The Effects of Initial Solution Generators under Genetic Algorithm Framework for the Heterogeneous Probabilistic TSP

Yu-Hsin Liu[1]

Abstract. The probabilistic travelling salesman problem (PTSP) is a topic of theoretical and practical importance in the study of stochastic network problems. It provides researchers with a modelling framework for exploring the stochastic effects in routing problems. This paper focuses on investigating the performance of three initial solution generators under genetic algorithm framework for solving the heterogeneous PTSP. A set of numerical experiments based on heterogeneous PTSP instances were conducted to test the validity of three generators and their combinations. Though initial statistical testing by the permutation tests did not yield satisfactory results, it, however, directed the researcher to phenomenon of research potential. Specifically, the performance of the three generators seem to be instance-dependent for the same combination of *n* and *p*. Based on the findings, it is suggested that algorithms that embed the notion of adaptively selecting the initial solution generator based on some experiences in searching ability, or that can take into account of the patterns of individual instance could be promising strategies for solving the PTSP. Finally, to obtain more valid 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.

1 INTRODUCTION

The probabilistic travelling salesman problem (PTSP) is an extension of the well-known travelling 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, given the distances between all pairs of customers whereas 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 [1, 2, 3]. The main difference between the PTSP and the TSP is that in the 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. 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 logistics applications, the results from the PTSP can provide insights into research in other probabilistic combinatorial optimization problems. Moreover, the PTSP can also 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, design of retrieval sequences in a warehouse or in a cargo terminal operations, meals on wheels in senior citizen services, trip-chaining activities, vehicle routing problem with stochastic demand, and home delivery service under e-commerce [4, 5, 6, 7, 8].

Early PTSP computational studies, dating from 1985, adopted heuristic approaches that were modified from the TSP (e.g., nearest neighbour, savings approach, spacefilling curve, radial sorting, 1-shift, and 2-opt exchanges) [1, 3, 9, 10, 11, 12]. With its less than satisfactory performance in yielding solution quality, researchers in the recent years switch to metaheuristic methods, such as ant colony optimization [13, 14], evolutionary algorithm [15], genetic algorithm [16], simulated annealing [17], threshold accepting [8] and scatter search [18, 19, 20]. Since the genetic algorithm (GA), a conceptual framework of the population-based metaheuristic method, has been shown to yield promising outcomes for solving the PTSP [16] and various complicated optimization problems in the past three decades [21, 22, 23, 24, 25], this study will further investigate the effects of different methods based on random generation and nearest neighbour algorithms for generating initial solutions under GA framework for solving the PTSP. To validate the effectiveness and efficiency of the proposed algorithmic procedure, a set of heterogeneous (90 instances) PTSP test instances as used in the previous studies [8, 16, 18, 19, 20] will be used as the base for comparison purpose.

The remainder of this paper is organized as follows. In the next section, expressions for exactly and approximately evaluating the a priori tour for the PTSP are introduced. The details of the proposed algorithmic procedure for the PTSP are then described. The results of the numerical experiments are presented and discussed in the next section, followed by concluding comments.

2 DEFINITION AND EVALUATION OF THE PTSP

The PTSP is defined on a directed graph $G := (V, E)$, where $V := \{0, v_1, v_2, ..., v_n\}$ 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 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* Hamiltonian 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 $E[\tau]$,

[1] Dept. of Civil Engineering, National Chi Nan Univeristy, Puli, Nantou Hsien 54561, Taiwan. Email: **yuhsin@ncnu.edu.tw**

depends on the relative location of nodes on that tour and the presence probability of each node in a given instance. Jaillet and Odoni [26] proposed an approach to exactly calculate $E[\tau]$ in the complexity of $O(n^3)$ for the PTSP.

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

d_{ij} represents the distance between nodes i and j; $\tau(i)$ denotes the node that has been assigned the i^{th} stop in tour τ and $p_{\tau(i)}$ is the probability of node $\tau(i)$. $\tau(0)$ and $\tau(n+1)$ represent node 0, which is the depot.

Even though (1) yields a polynomial evaluation time for the PTSP, the resulting $O(n^3)$ time for calculating $E[\tau]$ is still very long, especially for metaheuristic methods which need to repeatedly evaluate the objective function value $E[\tau]$. In this study, the proposed algorithm needs to repeatedly compare two solutions (i.e., the new solution before and after local search procedure, which is described in the next section) based on their values of $E[\tau]$. Therefore, the depth approximate evaluation [14, 18, 20] was used to increase the computation efficiency of the proposed algorithm.

$$E_{\lambda}^{AP}[\tau] = \sum_{i=1}^{n} \sum_{j=i+1}^{\min\{n+1, i+\lambda\}} \{d_{\tau(i)\tau(j)} p_{\tau(i)} p_{\tau(j)} \prod_{k=i+1}^{j-1} (1 - p_{\tau(k)})\} \quad (2)$$

The only difference between (1) and (2) is the choice of truncation position λ in (2). Equation (2) will have the computational complexity of $O(n\lambda^2)$, instead of $O(n^3)$ in (1). It is easy to see that (2) becomes more accurate when λ increases. A larger value of λ, however, requires more computation efforts for the computation of (2). Equation (2) can perform a very good approximation of $E[\tau]$ with a smaller value of λ when the value of $p_{\tau(k)}$ gets larger, because $\prod_{k=i+1}^{j-1} (1 - p_{\tau(k)})$ will yield a very small value and can be omitted. Nevertheless, Equation (2) will need a larger value of λ to perform a good approximation when the value of of $p_{\tau(k)}$ is small. The detailed procedure of using approximate evaluation shown in (2) to accelerate the algorithm is described in the next section.

3 SOLUTION ALGORITHMS

The proposed genetic algorithm consists of four components as shown in figure 1. They are the initialization (including three initial solution generators – random generator (RAN), nearest neighbour algorithm – type 1 (NN1), nearest neighbour algorithm – type 2 (NN2) and their combinations, local search with stochastically selecting from two different methods (i.e., 1-shift and 2-opt exchanges), selection, and edge recombination (ER) crossover. When starting to solve the PTSP (Generation 0, g = 0), initial solutions are generated based on three different nearest neighbour algorithms, which are then improved by the local search. Then, "selection" is called into place to further select solutions to be mated based on their solution quality (objective function value). Pairs of solutions are used to generate the new solutions via ER crossover. The newly generated solutions are then improved using the local search. The solutions are allowed to evolve through successive generations until a termination criterion is met. The detailed description of the embedded components is illustrated in the following sections.

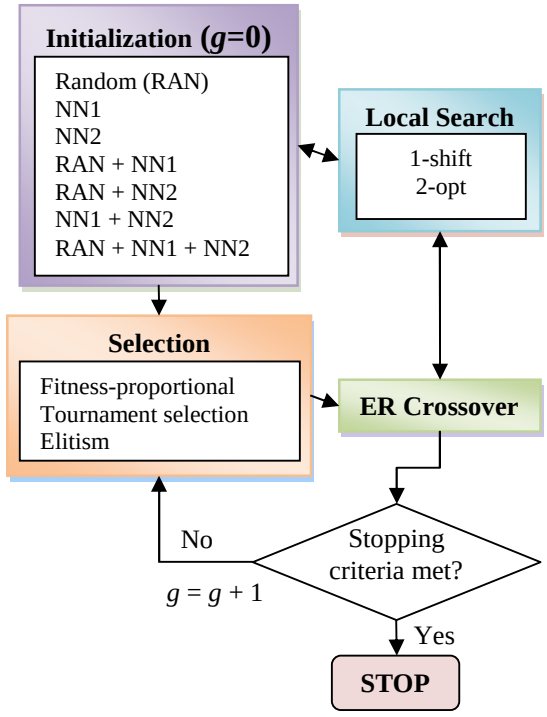

Figure 1. The solution procedure for the PTSP

3.1 Initialization

This procedure is designed to generate m initial solutions (m = 10 in this study). Three initial solution generation methods are proposed. The detailed procedures of these three methods are described as follows.

3.1.1 Random generator (RAN)

The random generator is used to randomly create a set of initial solutions. Considering a PTSP with n nodes (excluding the depot, node *0*), the node, a_{r1}, is randomly chosen from the set of unselected node (S_u) and inserted into the second location. Then, the set of unselected node (S_u) should be updated by erasing node a_{r1}. The third location can be filled by randomly choosing a node from S_u. Following the above procedure, the initial solution (tour) can be built.

3.1.2 Nearest Neighbour Algorithm – Type 1 (NN1)

Considering a PTSP with n nodes (excluding the depot, node *0*), the two nearest nodes, a_0 and b_0, from node *0* is selected first and randomly inserted into the second and the last locations, respectively. Basically, the nearest and second nearest nodes from the current node are selected by the probability of 0.9 and

0.1, respectively. This mechanism is used to generate different initial solutions. The procedure of building the initial solution is as follows. After selecting nodes a_0 and b_0, one of the top two nearest nodes from a_0 is randomly selected (a_1) based on the specific probability (i.e., 0.9 for the nearest node and 0.1 for the second nearest node) and inserted behind a_0. Similarly, the node (b_1) is selected and inserted in front of b_0. Then, among the remaining nodes, one of the top two nearest nodes from a_1 is randomly selected (a_2) based on the specific probability and inserted behind a_1, while node (b_2) is randomly selected based on the rule described previously and inserted in front of b_1. The initial solution (tour) is thus built by following the above rule and expressed as follows.

$0 \quad a_0 \quad a_1 \quad a_2 \quad \ldots \quad b_2 \quad b_1 \quad b_0 \quad 0$

3.1.3 Nearest Neighbour Algorithm – Type 2 (NN2)

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 ($\lfloor (n+1)/2 \rfloor - 4$) and ($\lfloor (n+1)/2 \rfloor + 4$). The reason why $\lfloor (n+1)/2 \rfloor - 4$ and $\lfloor (n+1)/2 \rfloor + 4$ 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 nearest neighbour algorithm, which find the unvisited node with the shortest distance, is used to build up the sequence of the tour. Basically, the nearest and second nearest nodes from the current node are selected by the probability of 0.9 and 0.1, respectively. This mechanism is used to generate different initial solutions. The procedure of building the initial solution is as follows. After selecting node a_0, one of the top two nearest nodes from a_0 is randomly selected (a_1) based on the specific probability and inserted in front of a_0. Similarly, the node (a_2) is selected and inserted behind a_0. Then, among the remaining nodes, one of the top two nearest nodes from a_1 is randomly selected (a_3) based on the specific probability and inserted in front of a_1, while node (a_4) is randomly selected based on the rule described previously and inserted behind a_2. The initial solution (tour) is thus built by following the above rule and expressed as follows.

$0 \quad \ldots \quad a_3 \quad a_1 \quad a_0 \quad a_2 \quad a_4 \quad \ldots \quad 0$

3.2 Local search

This component is used in an attempt to further enhance the solution generated via a local search procedure. As the previous studies [16, 20] showed that the diversified local search by randomly choosing two different local search methods (i.e., 1-shift and 2-opt) to improve the solution generated yielded the best results in both heterogeneous and homogeneous cases, the diversified local search strategy is used in this procedure. That is, one of these two local search methods is selected to improve the candidate solution based on the specific probability of 0.5 for these two local searches.

The procedures of 1-shift and 2-opt exchanges are briefly summarized as follows. Given an *a priori* tour τ, its 1-shift neighbourhood is the set of tours obtained by moving a node at position i to position j with the intervening nodes being accordingly shifted backwards one space. The 2-opt exchange is the set of tours obtained by reversing a section of τ.

The approximate evaluation of expected length of the *a priori* tour shown in (2) is then used to increase the computational efficiency. For a specific tour τ, $E_\lambda^{AP}[\tau]$ is always less than the value of $E[\tau]$ because of the truncation in calculating $E_\lambda^{AP}[\tau]$. Let τ_b and τ_a denote the *a priori* tour before and after a specific local search method, respectively. It means that no improvement has been found after the local search if $E_\lambda^{AP}[\tau_a] \geqq E[\tau_b]$. Equation (1) is used to exactly evaluate the solution after the local search if $E_\lambda^{AP}[\tau_a] < E[\tau_b]$. If the local search yields a better $E[\tau]$ value than the one from the original solution (i.e., $E[\tau_a] < E[\tau_b]$), the new solution (τ_a) will replace the original solution (τ_b). If no improvement has been found after the local search, no replacement will be made. The above procedure is repeated N_{LS} times for each solution ($N_{LS} = 30$ in this study).

3.3 Selection

Three selection methods are used in this study: fitness-proportionate, elitism and tournament selection. Under the fitness-proportionate selection method, the probability of selecting a particular solution for reproduction is proportional to its own fitness (i.e., $E[\tau]$) relative to the average fitness of the entire current generation. With this selection method, the best solution tends to produce the largest amount of offspring and hence survive to future generations. This procedure can be regarded as a "biased" roulette wheel where each string in the current population occupies a roulette wheel slot sized in proportion to its fitness [23]. Selection can be done by simply spinning the weighted roulette wheel, and fitter strings will have higher chances of being selected.

Tournament selection, inspired by the competition in nature among individuals for the right to mate, picks two solutions using the fitness-proportionate selection from the population and the fittest one is selected for reproduction [22, 23]. Each solution can participate in an unlimited number of tournaments. The two winning solutions in the tournament are then subjected to the crossover operators, as described in the next section.

Under the elitism selection strategy, the top m_1 strings (m_1 is determined by the analyst) of the current generation in terms of fitness value are kept and propagated to the next generation [22]. The remaining solutions in the next generation are then generated based on the tournament selection method and the crossover operators. This procedure guarantees that the best solution in the next generation is not worse than the one in the current generation.

3.4 Edge recombination (ER) crossover

The main purpose of this component is to create new solutions using a given pair of solutions chosen by "selection." Based on the results from previous studies [15, 27], the edge recombination (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.

ER crossover was proposed by Whitley et al. [28] 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 "selection"--(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 as follows. Assuming that node 0 is selected as the starting node 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 edge, node 4, is picked as the node 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 already deleted from the list). Therefore, node 3 is chosen to be the node next to node 4 in the new solution. The new solution generated is further improved by the local search.

3.5 The procedure after the first generation

The newly generated solutions from the ER crossover and local search are used to update the population in terms of the objective function value. The above procedure is repeated until a termination criterion is met. However, if there are no solutions to be updated for the population in the current generation, the initialization is used to generate (m - m_1) new solutions in the next generation, but keeping m_1 high quality solutions (m_1 = 2, in this study). In addition, if the previous three generations converge to the same best solution, the local search is used to improve that "converged" solution by repeating N_{LS2} times to exhaustively search the neighbourhood of that "converged" solution (N_{LS2} = 300, in this study).

4 NUMERICAL RESULTS AND DISCUSSIONS

Ninety instances generated by Tang and Miller-Hooks [4] with size n = 50, 75, and 100 were used as numerical experiments in this study to examine the effects of three different initial solution generators under genetic algorithm framework for solving the heterogeneous 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 $d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$.

All implementations in this study were coded in FORTRAN and run on an Intel Pentium IV 2.8 GHz CPU personal computer with 512 MB memory, running Windows XP operating system, using Compaq Visual Fortran 6.5 compiler. To compare the effectiveness among three initial solution generators, the preset maximum number of generations (g_{max}) was used as the termination criterion (g_{max} is set to be two times the number of nodes, i.e., g_{max} = 2n, in this study). In order to enhance the robustness of the results and conduct appropriate statistical analysis, the proposed methods were used to solve each problem instance 30 times. The obtained $E[\tau]$ values of ten instances were averaged (as shown in shade in Tables 1, 2, and 3 for 50-, 75-, and 100-node, respectively) and tested for the comparative solution quality of three generators by the permutation test [29]. Definitions of terms used in the column headings for Tables 1, 2, and 3 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, while "50-0.2" represents the average of all 10 instances with n = 50 and p = 0.2.

Table 1. Detailed results for problem size n = 50

	RAN	NN1	NN2	RAN+ NN1	RAN+ NN2	NN1+ NN2	RAN+ NN1+ NN2
p = 0.2							
01	**217.83**	**217.83**	**217.83**	217.83	217.83	217.83	217.83
02	205.96	205.96	**205.96**	205.96	205.96	205.96	205.96
03	212.36	**212.29**	**212.29**	212.29	212.29	212.29	212.29
04	215.70	**215.70**	**215.70**	215.70	215.70	215.70	215.70
05	249.59	**249.59**	**249.59**	249.59	249.59	249.59	249.59
06	**218.80**	**218.80**	**218.80**	218.80	218.80	218.80	218.80
07	**245.80**	**245.80**	**245.80**	245.80	245.80	245.80	245.80
08	**239.16**	**239.16**	**239.16**	239.16	239.16	239.16	239.16
09	**237.74**	237.74	237.74	237.74	237.74	237.74	237.74
10	205.45	205.45	**205.45**	205.45	205.44	205.44	205.45
Avg.	224.84	224.83	224.83	224.83	224.83	224.83	224.83
p = 0.5							
01	313.61	313.73	**313.40**	313.61	313.44	313.52	313.48
02	349.84	349.88	**349.81**	349.90	350.05	349.91	349.91
03	335.28	**334.98**	335.87	335.43	334.98	335.28	334.69
04	348.24	348.16	**348.14**	348.27	348.20	348.18	348.21
05	**327.06**	**327.06**	**327.06**	327.06	327.06	327.06	327.06
06	**373.58**	373.98	375.98	374.48	374.95	376.16	374.82
07	324.33	324.34	**324.32**	324.34	324.33	324.34	324.32
08	**337.58**	340.36	339.43	339.66	338.98	339.53	339.43
09	352.72	**352.59**	354.03	352.75	353.49	353.57	352.89
10	348.72	348.79	**348.70**	348.67	348.75	348.74	348.83
Avg.	313.61	313.73	**313.40**	313.61	313.44	313.52	313.48
p = 1.0							
01	462.57	**459.14**	466.64	461.87	463.61	463.64	465.19
02	405.35	406.11	**405.04**	405.22	405.17	406.03	407.66
03	468.83	468.68	**466.13**	469.95	467.91	467.19	467.85
04	459.96	**457.54**	460.87	458.99	460.87	459.94	459.69
05	425.03	**424.55**	428.34	424.34	425.78	426.25	425.88
06	506.82	**502.63**	503.56	505.04	506.50	505.18	506.47
07	**482.68**	485.09	482.75	484.91	485.61	483.35	485.23
08	439.99	**439.57**	441.53	439.33	440.09	440.88	439.93
09	439.55	**437.32**	440.13	437.38	439.34	438.67	439.01
10	**417.59**	417.83	417.60	417.70	417.49	417.67	417.70
Avg.	450.84	449.85	451.26	450.47	451.24	450.88	451.46

Contrary to the researcher's expectation, none of the permutation test done on the average $E[\tau]$ values obtained by the three generators are statistically significantly different, $p > 0.05$. As studies have shown the effectiveness of incorporating the concept of diversification into metaheuristic methods [20, 30]

four more methods were generated by combining any two or three of the proposed generators (i.e., RAN+NN1, RAN+NN2, NN1+NN2, RAN+NN1+NN2). Based on the permutation test done of the seven methods (i.e., RAN, NN1, NN2, RAN+NN1, RAN+NN2, NN1+NN2, RAN+NN1+NN2), again, no significant results were found.

Table 2. Detailed results for problem size n = 75

	RAN	NN1	NN2	RAN+ NN1	RAN+ NN2	NN1+ NN2	RAN+ NN1+ NN2
p = 0.2							
01	267.24	267.35	**267.22**	267.37	267.32	267.30	267.27
02	278.13	**278.13**	278.13	278.13	278.13	278.13	278.13
03	**268.25**	**268.25**	**268.25**	268.25	268.25	268.25	268.25
04	271.55	**271.52**	271.53	271.52	271.52	271.52	271.51
05	**264.67**	264.81	264.82	264.63	264.76	264.75	264.65
06	236.27	**236.05**	236.44	236.19	235.96	236.11	236.20
07	**273.18**	273.19	273.19	273.18	273.18	273.21	273.19
08	**251.45**	251.45	251.45	251.45	251.45	251.45	251.45
09	280.46	280.45	**280.42**	280.41	280.50	280.44	280.47
10	**268.31**	268.31	268.31	268.31	268.31	268.31	268.31
Avg.	265.95	265.95	265.98	265.94	265.94	265.95	265.94
p = 0.5							
01	**398.78**	402.60	401.82	401.92	401.95	403.29	402.60
02	**424.03**	431.67	430.47	428.09	428.59	427.78	428.27
03	**427.51**	427.76	428.23	427.33	427.83	428.03	428.06
04	410.10	410.28	**405.69**	410.80	407.13	406.62	408.29
05	400.87	403.56	**399.84**	401.97	401.89	401.05	400.25
06	**378.40**	379.60	379.13	379.09	378.70	379.00	379.36
07	428.46	431.76	**425.49**	428.26	426.52	428.48	427.00
08	397.67	**395.72**	399.70	395.82	396.47	397.00	396.14
09	**367.88**	367.91	367.95	367.94	367.88	367.93	367.81
10	**401.75**	405.60	402.88	406.27	402.80	406.78	404.18
Avg.	403.54	405.65	404.12	404.75	403.98	404.60	404.20
p = 1.0							
01	521.93	525.03	**516.10**	521.94	517.51	521.04	523.19
02	547.61	**532.37**	543.87	539.70	544.46	540.22	539.93
03	489.01	502.45	**486.27**	498.76	490.17	491.54	493.80
04	579.02	**574.62**	579.25	574.03	577.60	580.78	576.15
05	531.78	**530.34**	532.84	532.76	531.53	529.19	531.93
06	492.21	**489.56**	492.75	490.52	491.52	490.50	489.36
07	**517.22**	524.34	524.06	521.29	521.11	519.85	524.77
08	560.81	**551.07**	554.83	546.44	550.88	550.35	550.66
09	535.26	534.31	**534.29**	536.19	537.27	534.82	535.43
10	**514.01**	517.00	516.86	511.28	513.38	512.66	517.66
Avg.	528.89	528.11	528.11	527.29	527.54	527.10	528.29

These unexpected outcomes alerted the author to look at the results more deeply. Specifically, rather than looking at the averaged $E[\tau]$ values of all ten instances, each of the ten instances was analyzed separately to acknowledge its maybe distinct characteristics. Moreover, as the combination of different generators may obscure the effects of the three generators, *t*-tests are conducted only on the three generators for each problem instance, respectively. In Tables 1, 2, and 3, the underlined value denotes the average value yield by the specific initial solution generator is significantly different from the best average yielded value among these three generators.

As shown in Tables 1, 2, and 3, even though none of these three initial solution generators (i.e., RAN, NN1, and NN2) consistently performed best in terms of the obtained $E[\tau]$ value across all instances, it seems to be somewhat instance-dependent for the same combination of n and p. For example, for n = 75 and $p \in (0.0, 0.5]$, the value obtained by RAN (398.7818) are significantly better than the values obtained by NN1 (402.6016) and NN2 (401.8246) for instance number 01; the value obtained by NN2 (405.6888) are significantly better than the values obtained by RAN (410.0959) and NN1 (410.2795) for instance number 04; the value obtained by NN1 (395.7191) are significantly better than the values obtained by RAN (397.6664) and NN2 (399.6999) for instance number 08. 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 $E[\tau]$ values obtained by the specific generator where others have higher $E[\tau]$ values yielded by the same generators indicates that the possibility of finding the optimal solution based on the same generator may fluctuate across instances. The phenomenon among instances suggests 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) in solving the PTSP. As such, algorithms which can dynamically select the initial solution generator according to individual instance pattern might be one direction for effectively solving the PTSP. Another promising strategy would be algorithms that could adaptively evolve by learning from some search experience so as to select the best generator, rather than fixing on one solution generator at the commencement of the test, as done in most of existing PTSP studies.

Table 3. Detailed results for problem size n = 100

	RAN	NN1	NN2	RAN+ NN1	RAN+ NN2	NN1+ NN2	RAN+ NN1+ NN2
p = 0.2							
01	277.07	**277.07**	277.07	277.07	277.07	277.07	277.07
02	298.59	**298.59**	298.59	298.59	298.59	298.59	298.59
03	**309.57**	309.57	309.58	309.57	309.58	309.57	309.57
04	298.76	**298.76**	298.76	298.76	298.76	298.76	298.76
05	319.73	319.82	**319.70**	319.78	319.69	319.74	319.68
06	301.13	301.14	**301.12**	301.14	301.13	301.13	301.13
07	301.50	**301.49**	301.58	301.47	301.54	301.48	301.58
08	294.61	**293.88**	293.98	293.92	293.62	293.87	294.34
09	304.43	**304.36**	304.47	304.39	304.42	304.38	304.39
10	303.83	303.82	**303.82**	303.82	303.82	303.82	303.82
Avg.	300.92	300.85	300.87	300.85	300.82	300.84	300.89
p = 0.5							
01	**463.73**	465.00	464.96	463.86	464.03	468.90	464.74
02	464.89	**460.58**	468.93	462.08	465.52	460.11	464.67
03	**463.59**	469.47	464.37	470.16	464.33	469.35	467.78
04	445.44	**444.40**	451.64	444.66	447.70	445.12	448.98
05	**473.26**	475.68	474.57	473.02	473.45	480.85	476.05
06	**486.30**	486.39	489.04	487.77	488.30	488.03	488.92
07	**459.90**	465.83	462.18	465.18	460.32	470.36	464.58
08	454.77	454.64	**451.45**	454.68	453.88	453.61	452.96
09	**461.00**	462.88	465.46	463.33	462.68	459.34	462.96
10	**437.79**	440.19	441.30	439.36	438.94	441.06	440.28
Avg.	461.07	462.51	463.39	462.41	461.92	463.67	463.19
p = 1.0							
01	619.64	**612.96**	619.43	616.14	621.66	619.02	616.53
02	638.51	637.55	**632.21**	635.54	635.15	642.47	634.86
03	633.07	**625.04**	634.56	630.48	633.94	626.35	629.48
04	**639.12**	640.96	639.25	636.12	636.49	644.43	638.07
05	**586.23**	590.03	588.71	591.46	593.25	596.98	588.47
06	**661.64**	662.68	662.15	662.97	660.17	663.31	662.65
07	581.35	**578.14**	583.21	574.94	580.72	588.54	579.69
08	645.27	637.55	**632.14**	636.66	637.20	636.00	636.46
09	641.42	**636.91**	644.47	634.08	634.68	641.90	640.12
10	614.06	**613.86**	618.24	616.75	620.19	618.89	618.72
Avg.	626.03	623.57	625.44	623.51	625.35	627.79	624.51

Another related suggestion from this study is with regard to the inadequacy of past studies that relied only on a single-instance numerical experiment. Explicitly, one PTSP test instance for a specific combination of n and p, as used in previous studies [14], assumed that the same combination of n and p will yield similar results by the same method. As could be

clearly seen from the results of this study, the assumption is not upheld. The numerical experiment based on one PTSP test instance may be subject to biased sampling of the PTSP instances, which may inadvertently result in over- or under-estimation of the performance of the proposed algorithms for PTSP. As such, to obtain more valid 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.

5 CONCLUSIONS & FUTURE WORK

In this paper, three different initial solution generators under GA framework are first proposed and developed to solve the PTSP. Though the numerical results did not support the predominate superiority of one specific generator, nor did incorporating the notion of diversification enhances any of its performance, it, however, enlightened the researcher to be aware of the possibility of context-dependent phenomenon. Particularly, further data analysis and testing showed that the $E[\tau]$ values obtained by the three generators seem to be instance-dependent for the same combination of n and p. Based on the findings, it is suggested that algorithms that embed the notion of adaptively selecting the initial solution generator based on some experiences in searching ability, or that can take into account of patterns of individual instance could be promising strategies to solve the PTSP. Finally, to obtain more valid 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.

ACKNOWLEDGEMENTS

This work was supported primarily by the National Science Council of Taiwan under Grant and NSC 97-2410-H-260-012.

REFERENCES

[1] D. Bertsimas, *Probabilistic Combinatorial Optimization Problems*, Ph.D. Dissertation, Massachusetts Institute of Technology, MA, U.S.A. (1988).

[2] D. Bertsimas, P. Jaillet, and A. R. Odoni. A Priori Optimization. *Operations Research*, 38:1019-1033, (1990).

[3] P. Jaillet, Probabilistic Traveling Salesman Problems, Ph.D. dissertation, Massachusetts Institute of Technology, MA, U.S.A. (1985).

[4] J. J. Bartholdi, L. K. Platzman, R. L. Collins, and W. H. Warden. A Minimal Technology Routing System for Meals on Wheels. *Interfaces*, 13:1-8, (1983).

[5] D. Bertsimas, P. Chervi, and M. Peterson. Computational Approaches to Stochastic Vehicle Routing Problems. *Transportation Science*, 29:342-352, (1995).

[6] A. M. Campbell. Aggregation for the Probabilistic Traveling Salesman Problem. *Computers & Operations Research*, 33:2703-2724, (2006).

[7] P. Jaillet. A priori Solution of a Traveling Salesman Problem in which a Random Subset of the Customers are Visited. *Operations Research*, 36:929-936, (1988).

[8] H. Tang and E. Miller-Hooks. Approximate Procedures for the Probabilistic Traveling Salesperson Problem. *Transportation Research Record*, 1882:27-36, (2004).

[9] J. J. Bartholdi and L. K. Platzman. Heuristics Based on Spacefilling Curves for Combinatorial Problems in Euclidean Space. *Management Science*, 34:291-305, (1988).

[10] D. Bertsimas and L. Howell. Further Results on the Probabilistic Traveling Salesman Problem. *European Journal of Operational Research*, 65:68-95, (1993).

[11] P. Jaillet. Stochastic Routing Problems. In *Advanced School on Stochastics in Combinatorial Optimization*. G. Andreatta, F. Mason, P. Serafini (Eds.). World Scientific Publisher, pp. 192-213 (1987).

[12] F. Rossi and I. Gavioli. Aspects of Heuristic Method in the Probabilistic Traveling Salesman Problem. In *Advanced School on Stochastics in Combinatorial Optimization*. G. Andreatta, F. Mason, P. Serafini (Eds.). World Scientific Publisher, pp. 214-227 (1987).

[13] L. Bianchi, Ant Colony Optimization and Local Search for the Probabilistic Traveling Salesman Problem: A Case Study in Stochastic Combinatorial Optimization, Ph.D. Dissertation, Universite Libre de Bruxelles, Brussles, Belgium. (2006).

[14] J. Branke and M. Guntsch. Solving the Probabilistic TSP with Ant Colony Optimization. *Journal of Mathematical Modelling and Algorithms*, 3:403-425, (2004).

[15] Y.-H. Liu, R.-C. Jou, C.-C. Wang, and C.-S. Chiu. An Evolutionary Algorithm with Diversified Crossover Operator for the Heterogeneous Probabilistic TSP. Lecture Notes in Artificial Intelligence, 4617:351-360, (2007).

[16] Y.-H. Liu. A Memetic Algorithm for the Probabilistic Traveling Salesman Problem. In: *Procs. 2008 IEEE Congress on Evolutionary Computation (CEC 2008)*, Hong Kong. (2008).

[17] N. E. Bowler, T. M. A. Fink, and R. C. Ball. Characterization of the Probabilistic Traveling Salesman Problem. *Physical Review E*, 68:036703, (2003).

[18] Y.-H. Liu. A Scatter Search Based Approach with Approximation Evaluation for the Heterogeneous Probabilistic Traveling Salesman Problem. In *Procs. 2006 IEEE Congress on Evolutionary Computation (CEC 2006)*, Vancouver, Canada. (2006).

[19] Y.-H. Liu. A Hybrid Scatter Search for the Probabilistic Traveling Salesman Problem. *Computers & Operations Research*, 34:2949-2963, (2007).

[20] Y.-H. Liu. Diversified Local Search Strategy under Scatter Search Framework for the Probabilistic Traveling Salesman Problem. *European Journal of Operational Research*, 191:332-346, (2008).

[21] T. Bäck, D. B. Fogel and Z. Michalewicz, *Handbook of Evolutionary Computation*, Oxford University Press, U.K., (1997).

[22] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, U.S.A., (1991).

[23] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison Wesley, Reading, PA, U.S.A., (1989).

[24] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, Boston, MA, U.S.A., (1992).

[25] Y.-H. Liu and H. S. Mahmassani. Global Maximum Likelihood Estimation Procedure for Multinomial Probit Model Parameters. *Transportation Research B*, 34B:419-449, (2000).

[26] P. Jaillet and A. R. Odoni. The Probabilistic Vehicle Routing Problem. In *Vehicle Routing: Methods and Studies*. B. L. Golden, A. A. Assad (Eds.). North-Holland, pp. 293-318, (1988).

[27] J.-Y. Potvin. Genetic Algorithms for the Traveling Salesman Problem. *Annals of Operations Research*, 63:339-370, (1996).

[28] D. Whitley, T. Starkweather, and D. Fuquay. Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator. In: *Procs. 3rd Int'l Conf. on Genetic Algorithm* (ICGA '89), Fairfax, Virginia, USA, Morgan Kaufmann. (1989).

[29] D. Basso, M. Chiarandini, and L. Salmaso. Synchronized Permutation Tests in Replicated I × J Designs. *Journal of Statistical Planning and Inference*, 137:2564-2578, (2007).

[30] B. Gendron, J. Y. Potvin, and P. Soriano. Diversification Strategies in Local Search for a Nonbifurcated Network Loading Problem. *European Journal of Operational Research*, 142:231-241, (2002).