New}]$ are the objective function values based on tours τ^* and τ^{New}, respectively.

A new solution generated from the SCM is improved by IM if its value is $\mathscr{D} \leqslant \mathscr{D}^*$ and skips IM if its value is $\mathscr{D} > \mathscr{D}^*$. The magnitude of threshold value $\mathscr{D}^*$ will significantly affect the performance of the proposed algorithm. The larger value $\mathscr{D}^*$ has, the more solutions need to be improved using a local search algorithm, in which case, it would yield better results in terms of objective function value with more computation effort. Two different settings of $\mathscr{D}^*$ values were used in the numerical experiments in this study. First, the value of $\mathscr{D}^*$ equals infinite for the original scatter search, which improves all solutions generated by the SCM. Second, a decreasing series of $\mathscr{D}^*$ values, as shown in Eq. (5), were used to examine the effects of incorporating TA into the HSS for solving the PTSP

$$\mathscr{D}^* = \mathscr{D}_0 \left(1.0 - \frac{G}{G_{max}}\right), \tag{5}$$

where $\mathscr{D}_0$ denotes the initial value of $\mathscr{D}^*$ ($\mathscr{D}_0 = 0.1$, in this study); G and G_{max} represent the current iteration number and the maximum number of iterations in the HSS, respectively.

3.7. The procedure after iteration zero

The newly generated solutions from the SCM and IM are used to update the reference set in terms of the objective function value and DOS to the new solutions. The above procedure is repeated until the preset maximum number of iterations (G_{max}) is met (the maximum number of iterations is set to be two times the number of nodes, i.e., $G_{max} = 2n$, in this study). However, if there are no solutions to be updated in the reference set in the current iteration, the INIT

is used to generate $(m - m_1 - m_2)$ new solutions in the next iteration, but keeping m_1 high quality solutions and m_2 diversified solutions from the current reference set ($m_1 = m_2 = 2$, in this study). The reason for keeping some solutions of the current iteration to be used in the next iteration is to ascertain that the best solution of later iterations will not be worse than the one from previous iterations. In addition, if the previous three iterations converge to the same best solution, the IM is used to improve that "converged" solution by repeating N_{IM2} times to exhaustively search the neighborhood of that "converged" solution ($N_{IM2} = 25$, in this study).

4. Numerical experiment

A numerical experiment was performed to address the following three objectives: to assess the performance of the HSS as compared to a posteriori lower bounds and the results obtained by Tang and Miller-Hooks' study; to test the effects of incorporating TA into a scatter search conceptual framework; to examine the performance of three local search strategies used in the IM (i.e., 2-opt, 3-opt exchanges and the simple local search used in the HSS) for solving the PTSP under a scatter search framework. Ninety PTSP test instances and the performance measures used in this study are described in Section 4.1, followed by a description of the design of the experiment.

4.1. Test instances and performance measures

Ninety instances generated by Tang and Miller-Hooks [7] with size $n = 50, 75$, and 100 were used as numerical experiments in this study to examine the performance of the HSS for the PTSP. Three groups of problem sets categorized by different intervals of customer presence probabilities were created for each problem size ($n = 50, 75$, and 100). Presence probabilities of customer nodes were randomly generated from a uniform distribution on intervals (0.0, 0.2), (0.0, 0.5), (0.0, 1.0), one for each problem size. The presence probability of the depot (node 0) was assigned as 1.0. Ten different problem instances were randomly generated for each presence probability of customer nodes. For each instance, the coordinates of one depot and n customer nodes (x_i, y_i) were generated based on a uniform distribution from $[0, 100]^2$. The Euclidean distance for each pair of nodes was calculated by using $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

Since the true optimal solutions for these 90 PTSP instances are not known, an a posteriori optimum based on the concept of re-optimization [8] is used to calculate the lower bound of the optimal solution for each PTSP instance. The a posteriori optimum is defined as the average of the optimal TSP lengths for each subset of customers which require a visit on a specific realization. Since the length induced by the a priori PTSP tour on a specific subset of customers cannot be smaller than the optimal TSP solutions for that subset of customers, the a posteriori optimum is a lower bound on the optimal PTSP solution. Because it requires 2^n realizations to find the optimal TSP tour, it is impractical to exactly evaluate the a posteriori optimum. Therefore, two approximations are made to obtain the "approximate" lower bound (i.e., "approximate" a posteriori optimum) of the PTSP instances [8]. First, Monte Carlo sampling from 2^n realizations is used. Second, each random sample of customers, S, is solved to near optimality as a TSP by choosing the best of $|S|/\alpha$ random tours ($\alpha = 2$) and applying to it the Lin–Kernighan [39] algorithm.

Tang and Miller-Hooks [7] focused on examining the performance of three different local search procedures (i.e., 2-opt, or-opt, and enhanced 2-opt) for the problem instances with exact and approximate solution evaluations. Even though the purpose of their study was not intended to find the best solution for these test instances, as the work by Tang and Miller-Hooks [7] was thus far the only documented values for these specific PTSP instances, the best solutions yielded in their study using three local search procedures based on both approximate and exact evaluations are used and served as the upper bounds of the optimal solutions in this study.

4.2. Experimental design

To understand the effect of incorporating TA into the scatter search and three different local searches (i.e., 2-opt, 3-opt, and the local search described in Section 3.2), six algorithmic procedures were conducted and illustrated as follows. SS+2-opt and SS+3-opt represent the original scatter search by improving all the generated solutions from the SCM using 2-opt and 3-opt, respectively. SS + TA + 2-opt and SS + TA + 3-opt denote the scatter search incorporating TA by improving only qualified solutions from the SCM using 2-opt and 3-opt, respectively. Finally, HSS0 represents

the original scatter search to improve all the solutions generated by the SCM, while HSS1 represents the scatter search incorporating TA based on a series of decreasing threshold values to improve only qualified solutions from the SCM.

5. Results and discussion

In these experiments, the average run time and solution quality of six algorithmic procedures are examined. All implementations were coded in FORTRAN and performed on an Intel Pentium IV 2.8 GHz CPU personal computer with 512 MB memory, running Windows XP operating system (3479 MFlops), and using Compaq Visual Fortran 6.5 compiler. Instead of using one run for each instance as in Tang and Miller-Hooks' (TMH) study, each instance is tested by running each of the six algorithmic procedures designed in this study 100 times using different random seeds and averaging the results in an attempt to enhance the robustness of the results (as used in [16]). Since 10 different problem instances were tested for each presence probability of customer nodes associated with a specific problem size, there was a 1000-run average for each of these six algorithmic procedures, while TMHs study was based on a 10-run average and was conducted on a DEC AlphaServer 1200/533 computer with 1 GB memory (1277 MFlops). Average statistics for the experiments are reported in Table 1.

Definitions of terms used in the column headings are given as follows. n denotes problem size, which is the number of customer nodes. p represents the customer presence probability interval $(0.0, p)$. $E[\tau]$ denotes the average optimal value of the expected length of the a priori PTSP tour. $BEST_{\text{np}}$ represents the means of the 10 best $E[\tau]$s obtained among these six algorithmic procedures in this study for problem size n with customer probability interval p. LB denotes the lower bound to the optimal PTSP solution based on a posteriori analysis. CPU is the total CPU running time in seconds. Since computation time fluctuates among various computer platforms, it may not be an objective measure for cross platform comparison, so MFlops was used instead as the basis of comparison.

In Table 1, TMH-AP and TMH-EX represent the best solution found from three local search procedures using approximate and exact solution evaluation, respectively, in TMHs study. The best average value of $E[\tau]$ among these eight optimization methods (i.e., TMH-AP, TMH-EX, SS + 2-opt, SS + TA + 2-opt, SS + 3-opt, SS + TA + 3-opt, HSS0 and HSS1) for each problem size with different presence probability intervals is highlighted in bold.

Overall the average optimal values of expected length of the a priori PTSP tour, $E[\tau]$, obtained via HSS0 and HSS1 are better than those obtained via TMH-EX (see Table 1). The only exception is when the number of nodes is 50 and the presence probability intervals are smaller [i.e., (0.0, 0.2) and (0.0, 0.5)] where the solution quality of the three methods (i.e., TMH-EX, HSS0, and HSS1) appears to be similar. In terms of computation efficiency, when the number of nodes is 75, TMH-EX and HSS0 (without TA) performed similarly; nevertheless, when the number of nodes is 100, the performance of TMH-EX in terms of computation time and the average $E[\tau]$ is worse than that of HSS0. All average values of $E[\tau]$ yielded from the SS + 2-opt and SS + TA + 2-opt are worse than the ones obtained from the HSS0 and HSS1 with about the same computational effort. Furthermore, the SS + 3-opt and SS + TA + 3-opt not only obtained the highest $E[\tau]$ values but also required much more computational effort. This indicates that the local search used in the HSS consistently performed better (i.e., with lesser $E[\tau]$ value, less computational effort) than the traditionally used 2-opt and 3-opt exchanges as a local search strategy in the PTSP.

Since the average $E[\tau]$ value yielded from the HSS (the simplest) is the lowest and the one yielded from the 3-opt exchange (the most complicated) is the highest among these three local searches, the numerical results showed that the simpler local search yielded better results than the more complicated local searches. In details, the 2-opt exchange results in a set of tour by reversing a section of tour τ; the 3-opt exchange results in a set of tours by removing a section of τ and inserting it, with or without reversal, at another place in the tour, whereas the local search used in the HSS mainly keeps the same sequence of the original tour by only relocating one randomly selected node to the neighborhood of the other randomly selected node. The obtained finding suggested that significant changes from the current solution might not be in favor of the evolving process under scatter search framework for the PTSP. Noted that the above findings are based on test instances with nodes equal to or under 100 for heterogeneous PTSP, the transferability of current findings to other contexts (e.g., nodes larger than 100 for heterogeneous PTSP, different numbers of nodes for homogeneous PTSP) should be conducted to warrant external validity of the findings.

As for the HSS0 and HSS1, both yielded very similar average values of $E[\tau]$ for all test problems, the HSS0, however, requires about 39–58% more computation time than the HSS1 does. Similar results are also found in the pair comparison of SS + 2-opt and SS + TA + 2-opt as well as SS + 3-opt and SS + TA + 3-opt. In alignment with Liu's findings [21],

Table 1
Comparison of numerical results among the six algorithmic procedures (i.e., HSS0, HSS1, SS + 2-opt, SS + TA + 2-opt, SS + 3-opt and SS + TA + 3-opt) and Tang and Miller-Hooks' study on PTSP instances

n	p	TMH-AP		TMH-EX		SS + 2-opt		SS + TA + 2-opt		SS + 3-opt		SS + TA + 3-opt		HSS0		HSS1		$BEST_{np}$	LB
		$E[\tau]$	CPU[a]	$E[\tau]$	CPU[a]	$E[\tau]$	CPU[b]	$E[\tau]$	CPU[b]	$E[\tau]$	CPU[b]	$E[\tau]$	CPU[b]	$E[\tau]$	CPU[b]	$E[\tau]$	CPU[b]		
50	0.2	227.0	24.8	**224.8**	81.0	225.3977	54.8	225.3277	40.0	226.3047	95.3	226.1941	64.5	**224.8317**	54.1	**224.8318**	38.9	224.8303	219.6895
	0.5	342.0	3.9	**341.3**	72.4	343.7170	55.2	343.8559	38.7	346.1918	97.7	346.4212	63.3	341.5706	55.0	341.5031	38.5	340.7560	331.1057
	1.0	467.5	3.7	476.4	64.8	455.8059	55.3	456.6000	37.4	462.8650	99.4	462.9192	61.6	**452.6835**	54.5	453.1582	37.2	445.5598	433.2096
75	0.2	270.0	302.0	266.2	844.2	267.7197	240.8	267.8118	165.1	269.9921	421.3	269.7747	266.5	**265.9315**	240.6	265.9343	152.5	265.8555	258.3310
	0.5	409.5	23.5	404.9	721.1	409.9771	239.1	410.3344	157.8	416.1893	427.5	416.1240	262.4	404.0113	243.7	**403.9542**	155.2	399.2976	380.6214
	1.0	548.7	12.5	549.1	648.0	539.8799	238.9	540.6778	154.8	549.6745	433.4	549.7331	260.5	**532.4545**	245.4	533.7677	156.2	515.3975	498.6566
100	0.2	309.8	342.7	301.8	4484.7	304.6202	732.5	304.7778	481.8	309.7818	1281.0	309.5559	772.6	**300.8495**	689.9	300.8700	448.6	300.7714	290.4925
	0.5	470.0	154.1	480.2	4153.2	475.6531	719.8	476.9202	463.7	487.5215	1278.8	486.9553	748.3	**462.9770**	690.3	463.3905	459.2	455.4603	432.9905
	1.0	653.6	44.5	649.0	3752.8	641.9049	819.9	641.8654	505.8	659.2151	1505.4	659.7380	869.8	**632.1090**	790.1	632.4073	512.4	606.2822	580.6438

[a]Running on DEC AlphaServer 1200/533 computer with 1 GB RAM (1277 MFlops).

[b]Running on Intel Pentium IV 2.8 GHz CPU personal computer with 512 MB RAM (3479 MFlops).

Table 2
Percentage differences between $E[\tau]$ and lower bound (LB), δ_{LB} (%)

n	p	TMH-AP	TMH-EX	SS + 2-opt	SS + TA + 2-opt	SS + 3-opt	SS + TA + 3-opt	HSS0	HSS1	$BEST_{np}$
50	0.2	3.33	2.33	2.60	2.57	3.01	2.96	2.34	2.34	2.34
	0.5	3.29	3.08	3.81	3.85	4.56	4.63	3.16	3.14	2.91
	1.0	7.92	9.97	5.22	5.40	6.85	6.86	4.75	4.60	2.85
75	0.2	4.52	3.05	3.63	3.67	4.51	4.43	2.94	2.94	2.91
	0.5	7.59	6.38	7.71	7.81	9.34	9.33	6.15	6.13	4.91
	1.0	10.04	10.12	8.27	8.43	10.23	10.24	6.78	7.04	3.34
100	0.2	6.65	3.89	4.86	4.92	6.64	6.56	3.57	3.57	3.54
	0.5	8.55	10.90	9.85	10.15	12.59	12.46	6.93	7.02	5.19
	1.0	12.56	11.77	10.55	10.54	13.53	13.62	8.86	8.91	4.42

the results from these numerical tests indicate that incorporating TA into scatter search framework can significantly improve computational efficiency while maintaining solution quality.

The lower bound (LB), obtained from a posteriori method, to the solution from these algorithmic procedures would be an upper bound of the relative error performed by these methods with respect to the PTSP optimum. If $E[\tau_{opt}]$ is truly the optimal value of a specific PTSP instance, by definition $E[\tau_{opt}] \geqq LB$. If the solution value obtained from a specific algorithmic procedures is $E[\tau_h]$, the percentage difference to the true PTSP optimum (δ_{opt}, as shown in Eq. (6)) should be less than the percentage difference to LB (δ_{LB}, as shown in Eq. (7)), i.e., $\delta_{opt} \leqq \delta_{LB}$

$$\delta_{opt} = \frac{E[\tau_h] - E[\tau_{opt}]}{E[\tau_{opt}]} \times 100\%, \tag{6}$$

$$\delta_{LB} = \frac{E[\tau_h] - LB}{LB} \times 100\%. \tag{7}$$

Since the exact values of $E[\tau_{opt}]$ for these 90 PTSP instances are unknown, δ_{LB} is used for the purpose of comparison among different algorithmic procedures (shown in Table 2). The value of δ_{LB} should be positive since finding the optimal value which is lower than the corresponding LB is impossible. As can be seen in Table 2, the values of δ_{LB} for $BEST_{np}$ are in the range of 2.34–2.91% for $n = 50$, 2.91–4.91% for $n = 75$, and 3.54–5.19% for $n = 100$, and the expected length of the best a priori tour obtained from the HSS0 and HSS1 was at most 5.19% (for the case of $n = 100$ and $p = 0.5$) higher than the tour length obtained from re-optimization. Given the fact that the obtained lower bounds are approximate, because the heuristic procedure was used to solve the TSP and there were sampling errors, it implies that the expected length of the best a priori tour obtained from the HSS0 and HSS1 may be approximately at most 5.19% higher than the tour length obtained from re-optimization. Moreover, except the case of problem size $n = 50$ and $p = 0.5$, the values of δ_{LB} for HSS0 and HSS1 (2.34–8.91%) are consistently better than those of the other six procedures (i.e., SS + 2-opt, SS + TA + 2-opt, SS + 3-opt, SS + TA + 3-opt and TMHs). The results indicate that the average values of optimal solution obtained by the HSS0 and HSS1 are at most 9% higher than the truly optimal values. Additionally, the δ_{LB} values become larger when p values become larger for all three problem sizes. This shows that more computational effort is required to obtain the optimal solution in the PTSP with generalized probability interval. Finally, as expected, the larger the problem size is, the bigger the δ_{LB} value is.

TMHs study only showed the averaged statistics of 10 instances for each problem size with a given presence probability interval. Since the results from the HSS0 and HSS1 performed better than the ones from SS + 2-opt, SS + TA + 2-opt, SS + 3-opt and SS + TA + 3-opt, detailed results of the HSS0 and HSS1 for each of the 90 instances are given in Tables 3–5 to further explore the characteristics of the optimal solutions for the PTSP. Definitions of terms used in the column headings for Tables 3–5 are given as follows. In "PTSP instances", to take "50-0.2-01" for example, it represents the instance with problem size 50, customer presence probability interval (0.0, 0.2), and instance number 01. $E[\tau]$ and $\sigma_{E[\tau]}$, respectively, denote the average and standard deviation of the expected length of 100 runs for a given PTSP instance. CPU is the 100-run average CPU running time (in seconds) for each instance. N_{Best} represents the number of runs (out of 100 runs) of finding the best $E[\tau]$ value of the expected length. Best $E[\tau]$

Table 3
Detailed results of HSS0 and HSS1 for problem size $n = 50$

PTSP instances	HSS0				HSS1				Best $E[\tau]$	LB
	$E[\tau]$	$\sigma_{E[\tau]}$	CPU	N_{Best}	$E[\tau]$	$\sigma_{E[\tau]}$	CPU	N_{Best}		
50-0.2-01	217.8268	0.000	54.0	100	217.8268	0.000	39.4	100	217.8268	214.6886
50-0.2-02	205.9634	0.001	53.6	98	205.9632	0.001	38.4	98	205.9631	197.6997
50-0.2-03	212.2929	0.002	53.9	98	212.2932	0.004	38.4	98	212.2926	205.3531
50-0.2-04	215.6969	0.001	54.9	86	215.6974	0.002	39.4	84	215.6967	213.1593
50-0.2-05	249.5913	0.037	53.9	92	249.5904	0.037	39.5	97	249.5863	243.8822
50-0.2-06	218.7962	0.000	54.6	100	218.7962	0.000	38.9	100	218.7962	215.3893
50-0.2-07	245.8009	0.000	53.2	99	245.8011	0.000	37.0	96	245.8009	238.5387
50-0.2-08	239.1635	0.000	53.8	98	239.1634	0.000	37.3	100	239.1634	235.1963
50-0.2-09	237.7384	0.001	56.0	73	237.7384	0.001	43.2	76	237.7378	233.6960
50-0.2-10	205.4463	0.017	53.5	35	205.4474	0.017	37.2	31	205.4387	199.2916
50-0.5-01	313.4487	0.429	54.3	84	313.4981	0.475	37.3	79	313.2754	306.5944
50-0.5-02	349.8649	0.312	54.7	31	349.8497	0.368	38.0	15	349.6135	335.1117
50-0.5-03	335.0868	1.444	55.2	84	335.0268	1.390	38.0	86	334.5397	326.6912
50-0.5-04	348.8993	2.438	55.1	76	348.4791	1.473	39.6	83	348.0985	339.8941
50-0.5-05	327.0622	0.031	56.7	99	327.0676	0.063	41.5	98	327.0590	319.8797
50-0.5-06	376.0036	0.004	55.2	0	375.9711	0.316	39.5	1	372.8447	363.9523
50-0.5-07	324.3353	0.116	54.1	88	324.3203	0.021	37.7	91	324.3143	316.1415
50-0.5-08	338.7731	3.852	54.3	12	338.2905	1.888	37.4	16	336.8873	326.1261
50-0.5-09	352.7201	0.962	55.5	39	353.0536	1.420	38.8	23	352.3847	338.0644
50-0.5-10	349.5115	1.268	54.4	5	349.4741	0.990	37.2	8	348.5425	338.6011
50-1.0-01	484.5245	17.774	54.0	16	487.2461	15.731	37.7	9	453.8909	443.1984
50-1.0-02	406.7128	1.833	54.9	1	406.8130	1.752	37.7	0	400.3352	390.7903
50-1.0-03	464.8456	2.959	53.5	15	464.2556	3.074	36.9	21	460.5522	447.1663
50-1.0-04	457.9771	2.373	55.5	93	458.1937	3.014	38.0	89	457.5278	448.6270
50-1.0-05	425.6183	4.195	53.4	5	426.3734	5.924	36.9	11	422.4490	402.6439
50-1.0-06	500.9560	7.338	53.9	38	501.3387	8.266	36.0	41	495.3570	481.1812
50-1.0-07	486.9585	15.119	54.9	4	487.0555	16.173	37.2	4	476.5402	459.8339
50-1.0-08	441.6687	1.941	55.6	4	441.8262	1.630	38.1	2	436.6021	428.2492
50-1.0-09	439.9052	1.839	54.7	2	440.4553	1.258	37.2	1	435.3252	423.8814
50-1.0-10	417.6682	0.680	54.6	6	418.0240	1.571	36.5	10	417.0180	406.5244

denotes the best optimal value that can be found by these six algorithmic procedures conducted in this study for a given PTSP instance.

Tables 3–5 show the detailed results for problem size $n = 50, 75$ and 100, respectively. The results indicate that the HSS1 (with TA) can more efficiently obtain a competitive solution quality in comparison to the HSS0 in all PTSP instances. Due to the concept of stochastic optimization adopted in the HSS, the values of average $E[\tau]$ using the HSS1 are sometimes better than those of using the HSS0. However, the values of $\sigma_{E[\tau]}$ and N_{Best} somehow tend to be instance-dependent for the same combination of n and p. For example, the values of $\sigma_{E[\tau]}$ for instance numbers 01 and 07 are much higher than those of the other instances in $n = 50$ and $p = 1.0$; the value of N_{Best} for instance number 10 is much smaller than those of the other instances in $n = 50$ and $p = 0.2$; similar outcomes are found in other combinations of n and p. The fact that some instances under the same combination of n and p have smaller N_{Best} values where others have higher N_{Best} values indicates that the possibility of finding the optimal solution for some instances based on the HSS may fluctuate across instances. While the occasional inconsistencies found among instances may suggest that the effectiveness of finding the optimal solution might be mediated by some factors, such as the characteristics of the instance (e.g., the relative location and presence probability of each node), another notable observation is with regard to the inadequacy of past studies that relied on a single-instance numerical experiment. Explicitly, one PTSP test instance for a specific combination of n and p, as used in some of the previous studies [15,16], assumed that same combination of n and p will yield similar results by the same method. As could be seen from the results obtained in this study, shown in Tables 3–5, the assumption is not held and the numerical experiment based on one PTSP test

Table 4
Detailed results of HSS0 and HSS1 for problem size $n = 75$

PTSP instances	HSS0				HSS1				Best $E[\tau]$	LB
	$E[\tau]$	$\sigma_{E[\tau]}$	CPU	N_{Best}	$E[\tau]$	$\sigma_{E[\tau]}$	CPU	N_{Best}		
75-0.2-01	267.2297	0.174	242.8	21	267.2363	0.187	153.8	21	267.1769	255.9436
75-0.2-02	278.1329	0.002	239.8	47	278.1339	0.013	152.4	54	278.1316	272.0016
75-0.2-03	268.2563	0.024	240.6	74	268.2608	0.035	151.5	76	268.2517	262.0525
75-0.2-04	271.5160	0.059	242.2	59	271.5119	0.054	151.1	76	271.4945	263.9936
75-0.2-05	264.5358	0.261	237.8	47	264.5533	0.268	152.4	49	264.3252	257.9032
75-0.2-06	236.2287	0.757	239.5	46	236.2245	0.758	154.0	46	235.9334	224.9284
75-0.2-07	273.2101	0.071	238.4	36	273.2233	0.075	150.3	34	273.1455	263.8956
75-0.2-08	251.4527	0.004	241.6	86	251.4539	0.012	157.8	77	251.4515	248.8204
75-0.2-09	280.4426	0.180	241.7	25	280.4349	0.171	152.1	29	280.3401	270.6063
75-0.2-10	268.3100	0.009	242.0	47	268.3102	0.009	149.6	46	268.3049	263.1644
75-0.5-01	401.6708	2.661	245.9	5	401.6105	2.994	160.8	8	395.8143	374.1341
75-0.5-02	424.1370	10.957	243.3	48	424.7521	11.813	153.5	47	416.6071	395.4633
75-0.5-03	430.0806	2.523	242.8	1	430.3539	2.415	157.0	0	425.7107	410.5180
75-0.5-04	404.1698	0.362	241.5	14	404.1872	0.566	150.5	17	403.7272	383.8160
75-0.5-05	401.9754	4.332	243.9	6	401.6037	5.098	151.5	7	396.6220	372.4662
75-0.5-06	379.2370	0.423	240.9	7	379.2818	0.393	157.8	6	378.1053	365.4089
75-0.5-07	427.5199	6.811	247.1	1	428.0349	6.142	154.2	0	418.1928	397.7712
75-0.5-08	400.7357	2.403	241.1	2	399.9225	2.933	155.6	2	394.1244	371.9494
75-0.5-09	368.0461	0.375	241.4	35	367.9580	0.356	151.6	43	367.6530	358.5981
75-0.5-10	402.5403	8.369	248.9	47	401.8370	7.809	159.4	47	396.4189	376.0886
75-1.0-01	517.6145	8.472	243.0	1	518.8125	10.061	151.0	1	508.2188	495.7458
75-1.0-02	544.0231	10.928	248.0	0	548.9797	13.745	155.9	1	525.1215	514.0083
75-1.0-03	483.0474	6.831	241.8	48	483.7197	9.153	156.4	50	477.8853	463.8400
75-1.0-04	589.4052	11.633	247.7	1	591.9009	11.868	155.9	0	566.4671	540.7981
75-1.0-05	534.6487	7.761	247.6	1	534.5060	7.222	154.8	1	523.1762	507.0453
75-1.0-06	496.1454	8.641	247.8	0	496.1859	6.803	163.1	1	483.6226	455.7104
75-1.0-07	530.3000	11.259	245.6	1	530.7877	9.007	160.0	0	498.9775	485.0399
75-1.0-08	572.1857	17.318	243.9	1	575.8030	17.657	156.9	1	538.5227	521.5663
75-1.0-09	534.7928	8.456	240.2	5	532.9310	7.500	150.2	7	524.4635	508.5160
75-1.0-10	522.3820	18.545	248.1	7	524.0509	20.104	158.2	4	506.7259	494.2957

instance may be subject to problems caused by biased sampling of the PTSP instances, which may inadvertently result in over-estimation or under-estimation of the performance of proposed methods for PTSP. As such, to obtain more robust results, researchers are advised to include at least a certain amount of test instances with the same combination of n and p while conducting PTSP numerical experiments.

6. Conclusions

In this paper, the hybrid scatter search (HSS) is developed to solve the PTSP. Specifically, the algorithm incorporating the nearest neighbor rule (NNR), edge recombination (ER) crossover operator and threshold accepting (TA) into scatter search conceptual framework is found to provide the highest quality solutions for efficiently solving the PTSP as compared to the other methods. The numerical results show that the HSS1 (incorporating TA) yields the most promising solutions by considering the balance between solution quality and computation efficiency. Incorporating TA into scatter search framework makes it possible to tackle large size PTSP instances with significantly reduced computational effort while maintaining solution quality.

To summarize, the contributions of this paper are two-fold. First, the algorithm for applying the scatter search in solving the PTSP is delineated and its effectiveness investigated. The degree of similarity (DOS), used to identify the degree of diversification of the solutions, is proposed. This concept has great generalization value to other path or

Table 5
Detailed results of HSS0 and HSS1 for problem size $n = 100$

PTSP instances	HSS0				HSS1				Best $E[\tau]$	LB
	$E[\tau]$	$\sigma_{E[\tau]}$	CPU	N_{Best}	$E[\tau]$	$\sigma_{E[\tau]}$	CPU	N_{Best}		
100-0.2-01	277.0741	0.038	691.1	17	277.0725	0.037	444.7	20	277.0646	270.8800
100-0.2-02	298.5914	0.018	690.5	37	298.5917	0.017	448.3	30	298.5833	288.4480
100-0.2-03	309.5821	0.019	685.2	29	309.5882	0.036	440.6	31	309.5724	301.0215
100-0.2-04	298.7707	0.020	689.9	26	298.7709	0.018	446.6	28	298.7581	290.9564
100-0.2-05	319.7071	0.218	691.9	11	319.7043	0.186	439.8	15	319.6604	307.3508
100-0.2-06	301.1365	0.040	694.8	12	301.1450	0.045	461.0	9	301.0939	289.5183
100-0.2-07	301.6730	0.330	685.8	12	301.6719	0.354	453.3	9	301.2825	289.0584
100-0.2-08	293.6836	0.166	694.0	21	293.8878	1.502	460.4	24	293.5226	280.3611
100-0.2-09	304.4171	0.122	688.6	19	304.4350	0.156	451.2	10	304.3593	294.5659
100-0.2-10	303.8598	0.195	687.2	18	303.8326	0.023	439.7	17	303.8172	292.7645
100-0.5-01	460.8128	4.973	688.2	14	460.4716	4.252	455.1	20	457.6049	439.0936
100-0.5-02	470.7284	6.846	691.9	1	470.6758	7.621	452.7	1	456.9695	430.8148
100-0.5-03	466.4306	6.060	698.0	0	466.5975	7.411	452.7	3	460.7578	442.1253
100-0.5-04	453.4958	10.451	691.1	2	452.8060	10.590	460.8	1	443.1313	421.8299
100-0.5-05	477.2997	5.631	688.3	3	478.0997	6.174	469.2	2	463.3070	437.2832
100-0.5-06	487.6004	6.973	694.4	2	488.8272	7.416	460.8	1	480.9816	456.1878
100-0.5-07	458.1454	7.227	689.6	5	459.3902	7.526	450.8	3	449.0551	429.2602
100-0.5-08	451.8641	2.622	686.1	8	451.8133	2.706	462.0	4	449.9378	425.5203
100-0.5-09	461.1461	8.470	686.9	36	463.4657	10.896	460.3	13	458.1998	438.3832
100-0.5-10	442.2463	5.368	688.2	2	441.7575	5.288	467.2	1	434.6582	409.4063
100-1.0-01	630.6988	9.900	674.7	1	630.4203	9.273	448.6	0	600.8967	581.7041
100-1.0-02	630.4584	11.483	831.8	4	632.6600	13.409	531.6	5	617.2408	584.7798
100-1.0-03	648.3107	13.395	907.9	0	651.2759	15.686	582.1	1	618.1410	591.7672
100-1.0-04	652.3943	11.583	759.7	1	650.1259	9.427	484.6	0	625.6162	590.1654
100-1.0-05	597.6962	12.662	715.9	1	600.4709	15.554	470.2	0	571.9203	549.3956
100-1.0-06	671.3405	13.202	875.0	0	667.8283	13.999	567.4	1	637.0537	613.0466
100-1.0-07	580.3668	10.963	682.0	1	579.5737	8.792	447.9	0	561.3058	533.5301
100-1.0-08	632.7219	8.493	746.3	1	633.8284	8.634	502.0	0	620.4662	598.3241
100-1.0-09	656.1888	25.973	1023.8	1	656.3908	25.680	643.9	0	612.7653	592.4442
100-1.0-10	620.9131	12.443	683.6	0	621.4990	12.775	445.4	1	597.4162	571.2806

routing related problems, such as (stochastic) vehicle routing problems, etc. Second, the threshold accepting used as the screening tool for local search is further incorporated into the scatter search framework to improve the efficiency of the traditional scatter search.

Acknowledgements

The author is indebted to Dr. Miller-Hooks for providing me with the test instances to be used in this paper. Special thanks are extended to two anonymous referees for their constructive and informative suggestions for the refinement of the manuscript.

References

[1] Jaillet P. Probabilistic traveling salesman problems. Dissertation, Massachusetts Institute of Technology, USA, 1985.
[2] Bertsimas D. Probabilistic combinatorial optimization problems. Dissertation, Massachusetts Institute of Technology, USA, 1988.
[3] Bertsimas D, Jaillet P, Odoni AR. A priori optimization. Operations Research 1990;38(6):1019–33.
[4] Jaillet P. A priori solution of a traveling salesman problem in which a random subset of the customers are visited. Operations Research 1988;36(6):929–36.
[5] Bartholdi JJ, Platzman LK. Heuristics based on spacefilling curves for combinatorial problems in Euclidean space. Management Science 1988;34(3):291–305.

[6] Bartholdi JJ, Platzman LK, Collins RL, Warden WH. A minimal technology routing system for meals on wheels. Interfaces 1983;13(3):1–8.
[7] Tang H, Miller-Hooks E. Approximate procedures for the probabilistic traveling salesperson problem. Transportation Research Record 2004;1882:27–36.
[8] Bertsimas D, Chervi P, Peterson M. Computational approaches to stochastic vehicle routing problems. Transportation Science 1995;29(4): 342–52.
[9] Jaillet P. Stochastic routing problems. In: Andreatta G, Mason F, Serafini P, editors. Advanced school on stochastics in combinatorial optimization. Singapore: World Scientific Publisher; 1987. p. 192–213.
[10] Bertsimas D, Howell L. Further results on the probabilistic traveling salesman problem. European Journal of Operational Research 1993;65(1): 68–95.
[11] Bianchi L, Knowles J, Bowler N. Local search for the probabilistic traveling salesman problem: correction to the 2-p-opt and 1-shift algorithms. European Journal of Operational Research 2005;162(1):206–19.
[12] Rossi F, Gavioli I. Aspects of heuristic method in the "probabilistic traveling salesman problem". In: Andreatta G, Mason F, Serafini P, editors. Advanced school on stochastics in combinatorial optimization. Singapore: World Scientific Publisher; 1987. p. 214–27.
[13] Laporte G, Louveaux F, Mercure H. A priori optimization of the probabilistic traveling salesman problem. Operations Research 1994;42(3): 543–9.
[14] Bianchi L, Gambardella LM, Dorigo M. An ant colony optimization approach to the probabilistic traveling salesman problem. In: Merelo Guervós JJ, Adamidis P, Beyer H-G, Fernández-Villacañas J-L, Schwefel H-P, editors. Proceedings of the seventh parallel problem solving from nature (PPSN VII), Lecture notes in computer science, vol. 2439. Berlin: Springer; 2002. p. 883–92.
[15] Bianchi L, Gambardella LM, Dorigo M. Solving the homogeneous probabilistic traveling salesman problem by the ACO metaheuristic. In: Dorigo M, DiCaro G, Sampels M, editors. Proceedings of third international workshop ANTS 2002, Lecture notes in computer science, vol. 2463. Berlin: Springer; 2002. p. 176–87.
[16] Branke J, Guntsch M. Solving the probabilistic TSP with ant colony optimization. Journal of Mathematical Modelling and Algorithms 2004;3(4):403–25.
[17] Liu Y-H, Jou R-C, Wang C-J. Genetic algorithms for the probabilistic traveling salesman problem. In: Proceedings of the conference on E-logistics, Taoyuan, Taiwan. 2004. p. 77–82.
[18] Bowler NE, Fink TMA, Ball RC. Characterization of the probabilistic traveling salesman problem. Physical Review E 2003;68(3):036703.
[19] Glover F. A template for scatter search and path relinking. In: Hao J-K, Lutton E, Ronald E, Schoenauer M, Snyers D, editors. Artificial evolution, Lecture notes in computer science, vol. 1363. Berlin: Springer; 1998. p. 3–51.
[20] Laguna M, Martí R. Scatter search: methodology and implementations in C. London: Kluwer Academic Publishers; 2003.
[21] Liu Y-H. Incorporating scatter search and threshold accepting in multinomial probit model estimation. Technical Report NSC 93-2416-H-260-002, National Science Council, Taiwan, 2005.
[22] Bulut G. Robust multi-scenario optimization of an air expeditionary force: force structure applying scatter search to the combat forces assessment model. Thesis, Air Force Institute of Technology, USA, 2001.
[23] Hung WNN, Song X, Aboulhamid EM, Driscoll MA. BDD minimization by scatter search. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 2002;21(8):974–9.
[24] Liu Y-H, Jou R-C, Yeap B-K. Multinomial probit (MNP) model estimation—comparisons of different optimization methods. Journal of the Chinese Institute of Civil and Hydraulic Engineering 2005, in press.
[25] Glover F, Laguna M, Martí R. Scatter search and path relinking: advances and applications. In: Glover F, Kochenberger G, editors. Handbook of metaheuristics. Boston: Kluwer Academic Publishers; 2003. p. 1–36.
[26] Greistorfer P. A tabu scatter search metaheuristic for the arc routing problem. Computer and Industrial Engineering 2003;44(2):249–66.
[27] García-López F, Melián-Batista B, Moreno-Pérez JA, Moreno-Vega JM. Parallelization of the scatter search for the p-median problem. Parallel Computing 2003;29(5):575–89.
[28] Jaillet P, Odoni AR. The probabilistic vehicle routing problem. In: Golden BL, Assad AA, editors. Vehicle routing: methods and studies. Amsterdam: North-Holland; 1988. p. 293–318.
[29] Rosenkrantz DJ, Stearns RE, Lewis II PM. An analysis of several heuristics for the traveling salesman problem. SIAM Journal of Computing 1977;6(3):563–81.
[30] Hurkens CAJ, Woeginger GJ. On the nearest neighbor rule for the traveling salesman problem. Operations Research Letters 2004;32(1):1–4.
[31] Potvin JY. Genetic algorithms for the traveling salesman problem. Annals of Operations Research 1996;63:339–70.
[32] Whitley D, Starkweather T, Fuquay D. Scheduling problems and traveling salesmen: the genetic edge recombination operator. In: Proceedings of the third international conference on genetic algorithms (ICGA '89). Palo Alto, CA: Morgan Kaufmann; 1989. p. 133–40.
[33] Dueck G, Scheuer T. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. Journal of Computational Physics 1990;90(1):161–75.
[34] Schneider J, Froschhammer C, Morgenstern I, Husslein T, Singer JM. Searching for backbones—an efficient parallel algorithm for the traveling salesman problem. Computer Physics Communications 1996;96(2–3):173–88.
[35] Schrimpf G, Schneider J, Stamm-Wilbrandt H, Dueck G. Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 2000;159(2):139–71.
[36] Tafelmayer R, Hoffmann KH. Scaling features in complex optimization problems. Computer Physics Communications 1995;86(1–2):81–90.
[37] Tarantilis CD, Kiranoudis CT, Vassiliadis VS. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. European Journal of Operational Research 2004;152(1):148–58.
[38] Bräysy O, Hasle G, Dullaert W. A multi-start local search algorithm for the vehicle routing problem with time windows. European Journal of Operational Research 2004;159(3):586–605.
[39] Lin S, Kernighan BW. An effective heuristic algorithm for the traveling salesman problem. Operations Research 1973;21(2):498–516.