THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Doctor of Engineering (Industrial Engineering)
DEGREE

Industrial Engineering
FIELD

Industrial Engineering
DEPARTMENT

TITLE: The Prediction Approach for the Traveling Salesman Problem

NAME: Mr. Wanatchapong Kongkaew

THIS THESIS HAS BEEN ACCEPTED BY

THESIS ADVISOR
(Assistant Professor Juta Pichitlamken, Ph.D.)

THESIS CO-ADVISOR
(Associate Professor Peerayuth Charnsethikul, Ph.D.)

THESIS CO-ADVISOR
(Mr. Worawut Wangwatcharakul, Ph.D.)

DEPARTMENT HEAD
(Associate Professor Anan Mungwattana, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON _____

DEAN
(Associate Professor Gunjana Theeragool, D.Agr.)

Copyright by Kasetsart University All rights reserved

THESIS

THE PREDICTION APPROACH FOR THE TRAVELING SALESMAN PROBLEM

WANATCHAPONG KONGKAEW

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Engineering (Industrial Engineering)
Graduate School, Kasetsart University
2013

Copyright by Kasetsart University All rights reserved

Wanatchapong Kongkaew 2013: The Prediction Approach for the Traveling Salesman Problem. Doctor of Engineering (Industrial Engineering), Major Field: Industrial Engineering, Department of Industrial Engineering. Thesis Advisor: Assistant Professor Juta Pichitlamken, Ph.D. 171 pages.

This thesis aims to introduce the probabilistic theory and its approach for estimating the optimal tour of the symmetric traveling salesman problem (TSP). This approach formulates TSP tours and their corresponding lengths as a Gaussian process regression (GPR) model, then the estimated optimal tour can be predicted. In addition, the first sample tour is constructed by the nearest neighbor algorithm, and other sample tours are created by using the 2-exchange neighborhood strategy.

In order to improve solutions, two methods based on the iterated local search (i.e., the iterated 2-opt method and the iterated simulated annealing algorithm) are incorporated with GPR (called GPRTSP-0 and GPRTSP-1 algorithms, respectively). The GPRTSP-1 algorithm is shown to converge in the limit.

For the search performance on the symmetric TSP, two proposed algorithms are compared with the genetic algorithm (GA) and the simulated annealing (SA) algorithm. As a result, the GPRTSP-1 algorithm slightly outperforms GPRTSP-0 algorithm in term of solution quality while the GPRTSP-1 algorithm spends more running time than the GPRTSP-0 algorithm. In addition, both proposed algorithms achieve a reasonable trade-off between solution quality and running time, and they significantly outperforms GA and SA. Moreover, the GPRTSP-1 algorithm (executing under fixed runtime condition) is compared with the other six recent algorithms in term of solution quality. As a result, on the overall, the GPRTSP-1 algorithm yields worse solution quality. However, for some test problems, the GPRTSP-1 algorithm outperforms them. Furthermore, the GPRTSP-1 algorithm guarantees that it converges in probability to optimality.

_____ _____ ___ / ___ / ___

Student's signature Thesis Advisor's signature

Copyright by Kasetsart University All rights reserved

ACKNOWLEDGMENTS

I would like to gratefully thank and greatly indebted to Asst. Prof. Dr. Juta Pichitlamken my thesis advisor for her contributions, encouragements, advices, and valuable suggestions throughout this thesis and my education at Kasetsart University. I would sincerely like to thank Assoc. Prof. Dr. Peerayuth Charnsethikul and Dr. Worawut Wangwatcharakul my committees, and also Dr. Nantachai Kantanantha an examination chairperson and Asst. Prof. Dr. Waressara Weerawat an external examiner from Graduate School for their valuable comments and suggestions.

I would also like to sincerely thank Assoc. Prof. Dr. Rapeepan Pitakaso from Department of Industrial Engineering, Faculty of Engineering at Ubon Ratchathani University for many insightful suggestions and his generous assistance for reviewing in some part of the research work.

I gratefully acknowledge to the financial support, the Ph.D. 50% scholarship from Prince of Songkla University, Thailand. This thesis is partially funded by the Center of Graduate Study and Research, Faculty of Engineering, Kasetsart University.

The doctoral students and the staff of the Department of Industrial Engineering, Faculty of Engineering at Kasetsart University have always been very friendly and supportive.

I am heartfelt thank to my lovely girlfriend and my friends for their help and encouragement. Finally, my most important acknowledgments are towards my parents, my sister, and my brothers for their continuing encouragements until my graduate study completion.

Wanatchapong Kongkaew

April 2013

Copyright by Kasetsart University All rights reserved

TABLE OF CONTENTS

Page

TABLE OF CONTENTS	i
LIST OF TABLES	ii
LIST OF FIGURES	iii
INTRODUCTION	1
OBJECTIVES	5
LITERATURE REVIEW	6
MATERIALS AND METHODS	35
Materials	35
Methods	35
RESULTS AND DISCUSSION	73
CONCLUSION AND RECOMMENDATIONS	106
Conclusion	106
Recommendations	107
LITERATURE CITED	109
APPENDICES	126
Appendix A The related Theorems, Lemma, and Corollary	127
Appendix B The search performance of all algorithms	132
CURRICULUM VITAE	171

Copyright by Kasetsart University All rights reserved

LIST OF TABLES

Table		Page
1	Description of other six recent algorithms	70
2	Parameter settings of all algorithms	71
3	The search performance of GPRTSP-0 algorithm	74
4	The search performance of GPRTSP-1 algorithm	84
5	Comparison of the search performance of all four algorithms	91
6	Comparison of solution quality of GPRTSP-1 with GA, SA, and other six recent algorithms	103

Appendix Table

B1	The solutions of GPRTSP-0 algorithm	133
B2	The solutions of GPRTSP-1 algorithm	137
B3	The solutions of genetic algorithm	141
B4	The solutions of simulated annealing algorithm	145
B5	The running time of GPRTSP-0 algorithm	149
B6	The running time of GPRTSP-1 algorithm	153
B7	The running time of genetic algorithm	157
B8	The running time of simulated annealing algorithm	161
B9	The solutions of GPRTSP-1 algorithm with running time limitation	165
B10	The solutions of genetic algorithm with running time limitation	167
B11	The solutions of simulated annealing algorithm with running time limitation	169

Copyright by Kasetsart University All rights reserved

LIST OF FIGURES

Figure		Page
1	Pictorial framework of 2-opt move	9
2	General algorithmic outline of the iterated local search (ILS)	20
3	Pictorial representation of the iterated local search	21
4	Graphical model for Gaussian process regression	41
5	Pictorial framework of the 2-opt algorithm	48
6	Behavior plot of ulysses16 test problem	79
7	Behavior plot of gr21 test problem	80
8	Behavior plot of pr107 test problem	81
9	Behavior plot of si535 test problem	81
10	Comparisons of solution deviations between the two proposed algorithms	96
11	Comparisons of solution deviations among four algorithms	98
12	Comparison of average running time among four algorithms	101

Copyright by Kasetsart University All rights reserved

THE PREDICTION APPROACH FOR THE TRAVELING SALESMAN PROBLEM

INTRODUCTION

Since the advent of the industrial revolution and computational technology evolution, real-world problems have been increasing in size and complexity. Many mathematicians have been continually making their effort to determine an approach to solve complex problems, one of which is the combinatorial optimization problems, such as the traveling salesman problem, the vehicle routing problem, the bin packing problem, the cutting stock problem, and the knapsack problem. However, these problems become more and more difficult to solve for an optimal solution directly under a restricted capacity of computational technology. This research considers only the traveling salesman problem.

The traveling salesman problem (TSP) is one of the well-known NP-hard problems in the field of the combinatorial optimization. The objective of the TSP is to find the shortest Hamiltonian cycle among n_c cities, where the salesman visits each of the n_c cities exactly once and then returns to the starting city. There are several traditional methods proposed to solve for an optimal solution directly. The exact algorithm would be to try all possible permutations (order combinations), but it uses brute force and takes more time to find the optimal solution (Applegate *et al.*, 2006). Originally, this problem is solved by a method based on an integer linear programming (ILP), i.e., the cutting plane methods (Dantzig *et al.*, 1954), the branch and bound method (Eastman, 1958; Little *et al.*, 1963). These methods can solve efficiently and effectively an optimal solution; however, it lacks effectiveness when the problem becomes large-scale. It is necessary to use much capacity of computational technology and spend much solving time.

Later, mathematicians propose many approximate approaches for solving a near-optimal solution. Various heuristic and approximation algorithms have been

Copyright by Kasetsart University All rights reserved

presented, for example, the nearest neighbor algorithm (Bellmore and Nemhauser, 1968), Lin-Kernighan heuristics (Lin and Kernighan, 1973; Karapetyan and Gutin, 2011), the *k*-opt heuristic (Helsgaun, 2009). In addition, some randomized heuristics can find a route extremely close to the optimal route, e.g., the genetic algorithms (Moon *et al.*, 2002; Nagata and Soler, 2012) and the simulated annealing algorithm (Geng *et al.*, 2011).

In recent years, metaheuristics are widely used for solving the TSP and its applications, such as variable neighborhood search (Hansen *et al.*, 2008; Mladenović *et al.*, 2012), particle swarm optimization (Shi *et al.*, 2007; Liao *et al.*, 2012), bee optimization (Marinakis *et al.*, 2011), the iterated local search (Lourenço *et al.*, 2010), and a combined procedure of several metaheuristics (Chen and Chien, 2011). These methods give results within a reasonable time under a restricted capacity of computational technology. Consequently, the approximate approaches become a challenging and interesting method for solving the TSP. Some methods in a stochastic process are also used for solving the TSP, for example, the Markov decision processes (Eshragh *et al.*, 2009). Furthermore, a review of the previously proposed methods is provided in the literature review chapter.

The TSP applications can be found in many fields. For the job sequencing and scheduling problem, Bagchi *et al.* (2006) reviews the variants of the flowshop scheduling that are based on some TSP approach, and they provide the review of real-world applications of these variants. In addition, the industrial application of the scheduling problem based on TSP approach for large-scale zero-wait batch processes with setup times is presented by Dong and Wang (2012). Liang *et al.* (2013) formulate the drilling operation scheduling for printed circuit board (PCB) industries in Taiwan as the variations of the scheduling based TSP approach, and then solve this problem by metaheuristics. Other TSP applications in the job sequencing and scheduling problem can be seen in Grötschel *et al.* (1991), Duman and Or (2004), and Chernykh *et al.* (2013).

Copyright by Kasetsart University All rights reserved

The most widely-used applications of TSP are found in the field of logistics and supply chain management, such as Ruland and Rodin (1997), Baldacci *et al.* (2003), Hernández-Pérez and Salazar-González (2004), and Felipe *et al.* (2012). In the latest year, Schmid *et al.* (2013) provide a basic model of a variant of TSP (i.e., the classical vehicle routing problem) for extending in the context of supply chain with respect to lot sizing, scheduling, packing, batching, inventory, and intermodality. Moreover, the TSP is addressed to solve the order-picking problem in a rectangular warehouse by Ratliff and Rosenthal (1983). Later, Theys *et al.* (2010) apply the TSP heuristics to solve the sequencing and routing problem of order pickers in conventional multi-parallel-aisle warehouse systems. Furthermore, TSP is applied in the analysis of the biological evolution that relates to the genome rearrangement, e.g., Cox *et al.* (1990) and Sankoff and Blanchette (1997).

In this research, two new methods are proposed to solve the deterministic (symmetric) TSP which they focus on a probabilistic approach, namely the Gaussian process regression model. This method becomes popular for computer experiments because of their practical and theoretical simplicity and excellent generalization ability (Rasmussen and Williams, 2006).

Gaussian process regression (GPR) method proceeds by parameterizing a covariance function, and then infers the parameters given the training data which is a training set of input-output pairs (Rasmussen and Williams, 2006). Assume that the n observations of training inputs in a matrix $\boldsymbol{X}$ are given, which reside in an input space. The ith training input $\mathbf{x}_i$ is associated with a corresponding target value y_i (the ith training output), which it is assumed to have risen from some unknown function of the training input, but it may be also corrupted by unknown noise ε_i. For the prediction of an optimal solution, these unknown functions are minimized and used to predict mean and variance of function values (correspond to test inputs). Then the predictive mean of function becomes the estimated optimal solution. Moreover, this method is described (in detail) in the materials and methods chapter. Furthermore, GPR can be applied in several fields of research, such as the machine

Copyright by Kasetsart University All rights reserved

learning (Platt *et al.*, 2002; Sinz *et al.*, 2004; Ko *et al.*, 2007), the quality improvement in manufacturing processes (Yuan *et al.*, 2008; Chen and Ren, 2009; Pal and Deswal, 2010), and the traffic problem (Idé and Kato, 2009). Moreover, the GPR applications are described in the literature review chapter.

In this research, assuming the sample tours are given as the training inputs. The total distance of each sample tour (training output) is calculated, and then the sample tours and their total distances are used for predicting the optimal traveling tour and its total distance using the aforementioned approach. In addition, some improvement methods are applied for improving the estimated solution.

This thesis is organized as follows. First, the research objective is given in the objectives chapter. The literature surveys, including the previously proposed approaches for solving the TSP, the GPR approach and its applications, the iterated local search method, and the convergence analysis of some algorithms on optimality, are provided in the literature review chapter. Next, the materials and methods chapter explains our proposed GPR algorithms for solving the symmetric TSP, and the theoretical convergence analysis is provided. In addition, the experimental settings and the briefly description of the benchmarking methods are included in the last section of this chapter. Later, the results and discussion chapter provides the numerical results of our proposed GPR algorithms and the comparisons of these algorithms with some benchmarking approaches, in terms of solution quality and computation time. Finally, the overall result of this work is concluded in the conclusion and recommendations. Moreover, the recommendations are provided for extending in the further research.

Copyright by Kasetsart University. All rights reserved

OBJECTIVES

The objectives of this research are as follow.

1. To propose theory and methods for solving the symmetric (deterministic) traveling salesman problem by using Gaussian process regression.

2. To propose the metaheuristics for solving the symmetric (deterministic) traveling salesman problem.

3. To analyze the theoretical properties of the proposed algorithms for optimality.

4. For the search performance on the symmetric (deterministic) traveling salesman problem, the efficiency (i.e., the computation time) and the effectiveness (i.e., the quality of solutions) of the proposed algorithms are compared with two well-known methods (i.e., the genetic algorithm and the simulated annealing algorithm) using MATLAB program. Moreover, the effectiveness (i.e., the quality of solutions) of the provably proposed metaheuristic is compared with other six recent algorithms.

Copyright by Kasetsart University All rights reserved

LITERATURE REVIEW

This chapter reviews the literature survey for the problem at hand. Firstly, the literature surveys of the traveling salesman problem, including its characteristic and the previous methods for solving its solution, are provided in Section 1. Secondly, Section 2 explains the conceptual framework of the iterated local search approach, and some related literatures are illustrated. Thirdly, the conceptual framework of the Gaussian process regression is discussed, and the literature surveys on its applications are exemplified in Section 3. Finally, the approaches for analyzing the convergence on optimality of the recent algorithms are discussed in Section 4.

1. Traveling Salesman Problem

1.1 Characteristic of the traveling salesman problem

The objective of the TSP is to find the shortest Hamiltonian cycle among n_c cities, where the salesman visits each of the n_c cities exactly once and then returns to the starting city. There are many formulations for the TSP which are actively studied. In addition, the TSP formulations can be modified to correspond to the studied problems (with their variations). However, this research restricts to the most common one, and it is formulated as a classical TSP. This formulation is explained in the context of integer linear programming (ILP), which can be best understood (Dantzig *et al.*, 1954; Laporte, 1992), for the purpose of this thesis as shown below:

Copyright by Kasetsart University All rights reserved

$$\text{Minimize} \quad Z = \sum_{i \neq j} d_{ij} x_{ij}$$

$$\text{Subject to} \quad \sum_{j=1}^{n_c} x_{ij} = 1, \quad i = 1,2,...,n_c,$$

$$\sum_{i=1}^{n_c} x_{ij} = 1, \quad j = 1,2,...,n_c,$$

$$\sum_{i,j \in \mathcal{S}_v} x_{ij} \leq |\mathcal{S}_v| - 1,$$

$$\mathcal{S}_v \subset \mathcal{V},\ 2 \leq |\mathcal{S}_v| \leq n_c - 2,$$

$$x_{ij} \in \{0,1\},\ i, j = 1,2,...,n_c,$$

where Z is the total distance of traveling tour,

d_{ij} is the traveling distance from city i to city j,

x_{ij} is 1 if the path consists of the traveling arc from city i to city j, and 0 otherwise,

n_c is the number of cities,

$\mathcal{S}_v$ is a subset of vertices ($\mathcal{V}$), which contain in the subtour,

$|\mathcal{S}_v|$ is the number of arcs, which contain in the subtour.

This formulation can be solved directly by hand when the number of cities (size) is small; however, it is hard to solve for a solution when the size becomes bigger. The next subsection provides the literature review on either direct methods or indirect (approximate) approaches that are proposed for solving TSP.

1.2 Methods for solving TSP

1.2.1 Exact algorithms

There are several algorithms designed to solve the TSP (Applegate *et al.*, 2006). The brute-force exact algorithm would be to try all possible permutations (order combinations), but it takes more time to find the optimal solution than the branch and bound method (Eastman, 1958; Little *et al.*, 1963) or the cutting

Copyright by Kasetsart University All rights reserved

plane method (Dantzig *et al.*, 1954). These methods can solve efficiently and effectively an optimal solution; however, it lacks effectiveness when the problem becomes large-scale. It is necessary to use much capacity of computational technology and to spend much solving time.

1.2.2 Approximation approaches

Mathematicians propose many approximation approaches (or heuristics) for solving a near-optimal solution. One of traditionally approximation methods is the nearest neighbor algorithm or the so-called greedy algorithm, which is used to construct a good initial tour (Bellmore and Nemhauser, 1968). In addition, the Lin-Kernighan heuristic and the 2-opt algorithm are still implemented for solving the TSP in some recent literatures, as follow:

a) Lin-Kernighan heuristic

Lin and Kernighan (1973) propose an effective Lin-Kernighan heuristic to find the (near) optimal solution for the symmetric TSP, and it produces the (near) optimal solution on a given test-problem, up to 110 cities. In addition, Helsgaun (2000) improves the design and implementation of Lin-Kernighan heuristic (LKH). The experiments show that the modified version is highly effective, and it obtains the optimal solutions among test problems, up to 13,509 cities. Later, he presents an effective implementation of local search with *k*-exchange neighborhoods (*k*-opt) in his modified LKH version, and it can quickly find better tours for extremely large problems on Euclidean distances ranging from 10,000 to 10,000,000 cities (Helsgaun, 2009). Furthermore, Karapetyan and Gutin (2011) adapt the Lin-Kernighan heuristic to solve the generalized traveling salesman problem (GTSP). In the experiments, their adaptations achieve trade-off between solution quality and running time.

Copyright by Kasetsart University All rights reserved

b) 2-opt algorithm

The well-known original version of the *k*-exchange neighborhoods is the Lin-Kernighan heuristic, and it allows decomposing the sequence into 2- or 3-opt move (Lin and Kernighan, 1973; Helsgaun, 2009). The 2-opt algorithm is the well-known simple local search, and it is a special case of the *k*-exchange neighborhoods search method (Helsgaun, 2000). This algorithm executes on edges recombination, and it moves with a neighborhood search by starting at the first node in a given tour. It removes two edges, breaking the tour into two paths, and then reconnects those paths with other two possible edges. If the new tour is shorter, it becomes a current tour. This process is repeated with the next consecutive node until all nodes are selected (Laporte, 1992; Johnson and McGeoch, 2003).

Moreover, the 2-opt algorithm is embedded in other approaches as the improvement procedure, such as the parallel 2-opt algorithm by Verhoeven *et al.* (1995), some heuristics as appeared in Johnson and McGeoch (2003) and Rego *et al.* (2011), and the generalized chromosome genetic algorithm by Yang *et al.* (2008). The pictorial concept of 2-opt move is shown in Figure 1.



Figure 1 Pictorial framework of 2-opt move

Source: Johnson and McGeoch (2003)

Copyright by Kasetsart University All rights reserved

1.2.3 Heuristics based on randomized strategy

The heuristics based on randomized strategy can find TSP tours extremely close to the optimal solutions, such as the genetic algorithm (Chatterjee *et al.*, 1996; Freisleben and Merz, 1996; Potvin, 1996; Moon *et al.*, 2002; Choi *et al.*, 2003; Yang *et al.*, 2008; Nagata and Soler, 2012), the randomized arbitrary insertion algorithm (Brest and Žerovnik, 2005), the cross entropy method (De Boer *et al.*, 2005; Rubinstein, 2006), and the simulated annealing algorithm (Lo and Hsu, 1998; Tian *et al.*, 1999; Herault, 2000; Jeong *et al.*, 2009; Liu *et al.*, 2009; Geng *et al.*, 2011). In this section, some literatures based on the genetic algorithm and the simulated annealing algorithm are discussed in more detail, as follow:

a) Genetic algorithm

The genetic algorithm (GA) is a well-known search heuristic for solving optimization problems using technique inspired by natural evolution (Chatterjee *et al.*, 1996). It involves three basic steps: evaluation, crossover, and mutation. In the first step (evaluation), a population of individual chromosomes is reproduced, and better chromosomes (based on their objective function or "fitness" value) are selected for the next generation of population with some probability. The crossover step randomly selects pairs of survival chromosomes to the next generation, and mates them for producing new chromosomes. The mutation step randomly chooses a chromosome completed by crossover, and mutates it at a particular point for a new population. All steps are repeated until a solution deemed optimal is found or some stopping criterion is reached.

In order to execute on GA operators, many coding and representations are presented for crossover and mutation operators, e.g., binary representation, ordinary representation, and path representation. Larrañaga *et al.* (1999) provide a review of these different GA representations and operators for the traveling salesman problem, and show the empirical comparisons on performance of these operators with some well-known TSP problem. Moreover, many variations of

Copyright by Kasetsart University All rights reserved

GA applied for the TSP are proposed throughout three decades. The GA methods presented in the early 1996 can be seen in Potvin (1996); therefore, this research provides some literatures of GA variations proposed from 1996 to the present, as follow:

Chatterjee *et al.* (1996) develop the genetic algorithm with an asexual reproduction plan through a generalized mutation for an evolutionary operator, and apply their proposed method for solving a near-optimal solution of the TSP. In their proposed GA algorithm, they construct the initial populations (tours) by the nearest neighbor heuristic, and use the asexual reproduction scheme with randomly *k*-cut operation based on the geometric distribution to produce new tours. As a result, their proposed GA algorithm achieves good solution quality within approximately 3.2% of the deviation from the optimal solutions.

In addition, Freisleben and Merz (1996) present the combined approach between local search and genetic algorithm, where the local search is used to find the local optima in the search space, and the genetic algorithm is used for searching the space of local optima with respect to the global optimum. In their proposed GA algorithm, the new crossover operator based on the globally convex character allows jumping within the space of local optima until the optimal solution is found near the center of convex, and then the new mutation operator performs random jumps within neighborhood of a local optimum. The results show that this method can find high quality solutions within reasonable running time.

Choi *et al.* (2003) present a genetic algorithm with a mixed region search for solving the asymmetric TSP. It improves the quality of solutions by extending the search space through both feasible and infeasible (defined by the solutions with subtour) regions and generating solutions from these regions into the population. As a result, this method produces high quality solutions on benchmarking problems.

Copyright by Kasetsart University All rights reserved

Yang *et al.* (2008) adapt the generalized chromosome genetic algorithm (GCGA), that is proposed by Wu *et al.* (2004), to solve the traveling salesman problem. In their paper, they provide the numerical experiments of the proposed GCGA algorithms (both original GCGA and GCGA with 2-opt local search). In addition, the performances of their algorithms perform well in terms of both solution quality and running time, and they are compared with the classical GA. The results show that the GCGA with the 2-opt local search is the most efficient method, in comparison to the original GCGA and the classical GA.

Moreover, the genetic algorithms using edge assembly crossover (EAX) operator for solving the asymmetric TSP is proposed by Nagata and Soler (2012). The EAX operator generates offspring by combining edges or arcs from two parents (*A* and *B*), and then the combination of edges is partitioned into several *AB*-cycles. Next, the *AB*-cycles are selected to construct the new cycles with a given rule. Then the intermediate solutions are generated from them by adding short edges or arcs. If the intermediate solution contains subtours, they are reconnected so that the intermediate solution becomes a tour with minimum length. As a result, their GA methods achieve the optimal solutions correspond to the set of the benchmarking instances.

b) Simulated annealing algorithm

The simulated annealing (SA) algorithm is a probabilistic method developed based on the process of material annealing in metallurgy (Laporte, 1992). For solving TSP, SA starts from an initial tour (at high temperature), and temperature is gradually decreased to achieve a minimum tour length. It generates a new tour by randomly swapping two cities in a current tour, and calculates the difference in tour length between the current tour and a new tour as ΔE. If the new tour is shorter than the current tour, it becomes current tour; otherwise, the new tour is accepted with a probability given by:

Copyright by Kasetsart University All rights reserved

$$\exp\left(-\frac{\Delta E}{\beta T_k}\right), \tag{1}$$

where T_k is the current temperature, and β is a cooling rate. These processes are repeated until some stopping criterion is reached. The variations of SA applied for the TSP are explained in the following literatures.

Tian *et al.* (1999) apply the simulated annealing (SA) algorithm on six-generation schemes for randomly generating the neighboring solution from the current solution to three combinatorial optimization problems (with permutation property). These generation schemes are: two adjacent positions (in a tour) are exchanged, two random positions are exchanged, a single position is moved, a subsequence of cycle is moved, a subsequence of cycle is reversed, and a subsequence of cycle is reversed and/or moved. Their experiments discuss the performance of algorithms with perturbation schemes on the percentage of ratio between the number of runs reached to the optimal solution and the total number of runs, and the percentage deviation of the solutions from the optimal solution. As a result, for the TSP, the perturbation scheme with reversing and/or moving a subsequence of cycle in a tour is the best generation mechanism.

Liu *et al.* (2009) propose the hybrid search approach of tabu search and simulated annealing algorithm based on adaptive cooling schedule for the TSP. In their method, the tabu list of recently visited solutions is used to check the solution before accepting it in the acceptance step, in the case of turning back to the solutions visited in the previous steps. In addition, two perturbation methods are applied to generate the neighboring solutions: First, the perturbation scheme with reversing and/or moving a subsequence of cycle in a tour presented in Tian *et al.* (1999) is used to generate the neighboring solution at the beginning of the search process. Second, the interchange between two adjacent positions presented in Tian *et al.* (1999) is applied when the search process falls in the region of a near-optimal

Copyright by Kasetsart University All rights reserved

solution. As a result, this algorithm achieves high solution quality within reasonable running time, and it has better convergence rate than the traditional SA.

An adaptive simulated annealing algorithm with greedy search (ASA-GS) for the TSP is presented by Geng *et al*. (2011). This algorithm adopts the combination of three effective mutation strategies, i.e., vertex insert mutation, block insert mutation, and block reverse mutation, to generate the candidate solution. Then the greedy search technique is applied for speeding up the convergence rate of the ASA-GS algorithm, which is embedded into the acceptance step. This algorithm implements the two-stage adaptive local search strategy for the cooling schedule. In the first stage, the current temperature is decreased by the first cooling rate until it reaches to a given (firstly) ending temperature. Then the second stage is similarly executed with the initial temperature, the second cooling rate, and the secondly ending temperature. This ASA-GS algorithm performs well on the set of the benchmarking problems, and it achieves trade-off between solution quality and running time, in comparison to other nine recent algorithms.

1.2.4 Neural network approaches

Over the past three decades, an enormous amount of literature proposes the design of algorithms based on the neural networks that use a variety of learning principles (Aras *et al.*, 1999). In this approach, the nearest neuron is mapped to the data point (city) that is presented to the neural network, and then this neuron and its related data points are initiated into a set of neurons. Next, the rest of neurons (outside set of neurons) are moved toward the data point into the target location. These processes are repeated until all data points are in the target location. Cochrane and Beasley (2003) provide the review of other algorithms based on neural network approach for solving TSP that are published during between 1985 and 2000; in addition, some algorithms that are proposed since the past decade can be seen in Saadatmand-Tarzjan *et al.* (2007). However, this thesis provides the review of some algorithms that are widely used for the performance comparison of the proposed approach for solving TSP, as follow:

Copyright by Kasetsart University All rights reserved

Aras *et al.* (1999) present a self-organizing neural network based on Kohonen's self-organizing map for solving TSP, namely Kohonen network incorporating explicit statistics (KNIES-TSP). In every iteration, the KNIES-TSP consists of two modules: the attracting module and the dispersing module. The first one maps a closet neuron to the city that is presented to the neural network, then that neuron and its neighbors are activated into a subset of the neurons. These neurons are moved toward that city with their weights determined by using Gaussian kernel. In the second module, the rest of neurons outside the subset in the attracting module are migrated away from the current positions in such a way that they still maintain the mean of them to be the mean of the neurons within the subset in the first module. Finally, the cities in the order of their subset are connected to be a tour. As a result, this approach achieves 3.7% of the deviation from the optimal solutions (on the average), testing on 15 TSP instances.

Later, Cochrane and Beasley (2003) introduce a new self-organizing neural network approach for solving TSP, namely the co-adaptive net algorithm. Their approach is based on self-organizing map (SOM) which is combined with a new model of lateral interactions between neurons, or co-operation between them. In addition, their algorithm utilizes several algorithmic mechanisms that have contributions on the computational success of previous neural network algorithms, such as an adaptive neuron neighborhood, a use of co-operation in learning process, a neuron-specific adaptation process, and a near-tour to tour construction. As a result, this algorithm performs well on the given standard test problem, and it achieves trade-off between solution quality and running time, in comparison to other previous neural network approaches.

Saadatmand-Tarzjan *et al.* (2007) present constructive-optimizer neural network (CONN) by utilizing a feedback structure similar to Hopfield-type neural networks and a competitive training algorithm similar to the Kohonen-type self-organizing maps (K-SOMs). At the beginning of their algorithm, an initial tour is constructed, and it is trained in the constructive phase of CONN (by inserting a number of cities into the tour). Next, the optimizer phase of CONN is optimized the

Copyright by Kasetsart University All rights reserved

current tour, by displacing the cities in a given tour, until the process converges to some tour. Then these two phases are repeated until all cities are in tour. In the experimental results, the CONN algorithm outperforms the other previous neural network approaches (i.e., KNIES-TSP, Budinich's SOM, Co-adaptive net, and multivalued Hopfield network). In addition, the CONN algorithm consumes less computation time, but its solution quality is worst, in comparison to the 2-opt and 4-opt heuristics, the accurate variant of the simulated annealing algorithm, and the iterated tabu search.

The hybridization algorithm of the memetic method and the self-organizing map (SOM) operator, namely the memetic-SOM algorithm, is proposed by Créput and Koukam (2009). In it, at each iteration, the individuals in the population are randomly generated. Next, the standard SOM and its parameters are applied to the ring networks (each individual in population), then the mapping operator is applied for generating the admissible candidate solution and modifying the shape of each ring network. The mapping operator, in addition, maps the cities to their nearest neuron (to the unvisited nearest city), and moves the neurons toward the city in the tour until all cities are in tour. Consequently, these results in the ring network become a tour. Next, the fitness operator evaluates the fitness value of individuals in population (length of tours), and then the selection operator selects and replaces the worst individuals (having maximum fitness values in population) by the same number of the best individuals. These processes are iterated until the number of generation is reached, and the best individual becomes the resulting tour of the algorithm. As a result, the memetic-SOM algorithm outperforms other previous approaches (i.e., the co-adaptive net, the ESOM, and the eISOM algorithms) on the given standard test problems, in both terms of solution quality and computation time.

Yi *et al.* (2009) introduce the improved elastic net algorithm (called the improved-EN), where several time-dependent parameters are applied into the original elastic net algorithm as the penalty. The elastic net method is based on the physics theories. At the beginning of this algorithm, a set of cities and a set of dynamic points (defined by the rubber band) are given, and the rubber band is a small

Copyright by Kasetsart University All rights reserved

circle around the center of gravity of the cities. Next, the rubber band is extended toward the cities that are nearest by minimizing the energy function as the annealing temperature converges to zero; then those nearest cities are pulled toward the tour. These processes are iterated by extending the rubber band until all cities are mapped and formed to be a shortest tour. As a result, the improved-EN algorithm remarkably outperforms the original elastic net approach on the given TSP standard test problems, in both terms of solution quality and running time.

1.2.5 Metaheuristics

In the last decade, metaheuristics provide a generalized frameworks to build heuristics for solving the combinatorial and global optimization problems, by efficiently exploring the search space in order to produce high-quality solutions (Blum and Roli, 2003; Hansen *et al.*, 2008). They are constituted by applying the techniques, ranging from simple local search procedures to complex learning processes, and they may incorporate mechanisms to avoid getting the solution trap (Blum and Roli, 2003).

In addition, many metaheuristics are widely used for solving the TSP; for example, ant colony optimization (Dorigo and Gambardella, 1997; Qi, 2007; Dong *et al.*, 2012), bees optimization (Marinakis *et al.*, 2011), tabu search (Gendreau *et al.*, 1998), particle swarm optimization (Shi *et al.*, 2007; Chen and Chien, 2011; Liao *et al.*, 2012), guided local search (Voudouris and Tsang, 1999), iterated local search (Martin *et al.*, 1991, 1992; Martin and Otto, 1996; Stützle and Hoos, 2002), variable neighborhood search (Burke *et al.*, 2001; Hansen and Mladenović, 2001; Hansen *et al.*, 2008; Mladenović *et al.*, 2012). Moreover, well-known heuristics (including genetic algorithm and simulated annealing algorithm) are considered metaheuristics (Blum and Roli, 2003). Some recently proposed metaheuristics are discussed in more detail, as follow:

Marinakis *et al.* (2011) present the hybridization of honey bees mating optimization algorithm which combines the honey bees mating optimization,

Copyright by Kasetsart University All rights reserved

the multiple phase neighborhood search-greedy randomized adaptive search procedure algorithm, and the expanding neighborhood search strategy. In addition, a new crossover operator based on an adaptive memory procedure is applied, which allows storing the queen from previously selected good drones part of their solutions (after the previous mating flights). Then they can use it to mate in a new flight for producing fittest broods. As a result, this algorithm achieves solutions close to the optimal solutions, and it yields remarkably better solution quality than the benchmarking algorithm on the set of the standard test problems.

In addition, Chen and Chien (2011) propose the hybridized metaheuristic of the genetic algorithm (GA), the simulated annealing (SA) algorithm, the ant colony optimization (ACO), and the particle swarm optimization (PSO). In this algorithm, it starts with ACO by generating the group of ants and putting the initial pheromone on each edge. Then every ant, in every group, completes its traveling sequence of cities, and this sequence is computed its length together with performing the pheromone update for the next iteration. The sequence obtained from ACO is encoded into chromosomes of GA and these chromosomes are executed through a usual GA procedure. In addition, in the mutation step, the simulated annealing technique is applied to mutate the chromosomes produced by the crossover operation, and their fitness values for each group are evaluated. The PSO technique is then utilized to exchange the pheromone information among groups, and then this algorithm returns to the beginning. These processes from ACO throughout PSO are iterated until the stopping criterion (the maximum number of iterations) is reached, then it returns the best tour and its corresponding length. As a result, this hybridized metaheuristic outperforms the other previous approaches, in both terms of solution quality and computation time.

The hybridization of the genetic algorithm (GA) and the ant colony optimization (ACO) metaheuristic, called the cooperative genetic ant systems algorithm (CGAS), is introduced by Dong *et al.* (2012). In their algorithm, it is designed to co-operate between GA and ACO that they exchange the mutual information in the end of the current iteration. This cooperative procedure ensures to

Copyright by Kasetsart University All rights reserved

select the best solutions for the next iteration and to give a better chance for reaching the globally optimal solution. The experimental results of their algorithm show that it outperforms the other previous GA and ACO algorithms, in terms of searching the globally optimal solution and the solution quality, particularly for small test problem. However, on the overall, the CGAS algorithm spends more running time than both comparing algorithms.

Mladenović *et al.* (2012) propose the variable neighborhood search (VNS) method for solving the traveling salesman problem with the one-commodity pickup and delivery, where a vehicle starts at the depot and visits each of the customers (either pickup customers or delivery customers) exactly once with a given amount of a single product. In their proposed algorithm, they apply three different neighborhood operators: *k*-opt, double-bridge, and insertion (forward and backward) operators, and use the binary indexed tree structure for efficient feasibility checking and updating of solutions in these neighborhoods. As a result, their proposed VNS algorithm outperforms the best-known algorithms, e.g., UB2-heuristics by Hernández-Pérez and Salazar-González (2004), in terms of both the solution quality and running time. Moreover, by using their algorithm, the best-known solutions of some benchmark instances (with 200 to 500 customers) are improved, and it can solve instances with size up to 1,000 customers.

In order to improve the solution quality, the iterated local search is applied for the proposed algorithm in this research. The next section discusses the conceptual framework of the iterated local search approach for solving the optimization problem, and some related literatures of its applications are illustrated.

2. Iterated Local Search

The iterated local search (ILS) is a simple and widely applicable metaheuristic which iteratively applies the local search to improve a candidate solution (Lourenço *et al.*, 2010). ILS applies local search (by moving from one solution to another in the search space with the improving change in the value of objective function until either

Copyright by Kasetsart University All rights reserved

a satisfying solution is found or some stopping criterion is satisfied) to some initial solution at the beginning of algorithm, and then a main loop is iterated until some stopping criterion is satisfied. In each main loop of ILS, the modification step (kick-move) yields a new locally optimal solution according to a given local search and a preceding solution. These two solutions are evaluated in the acceptance step before starting the next loop, i.e., the new solution is accepted surely if it is better, or it is allowed to continue the next kick-move with some acceptance criterion. The general algorithmic framework of ILS is illustrated in Figure 2.

Procedure *Iterated Local Search*
  $s_0 \leftarrow$ Generate initial solution
  $s \leftarrow \text{Local Search}(s_0)$
  repeat
    $s' \leftarrow \text{Perturbation}(s, history)$
    $s'' \leftarrow \text{Local Search}(s')$
    $s \leftarrow \text{Acceptances Criterion}(s, s'', history)$
  until termination condition met
end *Iterated Local Search*

Figure 2 General algorithmic outline of the iterated local search (ILS)

Source: Stützle (2006)

In some aspect of ILS searching, the solution may be stuck at some locally optimal solution for a long time. Some previous literatures (Stützle and Hoos, 2002; Stützle, 2006; Lourenço *et al.*, 2010) introduce a restarting technique (namely solution perturbation) to avoid this trapped point; consequently, the search still executes with the next kick-move. This step takes a search away from a trapped point s^* to s' like hill climbing, as shown in Figure 3. In addition, the multi-start procedures are modern contributions applied to several algorithms in order to achieve high-quality solutions for solving combinatorial optimization problems, and two categories of historical multi-start methods are reviewed by Martí *et al.* (2013). These two categories of multi-start methods relate to memory-based and memoryless procedures;

Copyright by Kasetsart University All rights reserved

for instance, the randomized multi-start algorithm, the randomized greedy multi-start algorithm, the adaptive multi-restart method, the multi-restart method with adaptive reasoning techniques, and the multi-restart method with adaptive memory search principles.

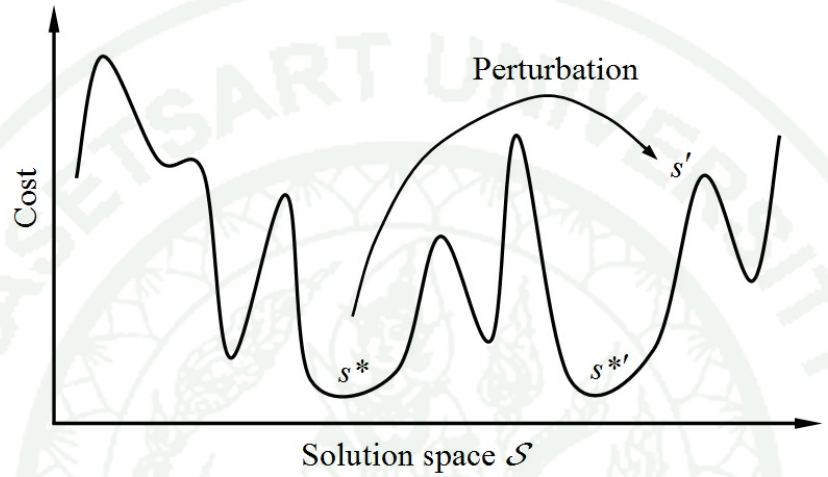


Figure 3 Pictorial representation of the iterated local search

Source: Lourenço *et al.* (2010)

Moreover, there are several preceding algorithms correspond to ILS under different name, such as large-step Markov chains (Martin *et al.*, 1992), chained local optimization (Martin and Otto, 1996), and variable neighborhood search (Hansen *et al.*, 2008). On the other hand, some previous literatures report ILS algorithm as the most powerful algorithm for TSP (Martin *et al.*, 1992; Martin and Otto, 1996; Johnson and McGeoch, 2003), on which the implementation of ILS with the Lin-Kernighan (LK) local search procedure provides the best trade-off between computer runtime and solution quality (Martin and Otto, 1996; Johnson and McGeoch, 2003).

In addition, the convergence analysis of the ILS algorithm becomes a way to show that the algorithm still proceeds after restarting the solution and concentrates to optimality. To address this onto ILS, the current solution of the ILS algorithm relates to only the preceding solution, and all solutions are considered together as a chain.

Copyright by Kasetsart University All rights reserved

This characteristic seems to have a Markov property, i.e., the conditional probability of making a next transition to any future state, given the past states and the present state, depends only on the present state (Lourenço *et al.*, 2010). Consequently, the appropriate method of convergence must be analyzed based on theory of Markov chain. In addition, Markov chain theory is widely-used to explain the convergence characteristic of some stochastic search method, such as hill-climbing algorithms (Johnson and Jacobson, 2002a, 2002b; Jacobson and Yücesan, 2004a, 2004b), simulated annealing algorithms (Nolte and Schrader, 2000; Aarts *et al.*, 2003; Aarts *et al.*, 2005), and genetic algorithm (Dawid, 1994; Suzuki, 1995; Dorea *et al.*, 2010).

Furthermore, the applications of the ILS method can be found in a variety of the combinatorial optimization problems. For example, Stützle (2006) applies the ILS algorithm to the quadratic assignment problem and analyzes its performance in both terms of solution quality and run-time behavior. Next, Erdoğan *et al.* (2009) introduce the pickup and delivery traveling salesman problem with first-in-first-out loading fashion, and they embed the ILS method into the proposed algorithm to improve the feasible solution for this problem. Later, Chen *et al.* (2010) introduce the iterated variable neighborhood descent algorithm to determine the minimum total cost routes for the capacitated vehicle routing problem.

In addition, Fanjul-Peyro and Ruiz (2010) propose the iterated tabu search approach for solving each sub-problem of the circular open dimension problem to minimize the length of the strip, in which a set of circles of known radii is packed. Next, Dong *et al.* (2011) apply the ILS approach for solving the permutation flowshop scheduling problem with total flowtime objective, and they investigate on the effect of four different search sequence strategy on this problem. Later, Derbel *et al.* (2012) propose the genetic algorithm (GA) incorporating with the ILS approach for the location routing problem with multiple capacitated depots and one uncapacitated vehicle per depot. Here, the ILS method is used to intensify the search space and to improve the solution generated by GA. Finally, Dong *et al.* (2013) introduce the ILS algorithm with multi-restart technique for minimizing total flowtime of the permutation flowshop problem.

Copyright by Kasetsart University All rights reserved

This section discusses the ILS framework and its applications in a variety of the combinatorial optimization problems. The next section introduces a probabilistic approach (that utilizes the Bayesian inference and the principle of learning), namely Gaussian process regression, which is applied in our proposed methods.

3. Gaussian Process Regression

The Gaussian process regression (GPR) is known as a probabilistic approach to be used in a regression model due to its practical and theoretical simplicity and excellent generalization ability (Rasmussen and Williams, 2006). The GPR is a model to estimate the value of a dependent variable or output by using some observations of dependent variables at certain values of the independent variable. The assumption of GPR is that the observation arises from some unknown function, and it may be corrupted by unknown Gaussian noise. Given the n observations of training input, the GPR parameterizes a given covariance function and infers the parameter values given the training data, which is a training set of input-output pairs, by using the maximum likelihood method (Rasmussen and Williams, 2006). The prediction step uses the inferred parameter values (called hyperameters) to make a prediction with a given test input. Consequently, the predictive function output and its variance are computed.

In addition, the computational cost is a serious issue for using GPR. Artless implementations need $O\left(N^3\right)$ time for inverting a kernel matrix. Many approaches for speeding up GPR have been proposed to date. Reducing the size of a problem is a popular approach to increase the computational speed, e.g., Nyström's method (Williams and Seeger, 2001) and pseudo-input-based methods (Quiñonero-Candela and Rasmussen, 2005; Snelson and Ghahramani, 2006). Though one can say that these methods are not completely adequate for analytical applications because most of them are sensitive to a parameter selection (Williams *et al.*, 2002), or they require solving a sophisticated non-convex optimization problem (Williams *et al.*, 2002; Rasmussen and Williams, 2006).

Copyright by Kasetsart University All rights reserved

For applications, GPR is applied in many fields, particularly in machine learning, e.g., to learn a user preference function for automatically generating music playlists (Platt *et al.*, 2002), to estimate the depth of a point in space from observing its image position in two different cameras (Sinz *et al.*, 2004), to learn motion and observation non-parametric system models for sequential state estimation and apply its algorithm to the problem of tracking an autonomous micro-blimp (Ko *et al.*, 2007), to optimize the automatic gait problem of the Sony AIBO ERS-7 with a Baysian approach based on GPR (Lizotte *et al.*, 2007), to find near-optimal sensor placements in the task as an instance of the art-gallery problem (Krause *et al.*, 2008).

In addition, GPR is applied to improve quality in some manufacturing processes, e.g., to optimize the reliable multi-objective of high-speed WEDM process based on Gaussian process regression (Yuan *et al.*, 2008), to estimate the quality variables, i.e., the melt flow rate (MFR), in a polypropylene polymerization process using the GPR model (Chen and Ren, 2009), to model pile capacity using GPR (Pal and Deswal, 2010). Furthermore, for the traffic problem, GPR is applied for predicting the traveling time for an arbitrary path and tested with realistic traffic data of downtown Kyoto, Japan (Idé and Kato, 2009).

This section explains the conceptual framework of Gaussian process regression, and its applications under the related fields are provided. In order to develop a provable approach, the next section provides a survey of the convergence methods for proving the optimality of some algorithms, which can be applied for solving the combinatorial optimization problems.

4. Optimality Proof for the Approximation Algorithms

The optimality proof of algorithm is an approach to analyze the theoretical convergence of a given algorithm, and to guarantee that the solution obtained converges to the optimal solution when it executes with sufficiently large iterations (i.e., approaching infinity). This proof investigates an in-depth analysis on the search performance of a given algorithm, in terms of solutions and the number of iterations,

Copyright by Kasetsart University All rights reserved

which reflects an effectiveness of an algorithm to be performed on visiting a globally optimal solution. Although there are not much previous literatures based on the convergence proof of the approximation algorithms (heuristics), it should be explored in order to convince that the solution concentrates on optimality. Consequently, this section provides literature reviews on the convergence approaches that are utilized to prove the optimality of five well-known methodologies: generalized hill climbing algorithm, simulated annealing algorithm, genetic algorithm, ant colony optimization, tabu search, particle swarm optimization, and Lin-Kernighan heuristic, which are widely applied for solving the combinatorial optimization problems, as follow:

4.1 Generalized hill climbing algorithm

For discrete optimization problem with minimized objective function, the generalized hill climbing (GHC) algorithm is an approach to model the local search strategies with a general probabilistic acceptance model, which probabilistically accepts the deteriorating solution generated by some neighborhood function over the solution space. This acceptance model can reach one of a set of the globally optimal solutions. On the convergence analysis of the GHC algorithms, Johnson and Jacobson (2002a) present a framework for GHC algorithms applied to discrete optimization (minimization) problem. They illustrate that the GHC algorithm converges to the set of globally optimal solutions through a homogeneous Markov chain. Moreover, they describe the class of convergent GHC algorithms with distinct acceptance probability forms that can be defined by various distributions. As a result, the GHC algorithm converges to one of a set of globally optimal solutions in probability.

They also introduce a concept of path probabilities between local and global optima for relaxing the sufficient conditions, which is required in most convergence proof for stochastic local search algorithm (Johnson and Jacobson, 2002b). In their work, they model the GHC algorithm as an inhomogeneous Markov chain, i.e., a sequence of many homogeneous chains based on the number of outer loop iterations. In addition, they provide not only a short-numerical example to show

Copyright by Kasetsart University All rights reserved

the GHC algorithm with a rational function of iteration index satisfies their convergence conditions, but also the results that GHC algorithm is formulated as other acceptance formulations.

Vaughan and Jacobson (2004) introduce the analysis framework to model the generalized hill climbing algorithm, in which the tabu search strategies are applied to probabilistically accept a current candidate solution for discrete optimization problems. In their paper, they provide sufficient conditions and properties for proving this algorithm based on the previous results in Johnson and Jacobson (2002b). In addition, the algorithm is modeled as a set of stationary Markov chains, by fixing the tabu list for each outer loop iteration. The result shows that this algorithm converges to the stationary distributions with zero weight for all solutions, that are neither locally or globally optimal solutions, when the number of iteration approaches infinity.

Jacobson and Yücesan (2004a) provide a framework to analyze the performance of metaheuristics, which have the scheme similar to the GHC algorithm, and which can be addressed to discrete optimization problems. In this literature, they provide necessary and sufficient convergence (in probability) conditions for such methods, and compare these conditions to the convergence conditions for the simulated annealing algorithm (in term of the rate when the cooling schedule approaches zero). Moreover, they show that the random restart local search outperforms the simulated annealing algorithm, in term of visiting a global optimum, when executing with a sufficiently large number of restarts.

In addition, they introduce the performance measures to visit a global optimum for modeling the generalized hill climbing (GHC) algorithm (Jacobson and Yücesan, 2004b). In their work, they provide the necessary asymptotic properties with the false-negative probability, which is the probability that the GHC algorithm does not visit any element of a set of globally optimal solutions over some initial number of outer loop iterations. This probability means to assess that the GHC algorithm visits a set of globally optimal solutions when the probability at the outer

Copyright by Kasetsart University All rights reserved

loop iteration is sufficiently close to zero, and then the GHC algorithm is terminated. In addition, the convergence analysis is derived through the first-step analysis of a homogeneous discrete-time Markov chain. In the experiments, they apply these performance measures to four schemes of GHC algorithms: Monte Carlo search, random-restart local search, threshold accepting algorithm, and simulated annealing algorithm. The results show that the Monte Carlo search, the random-restart local search, and the simulated annealing algorithm visit to the set of globally optimal solutions in probability while the threshold accepting algorithm does not converge.

Furthermore, Vaughan *et al.* (2007) present a framework to analyze the performance of simultaneous generalized hill climbing (SGHC) algorithm in which many well-known heuristics can be embedded, including the simulated annealing algorithm and the threshold accepting techniques. In addition, the SGHC algorithm is applied for simultaneously addressing instances of sets of the related discrete optimization problems, and it allows movements between these optimization problems during its execution according to a probability mass function of the problem. In their paper, they introduce the convergence properties of the SGHC algorithm through the Markov chain theory, where the solutions obtained by this algorithm satisfy the Markov property. In addition, two different types (homogeneous and inhomogeneous approaches) of Markov chain model are investigated. Consequently, the sufficient conditions for guaranteeing that the SGHC algorithm visits to one of the set of globally optimal solutions are provided, and the rate of convergence of SGHC algorithm is also preserved according to a given embedded GHC algorithm.

Because of the various type of the acceptance probability forms in the class of convergent GHC algorithm, the simulated annealing (SA) algorithm is based on GHC algorithm with its acceptance probability form defined by the exponential distribution (Johnson and Jacobson, 2002a). Consequently, the next subsection discusses the convergence methods of the SA algorithm.

Copyright by Kasetsart University All rights reserved

4.2 Simulated annealing algorithm

The simulated annealing (SA) algorithm is based on the concept of hill climbing that the deteriorating moves between solutions are probabilistically accepted by a function of the change in solution costs, as shown in Equation (1). In addition, the SA algorithm has proved to be a successful approach applied for a variety of the combinatorial optimization problems (Nolte and Schrader, 2000). In this subsection, some literatures of convergence proof of SA are provided below.

Alkhamis *et al.* (1999) modify the basic convergence proof of the simulated annealing (SA) algorithm to a stochastic optimization problem, where the objective function is corrupted by random noise, and it can be estimated through Monte Carlo simulation. They derive the convergence properties of their proposed SA algorithm through Markov chain. In addition, they demonstrate that the performance of their proposed algorithm converges in probability to the set of globally optimal solutions under suitable noisy conditions.

Next, Nolte and Schrader (2000) propose a proof technique to study the finite time behavior of simulated annealing (SA) algorithm. In their proof, the random variable in the algorithm is modeled as state of an inhomogeneous Markov chain. The transition probabilities are presumably fixed at temperature T_k, since the limit distribution exists in an ergodic homogeneous chain. Moreover, the logarithmic cooling schedule is applied in their proposed algorithm because it is proved an asymptotic convergence in previous literatures. As a result, their proposed technique give considerably better bound for the finite time behavior of SA.

According to the solution acceptance, in practice, the SA algorithm is executed with finite time, and its solutions may not reach a global optimum. However, the solutions may be acceptable if they are close enough to globally optimal solutions. Consequently, the asymptotic convergence properties and the finite-time performance measures for the acceptable solutions are proposed, analyzed by the Markov chain theory. Henderson (2001) introduces the convergence proof of a

Copyright by Kasetsart University All rights reserved

globally optimal solution with λ-acceptable solution concept. The λ-acceptable solution value is an element in the range between the global optima and the maximum value of local optima. In other words, the solution is accepted if its cost falls in the range of λ-acceptable solution value.

Jacobson *et al*. (2005) present a framework for assessing the performance of the cyclical simulated annealing (CSA) algorithms for addressing the discrete optimization problems. The CSA algorithms use a cyclical cooling schedule through a set of temperature values. In their paper, they consider the analysis on the expected number of iterations to visit the set of λ-acceptable solutions for the first time. Moreover, they apply the proposed framework to four instances of the traveling salesman problem taken from TSPLIB (Reinelt, 1995), and show that the expected number of iterations to visit the set of λ-acceptable solutions (by using the CSA algorithms) can be estimated.

Orosz and Jacobson (2002a, 2002b) analyze the static term of the simulated annealing algorithm, where the cooling schedule is fixed during the algorithm execution. Moreover, they introduce a conditional one-step λ-acceptable solutions to evaluate the run-length, or the finite-time performance measure of algorithm. Their proposed technique provides an expression for the number of iterations to reach a λ_1-acceptable solution, given the predetermined objective function value within the set of a λ_2-acceptable solution that has been reached in a finite number of iterations, where each set is close enough to a globally optimal solution and $\lambda_1 \leq \lambda_2$. Furthermore, they provide both of the proof of algorithm properties and the numerical results with the traveling salesman problem.

Ghate and Smith (2008) present the simulated annealing algorithm in continuous space for reaching a globally optimal solution with stochastic acceptance probabilities for the optimization (minimization) problem. They provide an algorithm under asymptotically diminishing errors condition, i.e., the objective function becomes stochastic with additive noise, and prove that it converges to the optimal

Copyright by Kasetsart University All rights reserved

value in probability by using the Markov chain Monte Carlo approach. In addition, they perform the numerical results of this algorithm on a noisy protein-folding problem.

This section illustrates that the widely-used convergence analysis for proving optimality of the simulated annealing algorithm is based on Markov chain theory. Next, the investigation on convergence analysis of the genetic algorithm is discussed.

4.3 Genetic algorithm

The genetic algorithm (GA) is a well-known heuristic that are successfully applied for solving a variety of the combinatorial optimization problems. In addition, the GA framework can be seen in Section 1.2.3 above. In order to prove the optimality, many literatures investigate the convergence analysis of the GA algorithm on several contributions upon genetic operations with its strategies, as follow:

Rudolph (1994) presents the convergence analysis of canonical genetic algorithm (CGA) applied to static optimization problems. The convergence analysis describes the properties of CGA and its modified version by keeping the best solution around the crossover operation. In addition, the homogeneous Markov chain with finite state space theory is used for investigating the convergence behavior of both algorithms. The results, under conditions in his paper, show that the CGA does not converge to the globally optimal solution while the modified one does with probability one in the limit.

Suzuki (1995) present a convergence analysis of simple genetic algorithms through Markov chain theory. In his work, he modifies the population generation strategy by reserving the best individual with the highest fitness value from the previous generation and placing them into the current population. In addition, he assumes that the analysis is based on the simple genetic operations (selection,

Copyright by Kasetsart University All rights reserved

crossover, and mutation), and it is evaluated on the eigenvalues of the transition matrix of the Markov chain. Consequently, the convergence rates of GA algorithms are evaluated in terms of a mutation probability, by minimizing the eigenvalues of the transition matrix.

Later, he proposes the limiting behavior analysis of the genetic algorithm, based on finite-state Markov chain of populations with the large deviation principle approach, which focuses on the best uniform population when the selective pressure approaches infinity, and the mutation probability becomes zero (Suzuki, 2010). He refines the sufficient conditions from previous works, e.g., to reorder the cycle of genetic operation, and to investigate the effect of crossover. Moreover, the property of stationary distribution is true for every positive number of population size.

Schmitt (2004) provides a theoretical framework of the genetic algorithms (GAs), for modeling several versions of genetic operator (over the string-tensor representation of populations) and asymptotically converging to globally optimal solution, through an inhomogeneous Markov chain. This literature shows that the alphabet can be interpreted as a finite set of equidistant real numbers such that the GAs are well-suited for optimization. In addition, it exposes that the contraction properties of the mutation and fitness selection operators are driven towards uniform populations. The scaled algorithm with explicit annealing schedules can be described through the inhomogeneous Markov chain with a weakly ergodic property, and the scaled genetic algorithm converges to a set of the globally optimal solutions in probability.

Dorea *et al.* (2010) analyze and model the genetic algorithm with multistage Markov chain (MMC), where each of genetic operations is modeled as stages of a Markov chain. MMC consists of three transition matrices in which the individual generation of population is represented as states of the chains. Moreover, their convergence results demonstrate that the stationary distribution exists certainly for all cases, including the simulated annealing algorithm and two cases of GA, as Markov chain.

Copyright by Kasetsart University All rights reserved

This section demonstrates that the convergence properties of the genetic algorithms are widely analyzed through the Markov chain theory, and the analysis is modeled upon the characteristics of each genetic operation. Next, the survey on convergence analysis of other heuristics, such as ant colony optimization algorithms, particle swarm optimization algorithms, tabu search heuristics, and Lin-Kernighan heuristic, is discussed.

4.4 Other heuristics

This section discusses on some research works that are related to the convergence analysis of four heuristics, i.e., tabu search, ant colony optimization, particle swarm optimization, and Lin-Kernighan heuristic.

Hanafi (2001) mathematically proves that the Glover's conjecture on tabu search metaheuristic (i.e., a choice rule for remedying the situation where the algorithm is forced to revisit the earliest encountered solutions), in the previous literature is true. Under his conditions that the neighborhood structure is symmetric and the graph induced by the neighborhood structure is strongly connected, the algorithm still explores the search space and its solution converges to global optimum in limit.

Gutjahr (2002) provides the stochastic convergence properties of the ant colony optimization (ACO) algorithms for two types of pheromone updates, i.e., time-dependent evaporation factor and time-dependent lower pheromone bound. This literature describes the states: the vector of pheromone values and the best-encountered path of the ACO algorithm at the beginning of iterations, as an inhomogeneous Markov process. Under the specified conditions, the algorithms guarantees to converge with probability one when the iteration number approaches infinity.

Gutjahr (2008) analyzes the runtime of two ACO algorithms: the first one is the ACO with time-dependent lower pheromone bound described in Gutjahr (2002),

Copyright by Kasetsart University All rights reserved

and the second one is the oldest ACO variant by Dorigo *et al.* (1996), by using the well-known OneMax function. In addition, in both cases, the stochastic process is applied to investigate the asymptotic convergence analysis of algorithms on the expected runtime until reaching the optimal solution. According to his theoretical convergence analysis, the runtime that is sufficient for reaching an optimal solution is of order $O(n \log n)$, where n is the test instance size.

Trelea (2003) discusses the dynamic behavior and convergence of the particle swarm optimization (PSO) algorithm by using the discrete-time dynamic system theory. The analysis is experimented on the number of particles in the swarm, the two set of parameters (the momentum factor and attraction coefficient), and the optimization test functions. Consequently, the results in his investigation are used as a graphical guideline for the PSO parameter selection to obtain the global optima according to the required exploration-exploitation trade-off.

Jiang *et al.* (2007) present the convergence analysis and the parameter selection of the standard PSO algorithm using the stochastic process theory. The analysis considers the particle's position in the swarm at each iteration as states of stochastic process. The particle position's expectation and variance are calculated corresponding to the iterative process. Without loss of generality, under the given convergence property, the result shows that the PSO algorithm converges in mean square to the particle's position with the best fitness in the neighborhood; in other words, it converges in probability to the best particle's position.

Nikolaev and Jacobson (2011) introduce the Markov chain pooling framework to reduce state space of the Lin-Kernighan heuristic. They use this technique to analyze the performance of local search methods, by selecting the best-to-date solution at each iteration. In addition, they provide a procedure and its convergence proof for determining the expected number of iteration for the algorithm to reach the set of globally optimal solutions with λ-acceptable solutions concept (Henderson, 2001). Moreover, they apply this framework with the Lin-Kernighan-

Copyright by Kasetsart University All rights reserved

Helsgaun algorithm (Helsgaun, 2000) executed with multiple restart. The numerical experiments are executed with eight instances of the TSP taken from TSPLIB (Reinelt, 1995). As a result, for 14 (out of 15) experiments, the estimated values are statistically indistinguishable from the validation estimates, and the computation time depends on the size of the TSP instance.

This chapter provides the literature reviews of related methods that are utilized in this research. These reviews consist of the characteristic of the traveling salesman problem and its related methods, the Gaussian process regression model, the iterated local search, and the convergence methods for the approximation algorithms. Next chapter explains the proposed GPR approach for solving the TSP and its convergence analysis for the optimality.

Copyright by Kasetsart University All rights reserved

MATERIALS AND METHODS

Materials

1. A laptop computer consists with the following specifications:

   1.1 Personal computer running Intel(R) Pentium(R) Dual CPU E2180 with a frequency of 2.00 GHz. processor

   1.2 1 GB. of memory

   1.3 250 GB. Hard disk drive

   1.4 ATI Radeon HD 2400 Pro graphic card

   1.5 17 inches LCD monitor

2. MATLAB computing software
3. Microsoft Excel software
4. Brother DCP-7010 Laser Printer

Methods

This chapter explains the theoretical framework of the Gaussian process regression for solving the traveling salesman problem (TSP), our two proposed methods, the convergence analysis of our proposed methods, and the experimental settings and comparison, respectively. The notations used in this thesis are: the small bold Roman letter (symbol) represents the vector and its subscript letter represents the entry number, e.g., $\mathbf{x}_i$ denotes the ith element of vector $\mathbf{x}$. In this thesis, all vectors are column vectors. The capital bold-italicized Roman letter (symbol) represents the matrix, such as a matrix $\boldsymbol{X}$. The small-italicized Roman letter (symbol) denotes the function value and its subscript letter represents the entry number, e.g., y_i denotes the function value of the ith input. The identity matrix is $\boldsymbol{I}$. The estimated value is denoted with a hat, e.g., $\hat{y}^*$, and the improved value of this value is denoted with double-hat, e.g., $\hat{\hat{y}}^*$. Finally, the uppercase superscript letter "T" represents the transposition operator of a vector (or a matrix).

Copyright by Kasetsart University All rights reserved

1. Gaussian Process Regression Model for TSP

This section discusses the theoretical framework of the Gaussian process regression (GPR) for solving the symmetric TSP. For completeness, the framework of standard linear regression is first explained in Section 1.1, and then the extension of this framework to the GPR model is described in Section 1.2.

1.1 Standard linear regression model

Given a training set $\mathcal{T}_s$ of n observations, $\mathcal{T}_s = \{(\mathbf{x}_i, y_i) | i = 1,...,n\}$, $\mathbf{x}_i$ denotes an input vector (the ith traveling tour) of n_c cities, and y_i denotes the length of the ith traveling tour. The column vector inputs $\mathbf{x}_i$ for all n cases are aggregated into the $n_c \times n$ matrix of $\boldsymbol{X}$, and the total distance y_i in each traveling tour are aggregated in the column vector $\mathbf{y}$, so the training set can be written as $\mathcal{T}_s = (\boldsymbol{X}, \mathbf{y})$. The standard linear regression model with Gaussian noise is

$$y_i = f(\mathbf{x}_i) + \varepsilon_i,$$

$$f(\mathbf{x}_i) = \mathbf{x}_i^{\mathrm{T}} \mathbf{w},$$

where $\mathbf{x}_i$ is the input vector, $\mathbf{w}$ is a vector of weight (parameters) of a linear model or offset, and $f(\mathbf{x}_i)$ is the function value of the input vector $\mathbf{x}_i$ (Rasmussen and Williams, 2006). Assume that the observed values y_i differs from the function value $f(\mathbf{x}_i)$ by an additive noise ε_i, which follows an independent and identically distributed Gaussian distribution with zero mean and variance σ_n^2, that is $\varepsilon_i \sim N(0, \sigma_n^2)$, where n is the number of observations.

Copyright by Kasetsart University All rights reserved

For Gaussian or normal distribution, the probability density of the observations given the parameters which is estimated over all cases in a training set is

$$p\left(\mathbf{y}\middle|\boldsymbol{X},\mathbf{w}\right)=\prod_{i=1}^{n}p\left(\mathbf{y}_i\middle|\mathbf{x}_i,\mathbf{w}_i\right)\sim N\left(\boldsymbol{X}^{\mathrm{T}}\mathbf{w},\sigma_n^2\boldsymbol{I}\right),$$

where $\mathbf{w}$ is a bias weight or offset with a zero mean Gaussian prior with covariance matrix $\sum_p$, $\mathbf{w}\sim N\left(0,\sum_p\right)$. The posterior distribution over the weights based on the Bayesian linear model is computed by Bayes' rule (See Rasmussen and Williams (2006) and Boyle (2007) for review of the Bayesian analysis). The form of posterior Gaussian distribution with mean $\bar{\mathbf{w}}$ and covariance matrix $\boldsymbol{A}^{-1}$ is

$$p\left(\mathbf{w}\middle|\boldsymbol{X},\mathbf{y}\right)\sim N\left(\bar{\mathbf{w}}=\frac{1}{\sigma_n^2}\boldsymbol{A}^{-1}\boldsymbol{X}\mathbf{y},\boldsymbol{A}^{-1}\right), \tag{2}$$

where $\boldsymbol{A}=\sigma_n^{-2}\boldsymbol{X}\boldsymbol{X}^{\mathrm{T}}+\sum_p^{-1}$. To make prediction for a test case, all possible parameter values are averaged and weighted by posterior probability. The predictive distribution for the function value $\mathbf{f}_s\triangleq f\left(\mathbf{x}_s\right)$ at $\mathbf{x}_s$ is given by averaging the output of all possible linear models. Thus the Gaussian posterior is written as

$$p\left(\mathbf{f}_s\middle|\mathbf{x}_s,\boldsymbol{X},\mathbf{y}\right)\sim \boldsymbol{N}\left(\frac{1}{\sigma_n^2}\mathbf{x}_s^{\mathrm{T}}\boldsymbol{A}^{-1}\boldsymbol{X}\mathbf{y},\mathbf{x}_s^{\mathrm{T}}\boldsymbol{A}^{-1}\mathbf{x}_s\right). \tag{3}$$

The predictive distribution in (3) is also Gaussian distribution, with a posterior mean of the weights from (2) multiplied by the all possible values $\mathbf{x}_s$ in a test case. Moreover, the predictive variance is a quadratic form of all possible values $\mathbf{x}_s$ in a test case multiplied with the posterior covariance matrix.

Copyright by Kasetsart University All rights reserved

1.2 Gaussian process regression model

Gaussian process (GP) is a collection of random variables (Rasmussen and Williams, 2006); its mean function $m(\mathbf{x}_r)$ and covariance function $k(\mathbf{x}_r,\mathbf{x}_t)$ are defined a real process as $m(\mathbf{x}_r)=\mathrm{E}\left[f(\mathbf{x}_r)\right]$ and $k(\mathbf{x}_r,\mathbf{x}_t)=\mathrm{E}\left[\left(f(\mathbf{x}_r)-m(\mathbf{x}_r)\right)\left(f(\mathbf{x}_t)-m(\mathbf{x}_t)\right)\right]$. The covariance function $k(\mathbf{x}_r,\mathbf{x}_t)$ is the covariance between two random variables (i.e., two vectors of tour $\mathbf{x}_r$ and tour $\mathbf{x}_t$). Therefore, the Gaussian process can be written as $f(\mathbf{x}_r)=GP\left[m(\mathbf{x}_r),k(\mathbf{x}_r,\mathbf{x}_t)\right]$. Usually the mean function is assumed to be a zero function; thus the Gaussian process can be written as

$$f(\mathbf{x}_r)\sim GP\left[0,k(\mathbf{x}_r,\mathbf{x}_t)\right].$$

Several types of covariance function can be used to define a GP prior, for example, Matérn class of covariance function, squared exponential (SE) covariance function, and radial basis covariance function (Rasmussen and Williams, 2006). The most popular choice is the squared exponential (SE) covariance function that specifies the covariance between pairs of random variables. Consequently, the SE covariance function is chosen for our proposed algorithm, and it is given by

$$\mathrm{cov}\left(f(\mathbf{x}_r),f(\mathbf{x}_t)\right)=k(\mathbf{x}_r,\mathbf{x}_t)=\sigma_f^2\exp\left[-\frac{1}{2\ell^2}(\mathbf{x}_r-\mathbf{x}_t)^{\mathrm{T}}(\mathbf{x}_r-\mathbf{x}_t)\right]+\sigma_n^2\delta(\mathbf{x}_r,\mathbf{x}_t), \quad (4)$$

where $\delta(\mathbf{x}_r,\mathbf{x}_t)$ is the Kronecker delta function which equals to 1 if and only if $r = t$, and 0 otherwise, ℓ denotes the characteristic length-scale, σ_f^2 is the maximum allowable covariance (this should be high for functions which cover a broad range on the y-axis), and σ_n^2 is the white noise variance; obviously, there is much flexibility built into (4).

Copyright by Kasetsart University All rights reserved

To prepare GPR, given the test input $\boldsymbol{X}_s$, the covariance function in (4) is calculated among all possible combinations of these points, summarizing in four matrices:

$$\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}) = \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & k(\mathbf{x}_1,\mathbf{x}_2) & \cdots & k(\mathbf{x}_1,\mathbf{x}_n) \\ k(\mathbf{x}_2,\mathbf{x}_1) & k(\mathbf{x}_2,\mathbf{x}_2) & \cdots & k(\mathbf{x}_2,\mathbf{x}_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & k(\mathbf{x}_n,\mathbf{x}_2) & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix},$$

$$\boldsymbol{K}(\boldsymbol{X}_s,\boldsymbol{X}_s) = k(\mathbf{x}_s,\mathbf{x}_s),$$

$$\boldsymbol{K}(\boldsymbol{X}_s,\boldsymbol{X}) = \begin{bmatrix} k(\mathbf{x}_s,\mathbf{x}_1) & k(\mathbf{x}_s,\mathbf{x}_2) & \cdots & k(\mathbf{x}_s,\mathbf{x}_n) \end{bmatrix},$$

$$\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_s) = \boldsymbol{K}(\boldsymbol{X}_s,\boldsymbol{X})^{\mathrm{T}}.$$

The diagonal elements of $\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})$ are added by $\sigma_f^2+\sigma_n^2$ because they are convenient for determining the stationary covariance function to make prediction, and that its extreme off-diagonal elements are close to zero when $\boldsymbol{X}$ spans a large enough domain. Now, the Gaussian process regression is explained. The prior on the noisy observations, independent and identically distributed Gaussian noise ε_i with variance σ_n^2, becomes $\mathrm{cov}(\mathbf{y}) = \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma_n^2\boldsymbol{I}$. In addition, the joint distribution of the observed target values and function values at the test locations is

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_s \end{bmatrix} \sim N\left(0, \begin{bmatrix} \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})+\sigma_n^2\boldsymbol{I} & \boldsymbol{K}(\boldsymbol{X},\boldsymbol{X}_s) \\ \boldsymbol{K}(\boldsymbol{X}_s,\boldsymbol{X}) & \boldsymbol{K}(\boldsymbol{X}_s,\boldsymbol{X}_s) \end{bmatrix}\right).$$

The predictive distribution for Gaussian process regression is $p(\mathbf{f}_s|\boldsymbol{X},\mathbf{y},\boldsymbol{X}_s) \sim N(\bar{\mathbf{f}}_s,\mathrm{cov}(\mathbf{f}_s))$, where $\bar{\mathbf{f}}_s$ is mean of function values, and $\mathrm{cov}(\mathbf{f}_s)$ is the predictive variance function values (corresponding to test inputs $\boldsymbol{X}_s$). For a

Copyright by Kasetsart University All rights reserved

single test point $\mathbf{x}_s$, let $\mathbf{k}_s \triangleq \mathbf{k}(\boldsymbol{X}, \mathbf{x}_s)$ denotes the vector of covariances between the test point and the n training points, let $\boldsymbol{K} \triangleq \boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})$ be the covariance matrix calculated among all possible combinations of the n training points, and let $\hat{f}_s \triangleq \hat{f}(\mathbf{x}_s)$ be the predictive function value. Thus the predictive function value $\hat{f}_s$ is

$$\boldsymbol{\alpha} = \left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1} \mathbf{y}, \tag{5}$$

$$\hat{f}_s = \mathbf{k}_s^{\mathrm{T}} \boldsymbol{\alpha}. \tag{6}$$

As discussed above, the regression using a Gaussian process with a given fixed covariance function is explained. In many practical applications, it may not be easy to specify all aspects of the covariance function with confidence. It is essential to develop the method of cross-validation for model selection; however, it may be easy to remedy this problem by determining a stationary covariance function. The value of hyperparameters (e.g., length-scales) is determined by the maximum likelihood (Rasmussen and Williams, 2006). The log marginal likelihood is given by

$$\log p(\mathbf{y} | \boldsymbol{X}, \boldsymbol{\theta}) = -\frac{1}{2} \mathbf{y}^{\mathrm{T}} \boldsymbol{\alpha} - \frac{1}{2} \log \left| \boldsymbol{K} + \sigma_n^2 \boldsymbol{I} \right| - \frac{n}{2} \log 2\pi. \tag{7}$$

where $\boldsymbol{\alpha}$ is defined in (5) and $\boldsymbol{\theta}$ is a vector of ℓ, σ_f^2, and σ_n^2. The partial derivative of (7) with respect to $\boldsymbol{\theta}$ is minimized by using a gradient-based optimizer executed generally until the maximum number of iterations is reached (Rasmussen and Williams, 2006; Rasmussen and Nickisch, 2010). However, in this thesis, it is executed until the function value of (7) converges to almost zero; the change in function values between any two consecutive iterations is less than some limit (Lasdon, 2002), which is set to 0.1 (by trial and error). Furthermore, the graphical model for Gaussian process regression can be seen in Figure 4.

Copyright by Kasetsart University All rights reserved

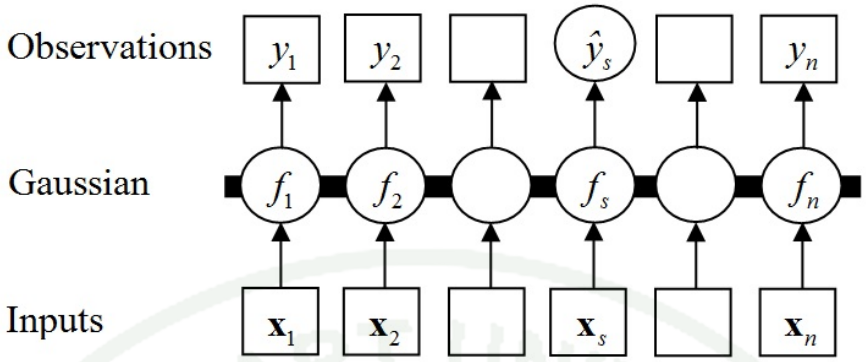


Figure 4 Graphical model for Gaussian process regression

Source: Rasmussen and Williams (2006)

In Section 1.1, a linear model is described with a n_c-city input vector $\mathbf{x}_i$. In this section, the basis function $\phi(\mathbf{x}_i)$ maps a n_c-city input vector $\mathbf{x}_i$ into N_c-dimension feature space (aggregate all n_c-city input vector $\mathbf{x}_i$ into an input matrix $\boldsymbol{X}$). Let the matrix $\Phi(\boldsymbol{X})$ be the aggregation of columns $\phi(\mathbf{x}_i)$ for all cases in training data set, so $f(\mathbf{x}_i) = \phi(\mathbf{x}_i)^{\mathrm{T}}\mathbf{w}$, where the vector of parameters has length N_c. Hence, from (3), the predictive distribution becomes

$$p(\mathbf{f}_s \mid \mathbf{x}_s, \boldsymbol{X}, \mathbf{y}) \sim N\left(\frac{1}{\sigma_n^2}\phi_s^{\mathrm{T}}\boldsymbol{A}^{-1}\Phi\mathbf{y}, \phi_s^{\mathrm{T}}\boldsymbol{A}^{-1}\phi_s\right), \tag{9}$$

with $\Phi = \Phi(\boldsymbol{X})$, $\phi_s = \phi(\mathbf{x}_s)$, and $\boldsymbol{A} = \sigma_n^{-2}\Phi\Phi^{\mathrm{T}} + \boldsymbol{\Sigma_p^{-1}}$ (Rasmussen and Williams, 2006). In the right-hand-side term of (9), the $\boldsymbol{A}$ matrix of size $N_c \times N_c$ is needed to invert for making predictions, which may not be convenient if N_c-dimension feature space is large. However, this term can be rewritten as

$$N\left(\phi_s^{\mathrm{T}}\boldsymbol{\Sigma}_p\Phi\left(\boldsymbol{K}+\sigma_n^2\boldsymbol{I}\right)^{-1}\mathbf{y}, \phi_s^{\mathrm{T}}\boldsymbol{\Sigma}_p\phi_s - \phi_s^{\mathrm{T}}\boldsymbol{\Sigma}_p\Phi\left(\boldsymbol{K}+\sigma_n^2\boldsymbol{I}\right)^{-1}\Phi^{\mathrm{T}}\boldsymbol{\Sigma}_p\phi_s\right), \tag{10}$$

Copyright by Kasetsart University All rights reserved

where the covariance matrix $\boldsymbol{K} = \Phi^{\mathrm{T}} \sum_p \Phi$. To show the proof of (10), first note that using the definitions of $\boldsymbol{A}$ and $\boldsymbol{K}$, we have that

$$\sigma_n^{-2}\Phi\left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right) = \sigma_n^{-2}\Phi\left(\Phi^{\mathrm{T}} \sum_p \Phi + \sigma_n^2 \boldsymbol{I}\right) = \boldsymbol{A}\sum_p \Phi .$$

Multiplying through by $\boldsymbol{A}^{-1}$ from left and $\left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1}$ from the right, we get $\sigma_n^{-2}\boldsymbol{A}^{-1}\Phi = \sum_p \Phi\left(\boldsymbol{K} + \sigma_n^2 \boldsymbol{I}\right)^{-1}$, showing the equivalence of the mean expressions in (9) and (10). For the variance, the matrix inversion lemma is used and it is also known as the Woodbury, Sherman and Morrison formula (Press *et al.*, 2007), which states that

$$\left(\boldsymbol{Z} + \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^{\mathrm{T}}\right)^{-1} = \boldsymbol{Z}^{-1} - \boldsymbol{Z}^{-1}\boldsymbol{U}\left(\boldsymbol{W}^{-1} + \boldsymbol{V}^{\mathrm{T}}\boldsymbol{Z}^{-1}\boldsymbol{U}\right)^{-1}\boldsymbol{V}^{\mathrm{T}}\boldsymbol{Z}^{-1},$$

assuming that the relevant inverses all exist, and then setting: $\boldsymbol{Z}^{-1} = \sum_p$, $\boldsymbol{W}^{-1} = \sigma_n^2 \boldsymbol{I}$ and $\boldsymbol{V} = \boldsymbol{U} = \Phi$ therein.

Notice that in (10) the entries of matrices, in the form of $\Phi^{\mathrm{T}} \sum_p \Phi$, $\phi_s^{\mathrm{T}} \sum_p \Phi$, or $\phi_s^{\mathrm{T}} \sum_p \phi_s$, are unvaryingly of the form $\phi\left(\mathbf{x}_r\right)^{\mathrm{T}} \sum_p \phi\left(\mathbf{x}_t\right)$, where $\mathbf{x}_r$ and $\mathbf{x}_t$ are in the training set and the test set, respectively. Let $k\left(\mathbf{x}_r, \mathbf{x}_t\right) = \phi\left(\mathbf{x}_r\right)^{\mathrm{T}} \sum_p \phi\left(\mathbf{x}_t\right)$, so $k\left(\mathbf{x}_r, \mathbf{x}_t\right)$ is called a covariance function or kernel (Rasmussen and Williams, 2006), which is defined in (4) .

This section concisely explains the theoretical framework of the GPR approach. The next section explains step by step on the implementation of the GPR approach that is applied for solving the symmetric TSP.

Copyright by Kasetsart University All rights reserved

2. Proposed Algorithms

This thesis proposes two GPR algorithms for solving the symmetric TSP, i.e., GPRTSP-0 and GPRTSP-1 algorithms. Both algorithms consist of three phases: the dataset preparation, the GPR prediction, and the tour improvement procedure. The first two phases of both methods are the same while the tour improvement procedures are distinct. The GPRTSP-0 algorithm integrates with the iterated 2-opt algorithm while the GPRTSP-1 algorithm incorporates the iterated local search based on the simulated annealing algorithm. In the first phase of both algorithms, the first tour is constructed by the nearest neighbor method, and the other tours are created by the random 2-exchange strategy. These tours and their corresponding lengths are taken as observations in the training input for the GPR method (Section 1) to make prediction in the second phase. Because the solution obtained from the second phase may not be optimal, the tour improvement procedure is called for improving this solution. The tour construction and representation is briefly explained in the Section 2.1. The tour improvement procedures are explained in more detail in Section 2.2. The procedures of the GPRTSP-0 and GPRTSP-1 algorithms are discussed in more detail in Sections 2.3 and 2.4, respectively.

2.1 Tour construction and representation

The nearest neighbor (NN) algorithm and the two-exchange swapping strategy are used to construct the sample tours (Larrañaga *et al.*, 1999; Applegate *et al.*, 2006). NN is also used to construct the first sample tour. It starts at a randomly chosen starting city, and then selects the next closest unvisited city (from the last city in the set of current cities in the tour) until all n_c cities are included in the tour (Laporte, 1992). In addition, other sample tours are obtained by using the 2-exchange method that randomly selects two cities in the antecedent sample tour, and swaps those cities (Larrañaga *et al.*, 1999).

The sample tours are also represented, and they are treated as an input of the GPR model. Many approaches have been proposed to represent a tour; for

Copyright by Kasetsart University All rights reserved

instance, path representation, binary string representation, binary matrix representation (Larrañaga *et al.*, 1999). Our proposed algorithm uses the binary string representation to represent all tours for GPR input because it is simple, and it performs well when making test prediction. This method encodes each city in a tour as a string of $\lceil \log_2 n_c \rceil$ bits, and then a complete tour becomes a string of $n_c \lceil \log_2 n_c \rceil$ bits (Larrañaga *et al.*, 1999), e.g., a tour $4 \to 3 \to 1 \to 2 \to 5$ is represented by $(011\ 010\ 000\ 001\ 100)$.

This subsection explains the tour construction and representation method for preparing the training input to GPR. The next subsection discusses the tour improvement procedures that are embedded into our proposed algorithms.

2.2 Tour improvement procedures

Several algorithms have been proposed to improve a tour, e.g., the *r*-opt algorithm, Lin-Kernighan heuristic, simulated annealing, and tabu search (Laporte, 1992). However, our proposed algorithms implement the iterated 2-opt algorithm and the iterated local search based on the simulated annealing (SA) algorithm to be the tour improvement procedures.

2.2.1 Iterated 2-opt algorithm

The first proposed algorithm (GPRTSP-0 algorithm) implements the iterated 2-opt algorithm which is a simple local search that moves with a neighborhood search by starting at the first node in a given tour. It removes two edges, breaking the tour into two paths, and then reconnects those paths with two other possible edges. If the new tour is shorter, it becomes the current tour. This process is repeated with the next node until all nodes are selected (Laporte, 1992; Nuhoglu, 2007). The iterated 2-opt procedure is explained in more detail in Section 2.3.3.

Copyright by Kasetsart University All rights reserved

2.2.2 Iterated local search based on the simulated annealing algorithm

The second proposed scheme (GPRTSP-1 algorithm) implements the iterated local search based on the simulated annealing (SA) algorithm (called the iterated SA or the ISA algorithm) because it is one of the provably optimal local searches (Johnson and Jacobson, 2002a; Lourenço *et al.*, 2010). The ISA algorithm applies SA as a local search to some initial tours at the beginning of the algorithm, and then a main loop is iterated until some stopping criterion is satisfied. In each main loop of algorithm, the modification step ("kick-move") yields a new locally optimal solution according to a given local search and a previous solution; then these two solutions are evaluated in the acceptance step before starting the next loop. SA starts from an initial tour at a high temperature that is gradually decreased. A new tour is generated, and the difference in tour-length between a current tour and a new tour is calculated; then the new tour is accepted to be a current tour if it is shorter, or it is accepted with some probability (Laporte, 1992).

In some practical aspects of ISA searching, the solution may be stuck at some locally optimal solutions for a long time; consequently, the ISA algorithm should restart with a new solution under some criterion. Therefore, the 2-opt algorithm is applied for restarting the solution of the ISA algorithm in order to escape from the current local optima. The procedure of ISA algorithm is further described in more details in Section 2.4.

This section discusses the tour improvement methods that apply to both proposed algorithms for improving the estimated solution obtained by the GPR prediction. The GPRTSP-0 algorithm is explained in the next section.

2.3 GPRTSP-0 algorithm

This is a procedure for the symmetric TSP with a single salesman, divided into three phases: preparing the dataset, GPR prediction, and iterated 2-opt procedure.

Copyright by Kasetsart University All rights reserved

2.3.1 Preparing the dataset

This phase describes how to prepare the training dataset $\mathcal{T}_s$ and the testing dataset.

Input: Distance matrix ($\boldsymbol{D}$) where $d_{v,z}$ is the distance from city *v* to city *z*.

Step 1 Construct the sample tours by using NN method and the 2-exchange swapping methods.

Step 2 Encode all tours as binary, and aggregate them into $\boldsymbol{X}$.

Step 3 Calculate the length of each tour (index the distance of each pair of cities in $\boldsymbol{D}$ and sum all values of them), and aggregate it into $\mathbf{y}$.

Step 4 Identify a tour that has a minimum total distance in $\boldsymbol{X}$, and set it as a test input $\mathbf{x}_s$ in (3).

Output: Binary matrix of tours ($\boldsymbol{X}$), vector of observed total distances ($\mathbf{y}$), and binary vector of test tour ($\mathbf{x}_s$).

2.3.2 GPR prediction

This phase implements the GPR model to predict the optimal tour and its length.

Input: Outputs from Section 2.3.1 ($\boldsymbol{X}$, $\mathbf{y}$, $\mathbf{x}_s$) and initial hyperparameters ($\ell, \sigma_f^2, \sigma_n^2$).

Step 1 Compute the SE covariance function of all possible pairs $(\mathbf{x}_r, \mathbf{x}_t)$, as defined by (4).

Copyright by Kasetsart University All rights reserved

Step 2 Compute and minimize the log marginal likelihood, as specified in (7).

Step 3 Compute $\hat{f}_s$ by Equation (6).

Step 4 Compute the squared difference between each member of $\mathbf{y}$ and $\hat{f}_s$, and identify the *r*-th index of its minimum value; that is,

$$r = \arg\min_r \left[\left(y_r - \hat{f}_s \right)^2 \right].$$

Step 5 Determine the binary vector of the chosen tour (which belongs to the *r*-th index in $\boldsymbol{X}$) to be the binary vector of the predictive tour, and then decode it to the vector of the predictive tour $\hat{\mathbf{x}}^*$ (in cities' number).

Output: Estimated optimal tour $\hat{\mathbf{x}}^*$ and its corresponding length $\hat{y}^*$.

2.3.3 Iterated 2-opt procedure

This phase implements the iterated 2-opt algorithm to improve $\hat{\mathbf{x}}^*$ from Section 2.3.2.

Input: Estimated optimal tour $\hat{\mathbf{x}}^*$.

Step 1 Start with $\hat{\mathbf{x}}^*$ and its first edge.

Step 2 Select an edge (*a*,*b*) and search for another edge (*c*,*d*), and then remove them to break the tour into two paths.

Step 3 Calculate the sum of distances between these two edges and the sum of distances between edge (*a*,*c*) and edge (*b*,*d*).

Copyright by Kasetsart University All rights reserved

Step 4 Reconnect the tour by these modified edges, only if the sum of distances between the two modified edges is less than that of distances between the two removed edges.

Step 5 Set the obtained tour as an initial tour, and repeat Steps 2 and 3 with another edges until no improvement (reaching a local optimum).

Step 6 Keep the best tour, and set it as an initial tour for next iteration. The pictorial framework of the 2-opt algorithm (Steps 1 through 6) can be seen in Figure 5.

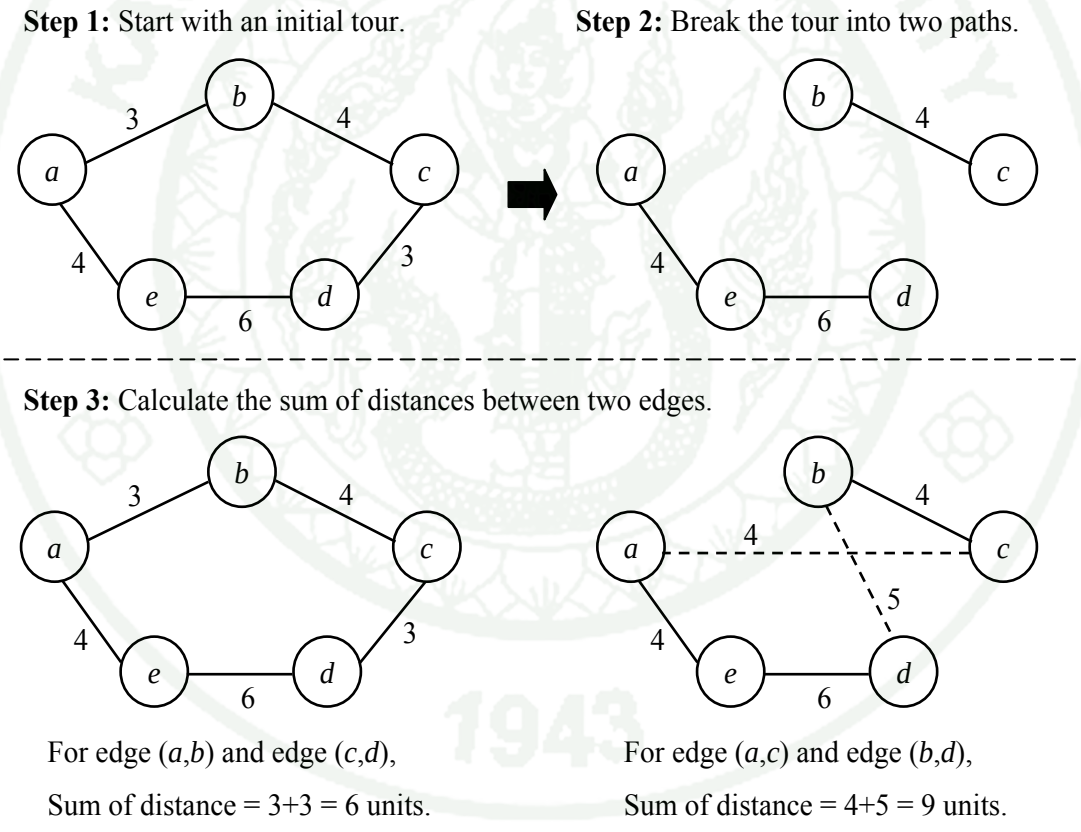


Figure 5 Pictorial framework of the 2-opt algorithm

Copyright by Kasetsart University All rights reserved

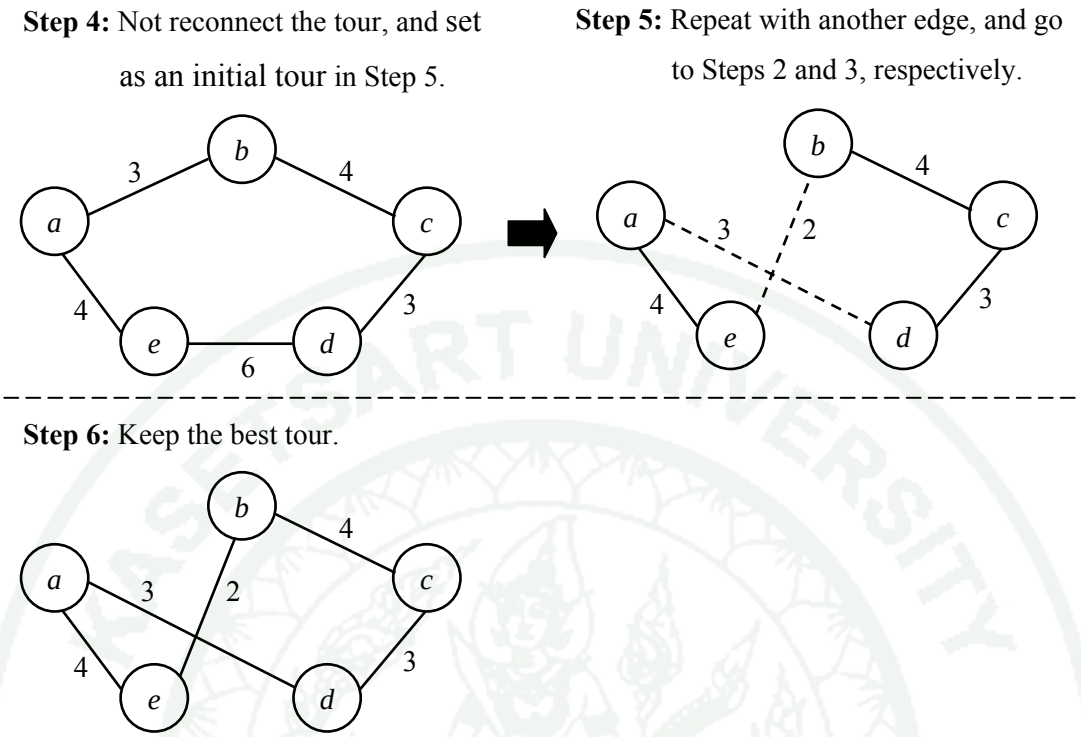


Figure 5 Pictorial framework of the 2-opt algorithm (Continued)

Step 7 Repeat Steps 2-6 until no improvement can be made in the two consecutive iterations, and return the resulting tour and its total distance.

Output: Resulting tour $\hat{\hat{\mathbf{x}}}^*$ and its corresponding length $\hat{\hat{y}}^*$.

The GPRTSP-0 algorithm stops if it cannot find better solution, so the estimated optimal solution may be stuck some local optimum. Consequently, the GPRTSP-0 algorithm may not converge to the globally optimal solutions. In order to escape the local optima, it is necessary to iterate a given local search many times, and to allow progressing on unimproved solutions until the solutions have departed from the trap, as in the iterated local search (Lourenço *et al.*, 2010).

The next section explains the improvement of our previous GPRTSP-0 algorithm. It extends a concept of the iterated local search (ILS) for improving better solution and proceeding with the hill climbing strategy in case of being stuck at local

Copyright by Kasetsart University All rights reserved

optima. This method is called the GPRTSP-1 algorithm. This thesis investigates the theoretical convergence properties of the GPRTSP-1 algorithm. With this provable algorithm, the simulated annealing (SA) is embedded as local search into ILS because SA is one of the provable local searches, which has provided the theoretical testimony in many literatures, e.g., Nolte and Schrader (2000), Johnson and Jacobson (2002a), Aarts *et al.* (2003), and Aarts *et al.* (2005).

2.4 GPRTSP-1 algorithm

The GPRTSP-1 algorithm for the symmetric TSP with a single salesman is divided into three phases: preparing the dataset, GPR prediction, and ISA procedure. The first two phases are the same as Sections 2.3.1 and 2.3.2 in the previous GPRTSP-0 algorithm; hence, the iterated simulated annealing (ISA) procedure is only explained in this section which is implemented to improve $\hat{\mathbf{x}}^*$, the output from Section 2.3.2. The procedure of the ISA algorithm can be seen below.

Input: Estimated optimal tour $\hat{\mathbf{x}}^*$ and its corresponding length $\hat{y}^*$

Procedure Iterated Simulated Annealing (ISA)

Initialization: $\hat{\mathbf{x}}^*$ (GPR solution), $y\left(\hat{\mathbf{x}}^*\right) \triangleq \hat{y}^*$ (Length of $\hat{\mathbf{x}}^*$), T_k (Initial temperature), β (Cooling rate), $u \in \mathbb{Z}^+$ (Micro-iteration number), U (Maximum number of u), $k \in \mathbb{Z}^+$ (Macro-iteration number), K (Maximum number of k), `MaxCount_`u (Stopping criterion when no improvement is found within some period of u iterations), `MaxCount_`k (Restarting criterion when no improvement is found within some period of k iterations), `MaxCount_Temp` (Maximum iteration number for changing temperature), `MaxCount_restart` (Stopping criterion when no improvement is found with multi-restart points).

Copyright by Kasetsart University All rights reserved

$\hat{\mathbf{x}}^*$, $y\left(\hat{\mathbf{x}}^*\right)$, $\texttt{Count_restart} = 0$, $\texttt{Count_}u = 0$, $\texttt{Count_}k = 0$,
$\texttt{Count_Temp} = 0$

Main procedure:

for $k = 1$ to K do

$\hat{\mathbf{x}}^*(k-1) \leftarrow \hat{\mathbf{x}}^*$, $y\left(\hat{\mathbf{x}}^*\right)_{k-1} \leftarrow y\left(\hat{\mathbf{x}}^*\right)$

$\hat{\mathbf{x}}^*_{best} \leftarrow \hat{\mathbf{x}}^*$, $y\left(\hat{\mathbf{x}}^*_{best}\right) \leftarrow y\left(\hat{\mathbf{x}}^*\right)$

for $u = 1$ to U do %% Start the simulated annealing loop

$y\left(\hat{\mathbf{x}}^*_{best}\right)_{u-1} \leftarrow y\left(\hat{\mathbf{x}}^*_{best}\right)$

Generate $\hat{\mathbf{x}}^{**}$ by random 2-exchange swapping of $\hat{\mathbf{x}}^*$

if $y\left(\hat{\mathbf{x}}^{**}\right) \le y\left(\hat{\mathbf{x}}^*\right)$ % Acceptance step

$\hat{\mathbf{x}}^* \leftarrow \hat{\mathbf{x}}^{**}$, $y\left(\hat{\mathbf{x}}^*\right) \leftarrow y\left(\hat{\mathbf{x}}^{**}\right)$

if $y\left(\hat{\mathbf{x}}^{**}\right) \le y\left(\hat{\mathbf{x}}^*_{best}\right)$

$\hat{\mathbf{x}}^*_{best} \leftarrow \hat{\mathbf{x}}^{**}$, $y\left(\hat{\mathbf{x}}^*_{best}\right) \leftarrow y\left(\hat{\mathbf{x}}^{**}\right)$

end

else

$RN_k \sim \text{Uniform}(0,1)$, if $RN_k < \exp\left(-\left[\dfrac{y\left(\hat{\mathbf{x}}^{**}\right) - y\left(\hat{\mathbf{x}}^*\right)}{T_k}\right]\right)$, then

$\hat{\mathbf{x}}^* \leftarrow \hat{\mathbf{x}}^{**}$, $y\left(\hat{\mathbf{x}}^*\right) \leftarrow y\left(\hat{\mathbf{x}}^{**}\right)$

end

if $y\left(\hat{\mathbf{x}}^*_{best}\right) = y\left(\hat{\mathbf{x}}^*_{best}\right)_{u-1}$

$\texttt{Count_}u = \texttt{Count_}u + 1$

else

$\texttt{Count_}u = 0$

end

Copyright by Kasetsart University All rights reserved

if $\texttt{Count_}u = \texttt{MaxCount_}u$

$\hat{\mathbf{x}}^* \leftarrow \hat{\mathbf{x}}^*_{best}$, $y\left(\hat{\mathbf{x}}^*\right) \leftarrow y\left(\hat{\mathbf{x}}^*_{best}\right)$

break % End searching in the micro-loop u

end

if $\texttt{Count_Temp} = \texttt{MaxCount_Temp}$

$T_k \leftarrow \beta T_k$

$\texttt{Count_Temp} = 0$

else

$\texttt{Count_Temp} = \texttt{Count_Temp} + 1$

end

$u \leftarrow u + 1$

end for %% End searching by the simulated annealing loop

if $y\left(\hat{\mathbf{x}}^*\right)_k \le y\left(\hat{\mathbf{x}}^*\right)_{k-1}$

$\hat{\mathbf{x}}^* \leftarrow \hat{\mathbf{x}}^*(k)$, $y\left(\hat{\mathbf{x}}^*\right) \leftarrow y\left(\hat{\mathbf{x}}^*\right)_k$

end

if $y\left(\hat{\mathbf{x}}^*\right)_k = y\left(\hat{\mathbf{x}}^*\right)_{k-1}$

$\texttt{Count_}k = \texttt{Count_}k + 1$

else

$\texttt{Count_}k = 0$

end

if $\texttt{Count_}k = \texttt{MaxCount_}k$

if $\texttt{Count_restart} = \texttt{MaxCount_restart}$

$\hat{\mathbf{x}}^* \leftarrow \hat{\mathbf{x}}^*(k)$, $y\left(\hat{\mathbf{x}}^*\right) \leftarrow y\left(\hat{\mathbf{x}}^*\right)_k$

break % End searching in the macro-loop k

else

$\hat{\mathbf{x}}^*_{2-opt} \leftarrow$ 2-opt algorithm$\left(\hat{\mathbf{x}}^*\right)$ % Restart after $k-$loop

$\hat{\mathbf{x}}^* \leftarrow \hat{\mathbf{x}}^*_{2-opt}$, $y\left(\hat{\mathbf{x}}^*\right) \leftarrow y\left(\hat{\mathbf{x}}^*_{2-opt}\right)$

Copyright by Kasetsart University All rights reserved

Count_restart = Count_restart + 1

Count_k = 0

end

end

$k \leftarrow k+1$

end for

Return: $\hat{\hat{\mathbf{x}}}^* \leftarrow \hat{\mathbf{x}}^*$ and $\hat{\hat{y}}^* \leftarrow y\left(\hat{\mathbf{x}}^*\right)$

Output: Resulting tour $\hat{\hat{\mathbf{x}}}^*$ and its corresponding length $\hat{\hat{y}}^*$.

Our two proposed algorithms (GPRTSP-0 and GPRTSP-1 algorithms) are presented in this section. In the next section, the theoretical properties of the GPRTSP-1 algorithm are analyzed through the Markov chain whose states are the solutions obtained, by utilizing the theoretical framework of Johnson and Jacobson (2002a). Then the proof show that it converges to one in the set of globally optimal solutions (in probability) as the number of iterations k approaches infinity.

3. Theoretical Properties of GPRTSP-1

This section provides the necessary and sufficient conditions of GPRTSP-1 algorithm, and shows that the GPRTSP-1 solution converges to the set of globally optimal solutions as the number of iterations approaches infinity.

3.1 Definition and notations

Define the TSP problem as a four-tuple $(\mathcal{I},\mathcal{S},\mathbf{x},y)$, where $\mathcal{I}$ is an instance of TSP, $\mathcal{S}$ is a finite set of all possible solutions which consists of a cardinality of $|\mathcal{S}|$ solutions, $\mathbf{x}$ is a TSP solution in $\mathcal{S}$, and $y:\mathcal{S}\rightarrow\mathbb{R}^+$ is a non-negative objective function value (length of legal tour). Define a neighborhood

Copyright by Kasetsart University All rights reserved

function correspond to each instance as $\mathcal{N}:\mathcal{S}\to 2^{\mathcal{S}}$, where $\mathcal{N}(\mathbf{x})\subseteq\mathcal{S}$ for all solutions $\mathbf{x}\in\mathcal{S}$. Let $\mathcal{G}$ be the set of globally optimal solutions according to $\mathcal{I}$, and $\mathcal{G}=\{\mathbf{x}^*\in\mathcal{S}:\ y(\mathbf{x}^*)\le y(\mathbf{x}),\forall\mathbf{x}\in\mathcal{S}\}$, where the objective function value $y^*\triangleq y(\mathbf{x}^*)$ is given by $y(\mathbf{x}^*)=\min_{\mathbf{x}\in\mathcal{S}}\{y(\mathbf{x})\}$. In addition, let $\hat{\mathbf{x}}^*$ be an initial solution for the ISA procedure that is computed by the GPR prediction (Section 2.3.2) or is the best solution obtained in each ISA loop (Section 2.4). Let $\hat{\mathbf{x}}^{**}$ be a candidate solution generated from $\hat{\mathbf{x}}^*$ with a suitable neighborhood function (Property 1 below). Moreover, the change in objective function values between two solutions is defined as

$$\Delta_{\hat{\mathbf{x}}^*,\hat{\mathbf{x}}^{**}}=y(\hat{\mathbf{x}}^{**})-y(\hat{\mathbf{x}}^*). \tag{11}$$

Property 1 Let the 2-exchange neighborhood function $(\mathcal{N}_2)$ be a neighborhood function used in ISA. For every solution $\mathbf{x}\in\mathcal{S}$, there exists some neighboring solutions generated by the 2-exchange neighborhood function $(\mathcal{N}_2(\mathbf{x}))$ in which those solutions contain a set of all neighbors of solution $\mathbf{x}$, $\mathcal{N}_2(\mathbf{x})\subset\mathcal{S}$. Consequently, this set is finite, and each solution in $\mathcal{N}_2(\mathbf{x})$ is accessible from solution $\mathbf{x}$ in a single transition.

3.2 Convergence analysis of the GPRTSP-1 algorithm

This subsection proposes a convergence proof of the GPRTSP-1 algorithm by utilizing the theory of discrete-time Markov chains. The Markov chain model used to analyze the optimality of the GPRTSP-1 algorithm is a homogeneous Markov chain model, whose size depends on the number of iteration k; that is, the Markov chain converges to an optimal solution with a stationary distribution $\varpi(k)$ as $k\to\infty$ (Johnson and Jacobson, 2002a; Aarts *et al.*, 2003). Therefore, the solution in each loop of our proposed algorithm is viewed as a Markov chain model based on

Copyright by Kasetsart University All rights reserved

convergence theories and conditions of a class of generalized hill climbing algorithms, discussed by Johnson and Jacobson (2002a).

Assume that the chain state is associated with the next accepted solution $\Psi(k) = j$ which is executed on ISA procedure with a current solution $\Psi(k-1) = i$. Let $P_{i,j}(k)$ be the transition probability that the solution j $\left(j = \hat{\mathbf{x}}^{**}\right)$ at iteration k, given the solution i $\left(i = \hat{\mathbf{x}}^{*}\right)$ at iteration $k-1$, where $k \in \mathbb{Z}^{+}$, that is

$$P_{i,j}(k) = \Pr\left\{\Psi(k) = j \middle| \Psi(k-1) = i\right\}, \quad \forall i, j \in \mathcal{S}.$$

Let $\zeta_{i,j}(k)$ be the generation probability that the neighboring solution j is generated from solution i at iteration k, and let $\mathcal{A}_{i,j}$ be the probability of accepting the neighboring solution j, $j \in \mathcal{N}_2(i)$. Because of independence between generation and acceptance, the probability that the chain will make a next transition into state $\Psi(k)$ is the product of the probability of generating a solution and the probability of acceptance. Thus for every iteration $k \in \mathbb{Z}^{+}$,

$$P_{i,j}(k) = \begin{cases} \zeta_{i,j}(k)\mathcal{A}_{i,j} & \text{for all } i \in \mathcal{S},\ j \in \mathcal{N}_2(i), i \neq j \\ 1 - \sum_{\substack{m \in \mathcal{N}_2(i) \\ m \neq i}} P_{i,m}(k) & i = j \end{cases}, \qquad (12)$$

where $P_{i,j}(k)$ is the transition probabilities with entries $i, j \in \mathcal{S}$, and they are formed as the transition matrix $\boldsymbol{P}(k)$ with $\sum_{j \in \mathcal{S}} P_{i,j}(k) = 1$ for all $i \in \mathcal{S}$. Note that in (12), the solution j is generated uniformly within the set of its neighbor solutions, i.e., $j \in \mathcal{N}_2(i)$, as specified in Property 2.

Copyright by Kasetsart University All rights reserved

Property 2 Assume a neighboring solution is generated uniformly by a given generating function $\zeta : \mathcal{S} \times \mathcal{S} \to [0,1]$. For each solution $i, j \in \mathcal{S}$, the value of the generation probability $\zeta_{i,j}$ equals to $\left|\mathcal{N}_2(i)\right|^{-1}$ if $j \in \mathcal{N}_2(i)$, and 0 otherwise.

For iteration k of the ISA procedure, the probability of generating the neighboring solution j given the solution i is defined by

$$\zeta_{i,j}(k) = \begin{cases} \left|\mathcal{N}_2(i)\right|^{-1} & j \in \mathcal{N}_2(i) \\ 0 & j \notin \mathcal{N}_2(i) \end{cases}. \tag{13}$$

Therefore, the generating probabilities $\zeta_{i,j}(k)$ satisfy $\sum_{j \in \mathcal{N}_2(i)} \zeta_{i,j}(k) = 1$.

The probability of accepting the neighboring solution j to be a solution in the next iteration $\left(\mathcal{A}_{i,j}\right)$ is defined by

$$\mathcal{A}_{i,j} = \begin{cases} 1 & \text{if } y\left(\hat{\mathbf{x}}^{**}\right) \le y\left(\hat{\mathbf{x}}^{*}\right) \\ \Pr\left\{RN_k(i,j) \ge \Delta_{i,j}\right\} & \text{otherwise} \end{cases}, \tag{14}$$

in which $RN_k(i,j)$ is a random variable distributed uniformly in the interval $(0,1)$ or $RN_k(i,j) \sim UNIF(0,1)$, and $\Delta_{i,j}$ is the change in tour-length. In addition, this algorithm uses the acceptance probability form as an exponential random variable. Thus in (14) and the event that $y\left(\hat{\mathbf{x}}^{**}\right) > y\left(\hat{\mathbf{x}}^{*}\right)$, the acceptance probability can be expressed as

Copyright by Kasetsart University All rights reserved

$$\Pr\left\{RN_k\left(i,j\right)\geq\Delta_{i,j}\right\}=\frac{\Pr\left\{RN_k\left(\mathbf{x}^*,j\right)\geq\Delta_{\mathbf{x}^*,j}\right\}}{\Pr\left\{RN_k\left(\mathbf{x}^*,i\right)\geq\Delta_{\mathbf{x}^*,i}\right\}}$$

$$=\frac{\exp\left(-\Delta_{\mathbf{x}^*,j}/T_k\right)}{\exp\left(-\Delta_{\mathbf{x}^*,i}/T_k\right)}$$

$$\equiv\exp\left(-\Delta_{i,j}/T_k\right), \tag{15}$$

where $RN_k\left(\mathbf{x}^*,i\right)$ and $RN_k\left(\mathbf{x}^*,j\right)$ are random variables, and T_k is the temperature at iteration k. Consequently, the acceptance probability in (14) can be rewritten as

$$\mathcal{A}_{i,j}=\min\left\{1,\exp\left(-\Delta_{i,j}/T_k\right)\right\}. \tag{16}$$

Note that the temperature decreases with a static cooling schedule, i.e., $\mathcal{M}_u\left(T_k,k,\beta\right)=\beta T_k$, where β is the cooling rate which is generally set between 0.8 to 0.99 (Aarts *et al.*, 2005).

Our convergence proof states that our GPRTSP-1 algorithm converges almost surely to one of the set of globally optimal solutions as k approaches infinity, as follow:

Theorem 1 The GPRTSP-1 algorithm converges almost surely to one of the set of globally optimal solutions as the iteration k approaches infinity; that is, the (tour) solution $\hat{\hat{\mathbf{x}}}^*\triangleq\hat{\mathbf{x}}^*_{best}\left(k\right)$ obtained from the GPRTSP-1 algorithm belongs to the set of globally optimal solutions almost surely as $k\to\infty$.

The sufficient conditions of our GPRTSP-1 algorithm are provided for the existence of a stationary distribution ϖ for each iteration k, based on Johnson and Jacobson (2002a), Aarts *et al.* (2003), and Aarts *et al.* (2005). Corollary 1 is

Copyright by Kasetsart University All rights reserved

important for the existence of a stationary distribution that is based on the result of Johnson and Jacobson (2002a), as can be seen in the Theorem A1 in the Appendix A.

Corollary 1 Consider a homogeneous Markov chain defined by a transition matrix $\boldsymbol{P}(k)$, $k \in \mathbb{Z}^+$ fixed, where $P_{i,j}(k)$ is the probability of moving from state i to state j in one step, and $P_{i,j}^h(k)$ is the probability of reaching state j in h steps, from state i. If the Markov chain is irreducible and aperiodic with finitely many states, then $\lim_{h\to\infty} P_{i,j}^h(k) = \varpi_j(k)$ exists for all solutions $i, j \in \mathcal{S}$. Moreover, $\varpi_j(k)$ is the unique strictly positive stationary distribution, defined as

$$\varpi_j(k) = \sum_{i\in\mathcal{S}} \varpi_i(k) P_{i,j}(k), \quad j \in \mathcal{S},$$

and

$$\sum_{j\in\mathcal{S}} \varpi_j(k) = 1.$$

Proof. Irreducibility implies that for every states in each iteration k must be accessible from every other states in finite number of transitions, and for all state $i, j \in \mathcal{S}$, there exists a positive transition probability $P_{i,j}(k)$ for all iterations k, i.e., $P_{i,j}(k) > 0$ (Ross, 2007). The one-step transition probability matrix $\boldsymbol{P}(k)$ reaches almost surely to the equilibrium distribution $\varpi(k)$ when it is executed with sufficiently large number of iteration u, since simulated annealing algorithm is done until reaching some locally optimal solution.

Let the h-step transition probabilities $P_{i,j}^h(k)$ be the probability that the search will be in state (solution) j, after h transition moves. For all the states, they must communicate with each other and $P_{i,j}^h(k) > 0$; then $\lim_{h\to\infty} P_{i,j}^h(k) = \varpi_j(k)$ and they satisfy $\sum_{j\in\mathcal{S}} \varpi_j(k) = 1$ (Ross, 2007). On the other hand, in most cases, aperiodicity

Copyright by Kasetsart University All rights reserved

means that all states in a chain have period 1 (Durrett, 2012). This satisfies if there exists $P_{i,i}(k) > 0$ for all k, then state $i \in \mathcal{S}$ has period 1. □

Corollary 2 proves the accessibility of all solutions with a non-zero value of generating probability.

Corollary 2 For all solutions $i, j \in \mathcal{S}$ and all macro iteration $k \in \mathbb{Z}^+$, there exists an integer $u \geq 1$, $u \in U$, and a corresponding sequence of solutions $\hat{\mathbf{x}}_0^{**}, \hat{\mathbf{x}}_1^{**}, \hat{\mathbf{x}}_2^{**}, ..., \hat{\mathbf{x}}_{U-1}^{**}$, $\hat{\mathbf{x}}_U^{**} \in \mathcal{S}$ with $\hat{\mathbf{x}}_0^{**} = \hat{\mathbf{x}}^*$, $\hat{\mathbf{x}}_U^{**} = \hat{\mathbf{x}}_{best}^*$ in which $\hat{\mathbf{x}}_{u+1}^{**} \in \mathcal{N}_2(\hat{\mathbf{x}}_u^{**})$ and $\zeta_{\hat{\mathbf{x}}_u^{**}, \hat{\mathbf{x}}_{u+1}^{**}}(k) > 0$, $u = 0,1,2,...,U-1$.

Proof. Let $i, j \in \mathcal{S}$ and iteration k. Define a solution i consists of a sequence of n_c cities. Let $\mathcal{N}_2$ be the 2-exchange neighborhood function (Property 1), and the neighboring solution j is constructed with $\mathcal{N}_2$ by randomly selecting two cities within the solution i and swapping their position in sequence (Property 2). Now it shows that the generating probability of the neighboring solution j with $\mathcal{N}_2$ is positive. For any $\hat{\mathbf{x}}_u^{**}$ be an initial solution with $u = 0$ (i.e., $\hat{\mathbf{x}}_0^{**}$), the generating probability of the neighboring solution of $\hat{\mathbf{x}}_0^{**}$ is computed as

$$u = 0,\ \hat{\mathbf{x}}_1^{**} \in \mathcal{N}_2(\hat{\mathbf{x}}_0^{**}) \qquad \zeta_{\hat{\mathbf{x}}_0^{**}, \hat{\mathbf{x}}_1^{**}}(k) = \frac{1}{\binom{n_c}{2}} > 0.$$

With $u \geq 1$ and $u \in U$, there is a sequence of possible solutions $\hat{\mathbf{x}}_0^{**}, \hat{\mathbf{x}}_1^{**}, \hat{\mathbf{x}}_2^{**}, ..., \hat{\mathbf{x}}_U^{**} \in \mathcal{S}$ such that the probabilities of generating each neighboring solution are positive; that is,

$$u = 1,\ \hat{\mathbf{x}}_2^{**} \in \mathcal{N}_2(\hat{\mathbf{x}}_1^{**}) \qquad \zeta_{\hat{\mathbf{x}}_1^{**}, \hat{\mathbf{x}}_2^{**}}(k) = \frac{1}{\binom{n_c}{2}} > 0$$

Copyright by Kasetsart University All rights reserved

$$u=2,\ \hat{\mathbf{x}}_3^{**}\in\mathcal{N}_2\left(\hat{\mathbf{x}}_2^{**}\right) \qquad \zeta_{\hat{\mathbf{x}}_2^{**},\hat{\mathbf{x}}_3^{**}}(k)=\frac{1}{\binom{n_c}{2}} \qquad >0$$

$$\vdots \qquad\qquad \vdots$$

$$u=U-2,\ \hat{\mathbf{x}}_{U-1}^{**}\in\mathcal{N}_2\left(\hat{\mathbf{x}}_{U-2}^{**}\right) \qquad \zeta_{\hat{\mathbf{x}}_{U-2}^{**},\hat{\mathbf{x}}_{U-1}^{**}}(k)=\frac{1}{\binom{n_c}{2}} \qquad >0$$

$$u=U-1,\ \hat{\mathbf{x}}_U^{**}\in\mathcal{N}_2\left(\hat{\mathbf{x}}_{U-1}^{**}\right) \qquad \zeta_{\hat{\mathbf{x}}_{U-1}^{**},\hat{\mathbf{x}}_U^{**}}(k)=\frac{1}{\binom{n_c}{2}} \qquad >0.$$

Without loss of generality, assume that the solution $\hat{\mathbf{x}}_0^{**}=\hat{\mathbf{x}}^*$ and $\hat{\mathbf{x}}_U^{**}=\hat{\mathbf{x}}_{best}^*$ (i.e., $\hat{\mathbf{x}}^*=i$, $\hat{\mathbf{x}}_{best}^*=j$), there exists a corresponding sequence of possible solutions $\hat{\mathbf{x}}_0^{**},\hat{\mathbf{x}}_1^{**},\hat{\mathbf{x}}_2^{**},...,\hat{\mathbf{x}}_U^{**}\in\mathcal{S}$ in which $\zeta_{\hat{\mathbf{x}}_u^{**},\hat{\mathbf{x}}_{u+1}^{**}}(k)>0$, $u=0,1,2,...,U-1$. Then the probability that the neighboring solution j generated from a given current solution i is given by

$$\zeta_{i,j}(k)=\zeta_{\hat{\mathbf{x}}_0^{**},\hat{\mathbf{x}}_U^{**}}(k)$$

$$=\zeta_{\hat{\mathbf{x}}_0^{**},\hat{\mathbf{x}}_1^{**}}(k)\,\zeta_{\hat{\mathbf{x}}_1^{**},\hat{\mathbf{x}}_2^{**}}(k)\,\zeta_{\hat{\mathbf{x}}_2^{**},\hat{\mathbf{x}}_3^{**}}(k)\cdots\zeta_{\hat{\mathbf{x}}_{U-2}^{**},\hat{\mathbf{x}}_{U-1}^{**}}(k)\,\zeta_{\hat{\mathbf{x}}_{U-1}^{**},\hat{\mathbf{x}}_U^{**}}(k)$$

By the Lemma A1 in the Appendix A and the multiplication rule for two independent events, therefore

$$\zeta_{i,j}(k)=\frac{1}{\binom{n_c}{2}}\cdot\frac{1}{\binom{n_c}{2}}\cdot\frac{1}{\binom{n_c}{2}}\cdots\frac{1}{\binom{n_c}{2}}\cdot\frac{1}{\binom{n_c}{2}} \quad >0.$$

Copyright by Kasetsart University All rights reserved

Since $\zeta_{i,j}(k) > 0$; thus the neighboring solution j is accessible from a given solution i, which completes the proof. □

Corollary 3 provides the homogeneous Markov chain theory on the existence of a stationary distribution $\varpi(k)$ for all finite iteration $k \in \mathbb{Z}^+$ that is based on the result of Johnson and Jacobson (2002a), as can be seen in the Theorem A2 in the Appendix A.

Corollary 3 Let $\hat{\mathbf{x}}^*$ be a solution obtained by the GPR prediction, and $\mathcal{N}_2$ denotes the 2-exchange neighborhood function over solution space $\mathcal{S}$. For a sufficiently large iteration number k and fixed temperature $(T_k = T,\ T > 0)$, let the ISA transition probability $P_{i,j}(k)$ which is fixed at iteration $k = K$ be defined by (12), where the probability of generating the neighboring solution is specified in (13), and the probability of solution acceptance is specified in (14). If $P_{i,j}(k)$ satisfies the conditions:

(a) for all $i, j \in \mathcal{S}$ and all macro iterations $k \in \mathbb{Z}^+$, there exists an integer $u \geq 1$, $u \in U$, and a corresponding sequence of solutions $\hat{\mathbf{x}}_0^{**}, \hat{\mathbf{x}}_1^{**}, \hat{\mathbf{x}}_2^{**}, ..., \hat{\mathbf{x}}_U^{**} \in \mathcal{S}$ with $\hat{\mathbf{x}}_0^{**} = \hat{\mathbf{x}}^*$, $\hat{\mathbf{x}}_U^{**} = \hat{\mathbf{x}}_{best}^*$ (i.e., $\hat{\mathbf{x}}^* = i$, $\hat{\mathbf{x}}_{best}^* = j$) where $\hat{\mathbf{x}}_{u+1}^{**} \in \mathcal{N}_2(\hat{\mathbf{x}}_u^{**})$ and $\zeta_{\hat{\mathbf{x}}_u^{**}, \hat{\mathbf{x}}_{u+1}^{**}}(k) > 0$, $u = 0,1,2,...,U-1$, defined in (13),

(b) for all $i, j \in \mathcal{S}$, the probability of generating the neighboring solution $\zeta_{i,j}(k)$ is independent of k, and $\zeta_{i,j}(k) = \zeta_{j,i}(k)$.

Furthermore, assume that the probability of solution acceptance satisfies:

(c) $\Pr\{RN_k(i,j) \geq \Delta_{i,j}\} > 0$ for all $i, j \in \mathcal{S}$ and all macro iterations $k \in \mathbb{Z}^+$.

Then there exists the stationary distribution $\varpi_i(T,k)$ for all solutions $i \in \mathcal{S}$ and for each macro iteration k, with fixed T, given by

Copyright by Kasetsart University All rights reserved

$$\varpi_i(T,k) = \frac{\Pr\left\{RN_k\left(\mathbf{x}^*, i\right) \geq \Delta_{\mathbf{x}^*,i}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{S}} \Pr\left\{RN_k\left(\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}. \quad (17)$$

Proof. The result of the proof of Theorem 2 in Johnson and Jacobson (2002a), pp. 370 – 371, is utilized. We first show that the Markov chain of the GPRTSP-1 solutions satisfy the irreducibility and aperiodicity conditions of Corollary 1, and then the transition probability in (12), the probability of generating solution in (13), and the probability of accepting the candidate solution in (14) are shown to satisfy the condition for time reversibility (Ross, 2007), defined as

$$\varpi_i(k) P_{i,j}(k) = \varpi_j(k) P_{j,i}(k) \text{ for all } i, j \in \mathcal{S}, \text{ and all iterations } k. \quad (18)$$

The irreducibility for the Markov chain of the GPRTSP-1 solutions can be shown as follows: From condition (a), the transition matrix corresponding to the probability of generating solution in (13) must assign a move with positive probability between any two states (solutions) $i, j \in \mathcal{S}$, which is addressed a proof in Corollary 2. Moreover, condition (c) requires that the probability of accepting the candidate solution in (14) for moving between all solutions is strictly positive for all finite iteration k. Therefore, conditions (a) and (c) altogether imply that, for all solutions, they communicate with each other; then this Markov chain is irreducible.

The aperiodicity for the Markov chain of the GPRTSP-1 solutions is to show that all states (solutions) in the chain have period 1, which is satisfied that $P_{i,i}(k) > 0$ exists for all iteration k. Let the solutions $i, j \in \mathcal{S}$ with $y(i) < y(j)$ if $\zeta_{i,j}(k) > 0$. By condition (a), all commutating pairs of solutions always exist in the solution space $\mathcal{S}$. Without loss of generality, assume that there exists at least one solution $\hat{\mathbf{x}}_u^{**} \in \mathcal{N}_2(i)$, $\Pr\left\{RN_k\left(i, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{i,\hat{\mathbf{x}}_u^{**}}\right\} < 1$. Note that, without this assumption, the transition occurred for some iteration k would be accepted with probability one, which represents the characteristic of local search; the algorithms based on hill

Copyright by Kasetsart University All rights reserved

climbing search are designed to employ between these two benefits (Johnson and Jacobson, 2002a). Therefore, we have that

$$\begin{aligned} P_{i,i}(k) &= 1-\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S},\,\hat{\mathbf{x}}_u^{**}\neq i}\zeta_{i,\hat{\mathbf{x}}_u^{**}}\Pr\left\{RN_k\left(i,\hat{\mathbf{x}}_u^{**}\right)\geq\Delta_{i,\hat{\mathbf{x}}_u^{**}}\right\} \\ &= 1-\zeta_{i,j}\Pr\left\{RN_k(i,j)\geq\Delta_{i,j}\right\}-\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S},\,\hat{\mathbf{x}}_u^{**}\neq i,j}\zeta_{i,\hat{\mathbf{x}}_u^{**}}\Pr\left\{RN_k\left(i,\hat{\mathbf{x}}_u^{**}\right)\geq\Delta_{i,\hat{\mathbf{x}}_u^{**}}\right\} \\ &> 1-\zeta_{i,j}-\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S},\,\hat{\mathbf{x}}_u^{**}\neq i,j}\zeta_{i,\hat{\mathbf{x}}_u^{**}} \\ &= 1-\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S},\,\hat{\mathbf{x}}_u^{**}\neq i}\zeta_{i,\hat{\mathbf{x}}_u^{**}}=0, \end{aligned}$$

which is consistent with the criterion for aperiodicity.

Now we show that the Markov chain corresponding to Corollary 3 satisfy the condition for reversibility, by substituting (12) and (17) into (18), that is

$$\begin{aligned} &\frac{\Pr\left\{RN_k\left(\mathbf{x}^*,i\right)\geq\Delta_{\mathbf{x}^*,i}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S}}\Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}\right)\geq\Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}P_{i,j}(k) \\ &= \frac{\Pr\left\{RN_k\left(\mathbf{x}^*,j\right)\geq\Delta_{\mathbf{x}^*,j}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S}}\Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}\right)\geq\Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}P_{j,i}(k). \end{aligned} \tag{19}$$

Note that, in (19), the transition probability $P_{i,j}(k)$ is the product of the generation probability $\zeta_{i,j}(k)$ in (13) and the solution acceptance probability $\mathcal{A}_{i,j}$ in (14). The solution acceptance probability $\mathcal{A}_{i,j}$ relates to three cases of accepting the neighboring solution j, as follow:

- The function value of the neighboring solution j is more than that of the solution i.

Copyright by Kasetsart University All rights reserved

- The function values of the neighboring solution j and the solution i are the same.
- The function value of the neighboring solution j is less than that of the solution i.

Therefore, the reversibility in (19) must hold on these three cases. Considering any two states $i, j \in \mathcal{S}$, and $i \neq j$, the reversibility in (19) leads to examination under the following cases:

Case 1. If $\Pr\left\{RN_k\left(\mathbf{x}^*, j\right) \geq \Delta_{\mathbf{x}^*, j}\right\} < \Pr\left\{RN_k\left(\mathbf{x}^*, i\right) \geq \Delta_{\mathbf{x}^*, i}\right\}$, then the reversibility in (19) can be rewritten as

$$\frac{\Pr\left\{RN_k\left(\mathbf{x}^*, i\right) \geq \Delta_{\mathbf{x}^*, i}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{S}} \Pr\left\{RN_k\left(\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}}\right\}} \zeta_{i,j}(k) \mathcal{A}_{i,j}$$

$$= \frac{\Pr\left\{RN_k\left(\mathbf{x}^*, j\right) \geq \Delta_{\mathbf{x}^*, j}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{S}} \Pr\left\{RN_k\left(\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}}\right\}} \zeta_{j,i}(k)$$

$$\frac{\Pr\left\{RN_k\left(\mathbf{x}^*, i\right) \geq \Delta_{\mathbf{x}^*, i}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{S}} \Pr\left\{RN_k\left(\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}}\right\}} \zeta_{i,j}(k) \frac{\Pr\left\{RN_k\left(\mathbf{x}^*, j\right) \geq \Delta_{\mathbf{x}^*, j}\right\}}{\Pr\left\{RN_k\left(\mathbf{x}^*, i\right) \geq \Delta_{\mathbf{x}^*, i}\right\}}$$

$$= \frac{\Pr\left\{RN_k\left(\mathbf{x}^*, j\right) \geq \Delta_{\mathbf{x}^*, j}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{S}} \Pr\left\{RN_k\left(\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}}\right\}} \zeta_{j,i}(k).$$

By condition (b), $\zeta_{i,j}(k) = \zeta_{j,i}(k)$ and it is independent of iteration k, thus the reversibility condition in (19) holds.

Case 2. If $\Pr\left\{RN_k\left(\mathbf{x}^*, j\right) \geq \Delta_{\mathbf{x}^*, j}\right\} = \Pr\left\{RN_k\left(\mathbf{x}^*, i\right) \geq \Delta_{\mathbf{x}^*, i}\right\}$, then the reversibility in (19) can be rewritten as

Copyright by Kasetsart University All rights reserved

$$\frac{\Pr\left\{RN_k\left(\mathbf{x}^*,i\right)\geq\Delta_{\mathbf{x}^*,i}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S}}\Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}\right)\geq\Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}\zeta_{i,j}\left(k\right)=\frac{\Pr\left\{RN_k\left(\mathbf{x}^*,j\right)\geq\Delta_{\mathbf{x}^*,j}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S}}\Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}\right)\geq\Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}\zeta_{j,i}\left(k\right).$$

By condition (b), $\zeta_{i,j}\left(k\right)=\zeta_{j,i}\left(k\right)$ and it is independent of iteration k, thus the reversibility condition in (19) again holds.

Case 3. If $\Pr\left\{RN_k\left(\mathbf{x}^*,j\right)\geq\Delta_{\mathbf{x}^*,j}\right\}>\Pr\left\{RN_k\left(\mathbf{x}^*,i\right)\geq\Delta_{\mathbf{x}^*,i}\right\}$. By the minimum function of the probability of acceptance in (16), this case reduces to be the form in case 2.

According to three cases of the reversibility, there exists the stationary distribution $\varpi_i\left(T,k\right)$ in (17) for all solutions $i\in\mathcal{S}$ and for iteration k, thus completing the proof. □

The convergence guarantee of our GPRTSP-1 algorithm is proved when the macro iteration approaches infinity ($k\to\infty$) with the time-homogeneous Markov chains theory. Corollary 4 provides a sufficient condition for the probability of accepting the solution j when the simulated annealing temperature approaches zero as $k\to\infty$, which utilizes the results of Johnson and Jacobson (2002a) in the Theorem A3 in the Appendix A and the condition (A3) of Aarts *et al.* (2003) in the Theorem A4 in the Appendix A.

Corollary 4 Under the conditions and assumptions of Corollaries 2 and 3, for all $i,j\in\mathcal{S}$ and for all macro iteration $k\in\mathbb{Z}^+$, if the objective function value (length of tour) of solution i is less than that of solution j (i.e., $y\left(i\right)<y\left(j\right)$), and the temperature T approaches zero as $k\to\infty$, then the probability of accepting the solution j is

$$\lim_{T\downarrow 0}\lim_{k\to\infty}\Pr\left\{RN_k\left(i,j\right)\geq\Delta_{i,j}\right\}=0.$$

Copyright by Kasetsart University All rights reserved

Proof. Let $\hat{\mathbf{x}}^*$ and $\hat{\mathbf{x}}_u^{**}$ be the solutions, $\hat{\mathbf{x}}_u^{**} \in \mathcal{N}_2\left(\hat{\mathbf{x}}^*\right)$, by Corollary 2. Assume that the solution $\hat{\mathbf{x}}_u^{**}$ is worse than $\hat{\mathbf{x}}^*$, so the length of tours: $y\left(\hat{\mathbf{x}}^*\right) < y\left(\hat{\mathbf{x}}_u^{**}\right)$. Taking the limit of function as the annealing temperature T_k approaches zero and $k \to \infty$ in (15), that is,

$$\lim_{T_k \downarrow 0} \lim_{k \to \infty} \Pr\left\{RN_k\left(\hat{\mathbf{x}}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\hat{\mathbf{x}}^*, \hat{\mathbf{x}}_u^{**}}\right\} = \lim_{T_k \downarrow 0} \lim_{k \to \infty} \exp\left(-\Delta_{\hat{\mathbf{x}}^*, \hat{\mathbf{x}}_u^{**}} / T_k\right)$$

$$= \lim_{T_k \downarrow 0} \lim_{k \to \infty} \frac{1}{\exp\left(\Delta_{\hat{\mathbf{x}}^*, \hat{\mathbf{x}}_u^{**}} / T_k\right)}.$$

By assumption and the change in length of tours in (11), we have that $\Delta_{\hat{\mathbf{x}}^*, \hat{\mathbf{x}}_u^{**}} = y\left(\hat{\mathbf{x}}_u^{**}\right) - y\left(\hat{\mathbf{x}}^*\right) > 0$. Recall that, for any positive a, $\lim_{b \to 0^+} \frac{a}{b} = +\infty$, so

$$\lim_{T_k \downarrow 0} \lim_{k \to \infty} \frac{1}{\exp\left(\Delta_{\hat{\mathbf{x}}^*, \hat{\mathbf{x}}_u^{**}} / T_k\right)} = \frac{1}{(+\infty)} = 0.$$

Without loss of generality, assume that the solutions $\hat{\mathbf{x}}^* = i$, $\hat{\mathbf{x}}_u^{**} = j$, and $T_k = T$, we have that $\lim_{T \downarrow 0} \lim_{k \to \infty} \Pr\left\{RN_k(i, j) \geq \Delta_{i,j}\right\} = 0$. Consequently, the neighboring solution j is not accepted to be the next solution, thus completing the proof. □

Corollary 5 addresses the existence of a stationary distribution $\varpi(T,k)$ as the annealing temperature T approaches zero and $k \to \infty$ from the result of Johnson and Jacobson (2002a) in Theorem A3 in the Appendix A and the result of Aarts *et al.* (2005) in Corollary A1 in the Appendix A.

Corollary 5 Under the conditions and assumptions of Corollaries 2 through 4, let $\varpi_i(T,k)$ be the probability that the GPRTSP-1 algorithm can find the solution i after

Copyright by Kasetsart University All rights reserved

searching with infinite iterations, and the temperature T approaching zero. Then the unique stationary probability $\varpi(T,k)$ of Corollary 3 exists, and it can be expressed as

$$\varpi_i^* \triangleq \lim_{T\downarrow 0}\lim_{k\to\infty} \varpi_i(T,k) = |\mathcal{G}|^{-1}, \quad \text{if } i \in \mathcal{G},$$

and $\varpi_i = 0$ if $i \in \mathcal{S} \setminus \mathcal{G}$.

Proof. The proof of Corollary 5 relies on the results of the proof of Theorem A3 in the Appendix A (Johnson and Jacobson, 2002a) and the result of the proof of Corollary A1 in the Appendix A (Aarts *et al.*, 2005). The proof shows that the stationary distribution $\varpi(T,k)$ exists for the sequence of stationary distributions $\varpi_i(T,k)$ as the annealing temperature T approaches zero, and k approaches infinity. Let $\hat{\mathbf{x}}^*$ and $\hat{\mathbf{x}}_u^{**}$ be the solutions of the GPRTSP-1 algorithm, and $\hat{\mathbf{x}}_u^{**} \in \mathcal{N}_2(\hat{\mathbf{x}}^*)$. Taking the limit in both sides of (17), we have that

$$\lim_{T\downarrow 0}\lim_{k\to\infty} \varpi_{\hat{\mathbf{x}}^*}(T,k) = \lim_{T\downarrow 0}\lim_{k\to\infty} \frac{\Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}^*\right) \geq \Delta_{\mathbf{x}^*,\hat{\mathbf{x}}^*}\right\}}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{S}} \Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}. \tag{20}$$

For every solution $\hat{\mathbf{x}}^* \in \mathcal{G}$, the limit in (20) reduces to

$$\lim_{T\downarrow 0}\lim_{k\to\infty} \varpi_{\hat{\mathbf{x}}^*\in\mathcal{G}}(T,k) = \lim_{T\downarrow 0}\lim_{k\to\infty} \frac{1}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{G}} \Pr\left\{RN_k\left(\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*,\hat{\mathbf{x}}_u^{**}}\right\}}.$$

Let $\varpi_{\hat{\mathbf{x}}^*}^* \triangleq \lim_{T\downarrow 0}\lim_{k\to\infty} \varpi_{\hat{\mathbf{x}}^*\in\mathcal{G}}(T,k)$, so it becomes

$$\varpi_{\hat{\mathbf{x}}^*}^* = \lim_{T\downarrow 0}\lim_{k\to\infty} \frac{1}{\sum_{\hat{\mathbf{x}}_u^{**}\in\mathcal{G}}(1)} = \frac{1}{|\mathcal{G}|}. \tag{21}$$

Copyright by Kasetsart University All rights reserved

For all solutions $\hat{\mathbf{x}}^* \in \mathcal{S} \setminus \mathcal{G}$, under assumption of Corollary 4, the limit form in (20) reduces to

$$\lim_{T \downarrow 0} \lim_{k \to \infty} \varpi_{\hat{\mathbf{x}}^* \in \mathcal{S} \setminus \mathcal{G}}(T,k) = \lim_{T \downarrow 0} \lim_{k \to \infty} \frac{0}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{G}} \Pr\left\{ RN_k\left(\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}\right) \geq \Delta_{\mathbf{x}^*, \hat{\mathbf{x}}_u^{**}} \right\}}.$$

Let $\varpi_{\hat{\mathbf{x}}^*} \triangleq \lim_{T \downarrow 0} \lim_{k \to \infty} \varpi_{\hat{\mathbf{x}}^* \in \mathcal{S} \setminus \mathcal{G}}(T,k)$, therefore

$$\varpi_{\hat{\mathbf{x}}^*} = \lim_{T \downarrow 0} \lim_{k \to \infty} \frac{0}{\sum_{\hat{\mathbf{x}}_u^{**} \in \mathcal{G}} (1)} = 0 . \tag{22}$$

Without loss of generality, assume that the solutions $i = \hat{\mathbf{x}}^*$ and $j = \hat{\mathbf{x}}_u^{**}$, then the stationary distributions in (21) and (22) satisfy the assumption of Corollary 5. Consequently, the GPRTSP-1 algorithm finds the solution i (belonging to the set of globally optimal solutions) according to the existence of stationary distribution ϖ_i^*, thus completing the proof. □

In conclusion, the proof of Theorem 1 can be shown by using Corollaries 1 through 5. It guarantees that our proposed GPRTSP-1 algorithm converges to one of the globally optimal solutions in the limit. In other words, it reaches a TSP tour that belongs to the set of globally optimal solutions, thus completing the proof. □

The convergence analysis shows that the GPRTSP-1 is a provable approach for solving the TSP, guaranteeing that it converges in probability to the set of globally optimal solutions. The next section provides numerical results on empirical performance comparison.

Copyright by Kasetsart University. All rights reserved

4. Experimental Settings and Comparison

This section provides the brief descriptions of the eight methods that are used to compare with our proposed approaches. In addition, the parameter settings of numerical experiments and the performance measures are explained. The next subsection briefly explains the comparing methods that are used to compare the search performance with our two proposed approaches.

4.1 Comparing methods

The search performances of our two proposed fashions (GPRTSP-0 and GPRTSP-1 algorithms) are compared with two well-known methods: classical genetic algorithm (GA), the standard simulated annealing algorithm (SA), and other six recent methods. The following subsections give the concise descriptions of these methods:

4.1.1 Genetic algorithm

In this thesis, GA code in a MATLAB toolbox (Kirk, 2007) is utilized. The main parameters are population sizes (`pop_size`), probability of crossover (p_c), probability of mutation (p_m), and the stopping criteria on the number of iteration (Kirk, 2007).

4.1.2 Simulated annealing algorithm

In this thesis, SA code in a MATLAB toolbox (Seshadri, 2006) is applied. The main parameters are the initial temperature (T_{int}), end temperature (T_{end}), cooling rate (β), and the stopping criteria on the number of iteration (Seshadri, 2006).

Copyright by Kasetsart University All rights reserved

4.1.3 Other six recent algorithms

This subsection provides the description of other six recent algorithms: KNIES–TSP (Aras *et al.*, 1999), Co-Adaptive Net (Cochrane and Beasley, 2003), CONN (Saadatmand-Tarzjan *et al.*, 2007), GCGA (Yang *et al.*, 2008), Memetic SOM (Créput and Koukam, 2009), and Improved-EN (Liu *et al.*, 2009). The brief description of these methods is shown in Table 1.

Table 1 Description of other six recent algorithms

Abbreviation	Short description of method	Reference paper
KNIES–TSP	A self-organizing neural network: the Kohonen network incorporating explicit statistics (KNIES) for the TSP	Aras *et al.* (1999)
Co-Adaptive Net	The co-adaptive neural network approach	Cochrane and Beasley (2003)
CONN	The constructive-optimizer neural network	Saadatmand-Tarzjan *et al.* (2007)
GCGA	The generalized chromosome genetic algorithm (GCGA) with 2-opt local search	Yang *et al.* (2008)
Memetic SOM	A memetic neural network incorporating with the self-organizing map (SOM) algorithm	Créput and Koukam (2009)
Improved-EN	An improved elastic net method for the TSP	Liu *et al.* (2009)

4.2 Experiment settings and performance measure

Our proposed GPRTSP algorithms are built upon the GPR codes in the GPML toolbox (Rasmussen and Nickisch, 2010). All algorithms are implemented in MATLAB, and they are executed on 60 problem instances (ranging from 16 to 2,103 cities) from the TSP library (Reinelt, 1995). For the GPR prediction, the training dataset consists of 50 sample tours, and the initial hyperparameters are: $\ell = 2$, $\sigma_f^2 = 1$, and $\sigma_n^2 = 0$ (by trial and error). Because our algorithms need a training dataset

Copyright by Kasetsart University All rights reserved

(Section 2.3.1), it is created randomly. Both versions of the GPRTSP algorithms are repeated nine times on each TSP instance. Then total distances of tour are averaged. In addition, the parameters of the GA, SA, and ISA algorithms can be seen in Table 2.

Table 2 Parameter settings of all algorithms

Method	Parameters	Value	Source
GA	`pop_size`	100	Kirk (2007)
	p_c	0.5	
	p_m	0.8	
	Number of iteration	10,000	Trial and error
SA	T_{int}	1,000	Geng *et al.* (2011)
	T_{end}	0.0025	
	β	0.97	Seshadri (2006)
	Number of iteration	20,000	
ISA	T_k	1,000	Geng *et al.* (2011)
	β	0.97	Seshadri (2006)
	U	20,000	
	K	20,000	Trial and error
	`MaxCount_temp`	10	
	`MaxCount_`u	200	
	`MaxCount_`k	50	
	`MaxCount_restart`	10	

The performance of all algorithms measures the deviation of its solution (e.g., a solution $\hat{\hat{y}}^*$ achieved by the GPRTSP-0 algorithm in each trial) from an optimal solution (y^*) correspond to instance, given by

Copyright by Kasetsart University All rights reserved

$$D\ (\%)\ = \frac{\hat{\hat{y}}^{*} - y^{*}}{y^{*}} \times 100\,, \tag{23}$$

and the computer runtime is also considered.

This chapter explains our approaches: GPRTSP-0 and GPRTSP-1 algorithms, which combine the Gaussian process regression model with the improvement methods for solving the symmetric TSP. Moreover, the theoretical properties of our proposed GPRTSP-1 algorithm are discussed through the framework of Markov chain theory to show the convergence and optimality of the GPRTSP-1 algorithm. The next chapter provides the numerical results of all algorithms, and the comparisons of the search performance are discussed.

Copyright by Kasetsart University. All rights reserved

RESULTS AND DISCUSSION

This chapter illustrates the numerical results of the two versions of the GPRTSP algorithms with the parameter settings in Table 2. The result of the search performance of both proposed GPRTSP algorithms (correspond to each TSP instance) includes the best solution obtained from nine trials, the average solution $\hat{\hat{y}}^*$ of 9 trials, the 95% confidence interval, the average percentage of the deviation of its solution from the optimal solution (as defined in Equation (23)), and the average running time. The performance comparisons of these two algorithms are investigated in terms of both solution quality and computer runtime, in comparison to the genetic algorithm (GA) and the simulated annealing (SA) algorithm. The behavior of the GPRTSP-1 algorithm is also discussed. The performance comparison of solution quality with the running time limitation is demonstrated among three algorithms: GPRTSP-1, GA, and SA algorithms, and their results are compared with the other six recent algorithms.

1. Result of GPRTSP-0 Algorithm

The search performance of the proposed GPRTSP-0 algorithm are shown in Table 3. In addition, the solutions obtained and the running time (upon 9 trials) of the GPRTSP-0 algorithm can be seen in the Tables B1 and B5 in the Appendix B, respectively.

In Table 3, the percentages of the deviation are less than 12%, and the average running time is between 2 to 4,520 seconds. For the gr17 problem, this algorithm can find an optimal solution within 3 seconds. In addition, for the ulysses16, ulysses22, bays29, swiss42, pr107 and si535 instances, this algorithm can find solutions that are within 1% of the optimal solutions. However, the worst case of the deviation from an optimal solution occurs in the rat783 instance, approximately 12%. For the ulysses16, gr17, fri26, bays29, hk48, pr124, and pr144 test problems, the 95% confidence interval cannot be formed because the algorithm returns the same solutions for all 9 trials. Finally, for all 60 instances, the GPRTSP-0 algorithm achieves 5.26% of average percentage of the deviations from the optimal solutions.

Copyright by Kasetsart University All rights reserved

Table 3 The search performance of GPRTSP-0 algorithm

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
ulysses16	16	6,859	6,909	6,909.0	-	-	0.7	3.3
gr17	17	2,085	2,085	2,085.0	-	-	0.0	2.3
gr21	21	2,707	2,816	2,870.2	2,854.6	2,885.9	6.0	2.9
ulysses22	22	7,013	7,013	7,032.3	7,002.8	7,061.8	0.3	4.2
gr24	24	1,272	1,272	1,371.6	1,342.9	1,400.3	7.8	2.9
fri26	26	937	961	961.0	-	-	2.6	3.2
bayg29	29	1,610	1,660	1,666.2	1,663.5	1,668.9	3.5	3.8
bays29	29	2,020	2,035	2,035.0	-	-	0.7	3.6
dantzig42	42	699	736	739.0	736.8	741.2	5.7	6.3
swiss42	42	1,273	1,274	1,283.9	1,267.0	1,300.8	0.9	5.5
att48	48	10,628	10,954	11,142.2	11,060.2	11,224.2	4.8	6.9
gr48	48	5,046	5,286	5,359.2	5,324.2	5,394.2	6.2	10.8
hk48	48	11,461	12,003	12,003.0	-	-	4.7	6.9
eil51	51	426	428	432.4	429.2	435.7	1.5	7.3

Copyright by Kasetsart University All rights reserved

Table 3 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
berlin52	52	7,542	7,596	7,666.3	7,504.1	7,828.5	1.6	7.7
brazil58	58	25,395	25,699	25,947.0	25,875.5	26,018.5	2.2	8.7
st70	70	675	711	725.0	718.6	731.4	7.4	11.3
eil76	76	538	558	567.8	561.0	574.5	5.5	12.8
pr76	76	108,159	112,220	113,981.2	112,450.7	115,511.7	5.4	13.4
rat99	99	1,211	1,278	1,311.0	1,288.9	1,333.1	8.3	19.7
kroA100	100	21,282	21,768	21,956.8	21,834.1	22,079.4	3.2	19.2
kroB100	100	22,141	22,755	23,133.0	22,882.8	23,383.2	4.5	17.4
kroC100	100	20,749	22,005	22,013.9	22,011.3	22,016.5	6.1	18.3
kroD100	100	21,294	22,857	23,268.6	23,086.2	23,450.9	9.3	18.3
kroE100	100	22,068	22,596	22,729.0	22,531.1	22,926.9	3.0	18.7
rd100	100	7,910	8,311	8,596.8	8,388.4	8,805.2	8.7	18.9
lin105	105	14,379	14,984	15,070.7	15,008.8	15,132.5	4.8	20.2
pr107	107	44,303	44,573	44,666.9	44,553.6	44,780.1	0.8	19.4

Copyright by Kasetsart University All rights reserved

Table 3 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
gr120	120	6,942	7,305	7,469.8	7,372.9	7,566.7	7.6	24.7
pr124	124	59,030	61,746	61,746.0	-	-	4.6	23.7
bier127	127	118,282	121,772	124,480.8	122,390.8	126,570.7	5.2	24.8
ch130	130	6,110	6,513	6,630.2	6,544.7	6,715.7	8.5	31.2
pr136	136	96,772	104,093	105,196.3	104,454.0	105,938.7	8.7	29.6
pr144	144	58,537	60,754	60,754.0	-	-	3.8	29.1
ch150	150	6,528	6,759	6,772.3	6,764.6	6,780.0	3.7	37.7
kroA150	150	26,524	28,372	28,860.2	28,695.6	29,024.8	8.8	38.5
kroB150	150	26,130	27,484	27,688.8	27,622.2	27,755.3	6.0	37.3
pr152	152	73,682	75,774	75,970.3	75,810.2	76,130.5	3.1	32.1
u159	159	42,080	44,380	44,899.7	44,667.0	45,132.4	6.7	35.5
si175	175	21,407	21,656	21,671.0	21,665.2	21,676.8	1.2	40.2
d198	198	15,780	16,042	16,271.1	16,082.4	16,459.9	3.1	52.1
kroA200	200	29,368	30,457	30,689.1	30,559.9	30,818.3	4.5	61.3

Copyright by Kasetsart University All rights reserved

Table 3 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
kroB200	200	29,437	31,731	31,954.6	31,879.9	32,029.2	8.6	57.3
tsp225	225	3,916	4,137	4,176.1	4,158.7	4,193.6	6.6	85.5
ts225	225	126,643	130,742	132,839.0	132,197.2	133,480.8	4.9	55.1
pr226	226	80,369	83,184	84,415.9	83,937.8	84,894.0	5.0	61.1
gil262	262	2,378	2,580	2,623.0	2,606.0	2,640.0	10.3	95.2
a280	280	2,579	2,668	2,713.6	2,689.4	2,737.7	5.2	102.8
lin318	318	42,029	44,229	45,241.8	44,787.8	45,695.8	7.6	120.6
rd400	400	15,281	16,354	16,491.1	16,385.8	16,596.4	7.9	232.1
pcb442	442	50,778	53,799	54,363.2	54,100.8	54,625.7	7.1	223.9
d493	493	35,002	36,553	36,971.9	36,706.1	37,237.7	5.6	268.0
si535	535	48,450	48,771	48,865.3	48,827.1	48,903.5	0.9	400.6
pa561	561	2,763	2,923	2,946.8	2,933.6	2,959.9	6.7	338.2
d657	657	48,912	52,916	53,281.7	53,125.2	53,438.1	8.9	533.0
rat783	783	8,806	9,779	9,846.7	9,809.8	9,883.6	11.8	734.3

Copyright by Kasetsart University All rights reserved

Table 3 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
pr1002	1,002	259,045	276,226	279,077.1	277,476.5	280,677.8	7.7	1,426.0
d1291	1,291	50,801	53,924	54,545.3	54,066.6	55,024.0	7.4	1,924.3
fl1577	1,577	22,249	23,861	24,099.7	23,974.0	24,225.3	8.3	4,519.2
d2103	2,103	80,450	82,822	83,161.7	83,017.1	83,306.2	3.4	3,396.6
Average							5.26	256.09

Copyright by Kasetsart University All rights reserved

2. Results of GPRTSP-1 Algorithm

2.1 Behavior results of GPRTSP-1

This subsection provides the graphical results of behavior analysis on four TSP instances: ulysses16, gr21, pr107, and si535 problems, which are from TSP instances library (Reinelt 1995). The plots show the search progression (upon one trial): the tour length versus the number of macro iteration. Figure 6 shows that the GPRTSP-1 algorithm converges to optimal solution within a short period of time for the ulysses16 test problem (approximately 87 iterations). During the execution, the search progression is stuck at local optima for two times: the first time is at iteration number 19 (with 7,200 units of tour length), and the second one is at iteration number 59 (with 6,900 units of tour length). The search is stuck for a period of time, and then it escapes from local optima by restarting solution with the 2-opt algorithm to reach new better solution.

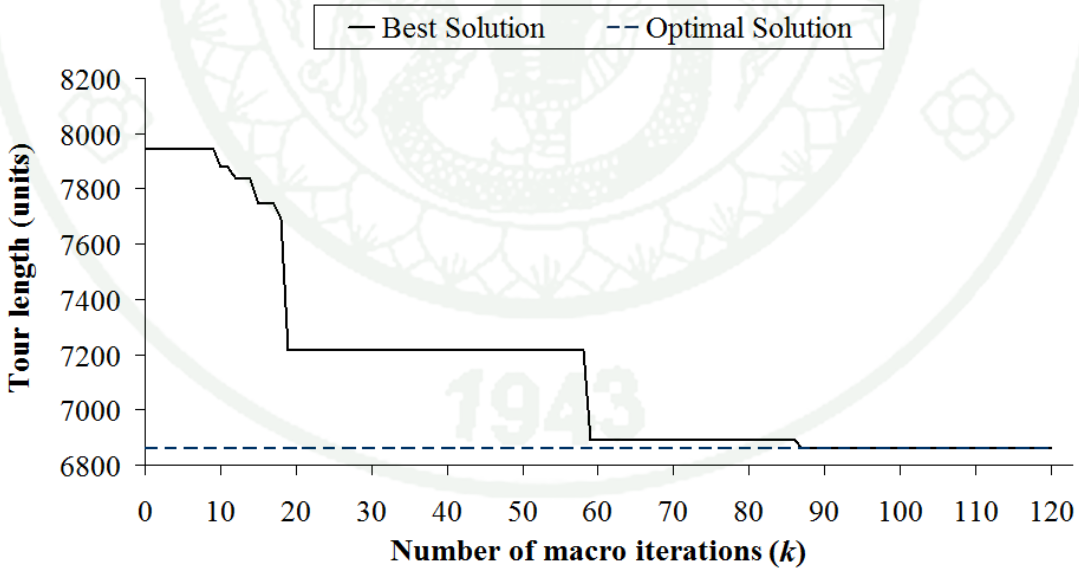


Figure 6 Behavior plot of ulysses16 test problem

For the gr21 test problem (Figure 7), the GPRTSP-1 algorithm converges to optimal solution within approximately 470 iterations. During the execution, the

Copyright by Kasetsart University All rights reserved

search progression is stuck at a local optimum after starting the search (with roughly 3,000 units of tour length), and then it escapes from local optimum at 150 iterations to reach a new better solution (with approximately 2,940 units of tour length). Then the search progression goes along the iterations with new better solutions until it reaches the optimal solution.

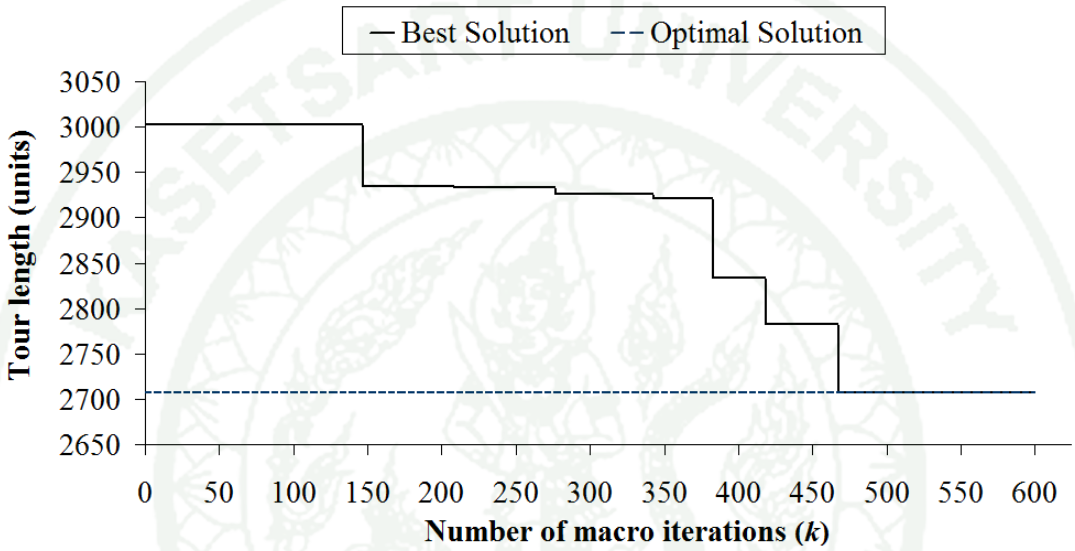


Figure 7 Behavior plot of gr21 test problem

In Figures 6 and 7, the proposed GPRTSP-1 algorithm converges to optimal solutions within a short period of time for the ulysses16 and gr21 problems while it takes a long time for the pr107 and si535 instances (shown in Figures 8 and 9, respectively). The estimated optimal solutions are improved, but the solutions do not belong to the sets of optimal solution.

For the pr107 test problem (Figures 8), the proposed GPRTSP-1 algorithm does not converge to optimal solution within the given 50,000 iterations. During the execution, the search progression is stuck at a local optimum at roughly 1,000 iterations (with roughly 44,600 units of tour length), and then the solution is kicked off to new better solution at approximately 33,000 iterations (with approximately 44,550 units of tour length). However, the solutions still sticks to a

Copyright by Kasetsart University All rights reserved

new local optimum until the iteration number reaches the maximum number (50,000 iterations).

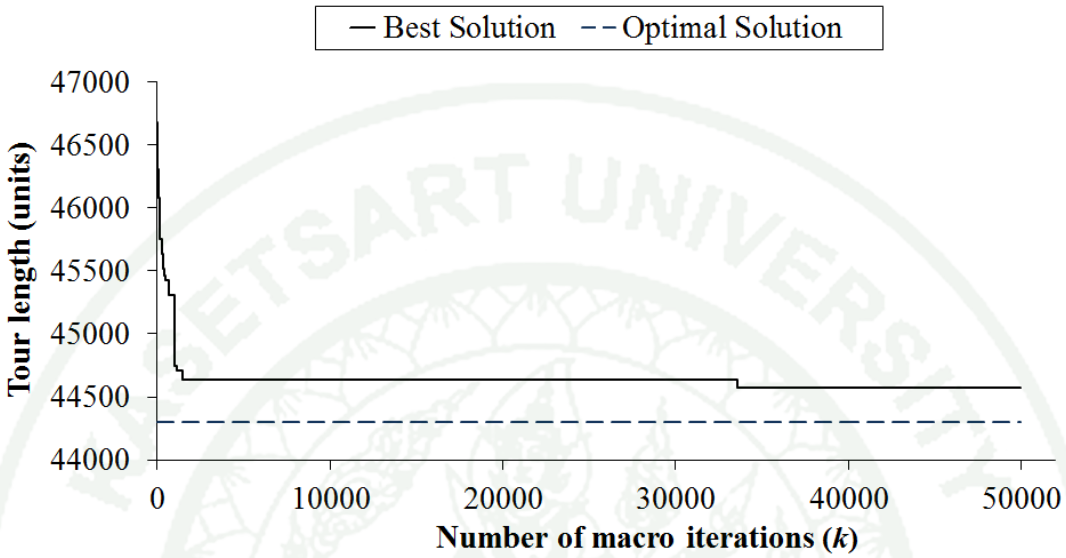


Figure 8 Behavior plot of pr107 test problem

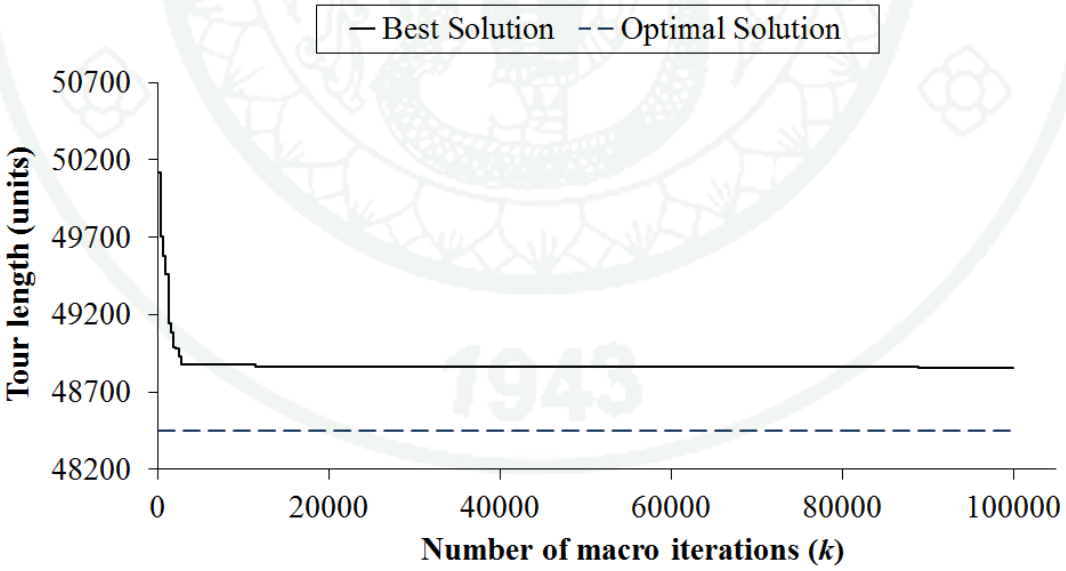


Figure 9 Behavior plot of si535 test problem

For the si535 test problem, in Figure 9, the proposed GPRTSP-1 algorithm does not converge to optimal solution within the given 100,000 iterations.

Copyright by Kasetsart University All rights reserved

During the execution, the search progression is stuck at a local optimum at roughly 12,000 iterations (with roughly 48,900 units of tour length) for a long period of iterations, and then the solution is kicked off to new slightly better solution at approximately 90,000 iterations (with approximately 48,850 units of tour length). However, the solutions are still stuck at this local optimum until the search progression reaches to maximum number of iterations (100,000 iterations).

For the pr107 and si535 test probems (in Figures 8 and 9), we speculate that the algorithm does not converge to the sets of optimal solution within the specified number of macro iterations for the following reasons:

- Problem size. The search space is the factorial function of the number of cities, and it takes more search time (Applegate *et al*. 2006).
- Some parameters of the ISA algorithm. They may not be suitable for these problems. It can probably be remedied by fine tuning parameters, for example, to allow searching for longer than the specified number of macro iterations, and to adjust the stopping criteria (as defined in Table 2).

In addition, the convergence rate of the GPRTSP-1 algorithm depends on the problem size, for instance, the convergence rate of GPRTSP-1 algorithm on the gr21 instance (Figure 7) is slowlier than that of the ulysses16 instance (Figure 6). Moreover, the convergence trends of the pr107 and si535 instances are slower than gr21 instance, as shown in Figures 8 and 9 respectively. In order to hit the set of globally optimal solutions, the proposed GPRTSP-1 algorithm should be further investigated on its behavior when a large number of iterations is available. In particular, for the pr107 and si535 test problems (Figures 8 and 9), the number of iteration must increase to reach the optimal solution. Since our experiments perform with the fixed values of the ISA parameters (Table 2), the solutions obtained and its convergence behaviors may not be good enough. Consequently, our proposed GPRTSP-1 algorithm is allowed to adjust the parameters for acquiring better ones.

Copyright by Kasetsart University All rights reserved

2.2 Numerical results of GPRTSP-1

The results of the proposed GPRTSP-1 algorithm are shown in Table 4. In addition, the solutions obtained and the running time (upon 9 trials) of the GPRTSP-1 algorithm can be seen in Tables B2 and B6 in the Appendix B, respectively. In Table 4, the percentages of the deviation (as defined in Equation (23)) are less than 12%, and the average running time is between 20 and 6,000 seconds. For the gr17 problem, the GPRTSP-1 algorithm can find an optimal solution within 18 seconds.

For the gr17, gr21, ulysses22, gr24, dantzig42, and swiss42 instances, it can find the optimal solutions in 9, 1, 2, 2, 1, and 1 trials, out of 9 trials (shown in Table B2 in the Appendix B), respectively. For the ulysses16, bays29, swiss42, eil51, pr107, si175, and si535 test problems, the algorithm can find average solutions to roughly within 1% of the optimal solutions. However, the worst case of the deviation from an optimal solution occurs in the rat783 instance at 12% of the optimal solution. For the gr17, bays29, hk48, and pr144 test problems, the 95% confidence interval cannot be formed because the algorithm returns the same solutions for all 9 trials. Note that, for the ulysses16 and gr21 instances, the algorithm quickly reaches the optimal solutions (Figures 6 and 7), but the algorithm still executes until it meets the stopping criterion; consequently, it takes more time than necessary to reach an optimal solution. Finally, for all 60 instances, the GPRTSP-1 algorithm achieves 5.15% of average percentage of the deviations from the optimal solutions.

Copyright by Kasetsart University All rights reserved

Table 4 The search performance of GPRTSP-1 algorithm

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
ulysses16	16	6,859	6,870	6,934.9	6,880.4	6,989.4	1.1	20.2
gr17	17	2,085	2,085	2,085.0	-	-	0.0	17.1
gr21	21	2,707	2,707	2,833.0	2,787.1	2,878.9	4.7	19.7
ulysses22	22	7,013	7,013	7,134.1	7,060.8	7,207.4	1.7	22.0
gr24	24	1,272	1,272	1,322.3	1,299.7	1,345.0	4.0	18.9
fri26	26	937	955	960.3	958.8	961.9	2.5	19.1
bayg29	29	1,610	1,628	1,663.7	1,638.0	1,689.3	3.3	21.2
bays29	29	2,020	2,035	2,035.0	-	-	0.7	20.8
dantzig42	42	699	699	731.2	717.5	745.0	4.6	25.7
swiss42	42	1,273	1,273	1,276.3	1,272.5	1,280.1	0.3	26.8
att48	48	10,628	10,918	11,045.6	10,952.2	11,138.9	3.9	30.4
gr48	48	5,046	5,286	5,324.3	5,289.4	5,359.3	5.5	36.6
hk48	48	11,461	12,003	12,003.0	-	-	4.7	29.5
eil51	51	426	428	431.1	428.2	434.0	1.2	28.3

Copyright by Kasetsart University All rights reserved

Table 4 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
berlin52	52	7,542	7,596	7,749.6	7,558.3	7,940.8	2.8	30.1
brazil58	58	25,395	25,978	26,129.9	25,956.3	26,303.5	2.9	36.0
st70	70	675	724	732.2	728.4	736.0	8.5	42.8
eil76	76	538	558	563.4	557.3	569.6	4.7	40.4
pr76	76	108,159	110,915	112,270.2	110,801.1	113,739.3	3.8	46.9
rat99	99	1,211	1,278	1,309.9	1,286.4	1,333.4	8.2	56.0
kroA100	100	21,282	21,572	22,197.6	21,895.3	22,499.9	4.3	58.5
kroB100	100	22,141	22,778	23,104.3	22,877.9	23,330.8	4.4	62.3
kroC100	100	20,749	21,613	21,970.3	21,867.3	22,073.3	5.9	57.1
kroD100	100	21,294	22,857	23,278.7	23,103.1	23,454.2	9.3	60.4
kroE100	100	22,068	22,570	22,760.3	22,525.7	22,995.0	3.1	57.9
rd100	100	7,910	8,253	8,447.8	8,275.9	8,619.7	6.8	60.1
lin105	105	14,379	14,837	15,082.6	14,931.5	15,233.6	4.9	63.9
pr107	107	44,303	44,573	44,716.9	44,598.8	44,835.0	0.9	67.3

Copyright by Kasetsart University All rights reserved

Table 4 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
gr120	120	6,942	7,296	7,437.4	7,337.2	7,537.7	7.1	67.8
pr124	124	59,030	61,396	62,131.4	61,597.8	62,665.1	5.3	77.0
bier127	127	118,282	121,576	123,756.7	122,408.7	125,104.6	4.6	84.0
ch130	130	6,110	6,499	6,601.1	6,517.7	6,684.5	8.0	83.6
pr136	136	96,772	105,005	106,213.0	105,714.0	106,712.0	9.8	91.9
pr144	144	58,537	60,754	60,754.0	-	-	3.8	94.2
ch150	150	6,528	6,759	6,768.0	6,759.8	6,776.2	3.7	99.0
kroA150	150	26,524	28,372	28,925.0	28,751.9	29,098.1	9.1	101.9
kroB150	150	26,130	27,078	27,707.4	27,453.7	27,961.1	6.0	101.5
pr152	152	73,682	75,402	75,890.0	75,648.3	76,131.7	3.0	102.6
u159	159	42,080	44,380	45,201.0	44,888.7	45,513.3	7.4	97.9
si175	175	21,407	21,648	21,668.2	21,661.5	21,674.9	1.2	108.9
d198	198	15,780	16,133	16,291.8	16,122.8	16,460.8	3.2	129.4
kroA200	200	29,368	30,477	30,716.9	30,592.3	30,841.4	4.6	132.0

Copyright by Kasetsart University All rights reserved

Table 4 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\hat{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
kroB200	200	29,437	31,294	31,642.2	31,409.7	31,874.7	7.5	133.4
tsp225	225	3,916	4,187	4,199.0	4,191.5	4,206.5	7.2	191.6
ts225	225	126,643	132,645	133,217.4	132,890.4	133,544.5	5.2	157.5
pr226	226	80,369	82,594	83,878.7	83,148.7	84,608.6	4.4	160.9
gil262	262	2,378	2,547	2,596.1	2,573.9	2,618.3	9.2	226.3
a280	280	2,579	2,681	2,716.3	2,700.9	2,731.8	5.3	247.9
lin318	318	42,029	44,464	45,102.2	44,700.0	45,504.4	7.3	277.7
rd400	400	15,281	16,354	16,562.2	16,476.2	16,648.3	8.4	395.5
pcb442	442	50,778	53,628	54,228.9	53,854.2	54,603.5	6.8	465.2
d493	493	35,002	36,553	37,053.1	36,808.5	37,297.7	5.9	560.2
si535	535	48,450	48,842	48,881.3	48,864.5	48,898.1	0.9	655.3
pa561	561	2,763	2,924	2,953.1	2,938.9	2,967.3	6.9	768.4
d657	657	48,912	52,916	53,354.0	53,171.8	53,536.2	9.1	950.9
rat783	783	8,806	9,799	9,860.0	9,832.0	9,888.0	12.0	1,278.5

Copyright by Kasetsart University All rights reserved

Table 4 (Continued)

Instances	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{\bar{y}}^*$ (units)	95% Confidence interval		Average Deviation $\bar{D}$ (%)	Average running time (seconds)
					Lower (units)	Upper (units)		
pr1002	1,002	259,045	275,087	277,343.7	276,146.9	278,540.4	7.1	1,464.1
d1291	1,291	50,801	53,776	54,498.7	54,169.2	54,828.2	7.3	2,358.3
fl1577	1,577	22,249	24,017	24,406.0	24,191.2	24,620.8	9.7	3,467.9
d2103	2,103	80,450	82,822	83,124.6	82,934.0	83,315.1	3.3	6,000.0
Average							5.15	370.12

Copyright by Kasetsart University All rights reserved

3. Performance Comparison of Algorithms

This section discusses three dimensions of the performance comparisons: the performance comparison of the GPRTSP-0, GPRTSP-1, GA, and SA algorithms, in term of solution quality, on 60 TSP instances, the performance comparison of the GPRTSP-0, GPRTSP-1, GA, and SA algorithms (on 60 TSP instances), in term of the running time, and the comparison of the proposed GPRTSP-1 algorithm with GA, SA, and other six recent algorithms (Table 1), in term of solution quality, on 30 common TSP instances. In addition, in the third comparison, the GPRTSP-1, GA, and SA algorithms are implemented with fixed period of runtime, i.e., the algorithm terminates when reaching the runtime as specified, and then the solution is collected. The summary results of the first two dimensions, including the average percentage of the deviation of their solutions from the optimal solutions and the average running time, can be seen in Table 5. The summary results of the third measure, including the average percentage of the deviation of their solutions from the optimal solutions (among nine methods), can be seen in Table 6.

3.1 Comparisons of solution quality

3.1.1 Comparison between the two proposed algorithms

Comparing with GPRTSP-0, the proposed GPRTSP-1 algorithm can find at least better solutions for 36 instances, out of 60 TSP instances (Table 5). For all 60 TSP instances, the GPRTSP-0 algorithm achieves 5.26% of the average deviation from the optimal solutions (Table 3) while the GPRTSP-1 algorithm achieves 5.15% of the average deviation from the optimal solutions (Table 4), which is decreased by approximately 2.1%. In addition, the performances on solution quality between our two proposed algorithms are slightly different (in comparison to each problem instance). The comparison plot of the deviation of our two proposed algorithms' solutions from optimal solutions is depicted in Figure 10.

Copyright by Kasetsart University All rights reserved

A possible reason is an aspect of two-exchange neighborhood function (Property 1) applied in the ISA procedure of GPRTSP-1 algorithm that all solutions in the subset of neighboring solutions can be generated uniformly to be a candidate, and they are accessible during iterations. When the solution of GPRTSP-1 algorithm is close to a solution deemed optimal, and the simulated annealing temperature is decreased to near zero, the worse generated solution will be discarded through the solution acceptance step. Then the GPRTSP-1 solution concentrates on the solution deemed optimal.

In addition, the neighborhood function applied in the iterated 2-opt procedure of GPRTSP-0 algorithm is the two-exchange neighborhood function (Helsgaun, 2009). Their average percentages of the deviation from optimal solutions (as shown in Table 5) are slightly different.

Copyright by Kasetsart University All rights reserved

Table 5 Comparison of the search performance of all four algorithms

Instances	# of nodes	Optimal solution (units)	Average Deviation $\bar{D}$ (%)				Average running time (seconds)			
			GPRTSP-0	GPRTSP-1	GA	SA	GPRTSP-0	GPRTSP-1	GA	SA
ulysses16	16	6,859	0.7	1.1	0.0	0.8	3.3	20.2	22.5	6.8
gr17	17	2,085	0.0	0.0	0.1	1.4	2.3	17.1	27.5	3.9
gr21	21	2,707	6.0	4.7	0.4	2.8	2.9	19.7	29.4	3.0
ulysses22	22	7,013	0.3	1.7	0.4	2.3	4.2	22.0	42.2	8.3
gr24	24	1,272	7.8	4.0	0.4	7.4	2.9	18.9	29.2	27.1
fri26	26	937	2.6	2.5	0.0	6.7	3.2	19.1	60.3	39.7
bayg29	29	1,610	3.5	3.3	0.8	5.9	3.8	21.2	63.6	12.3
bays29	29	2,020	0.7	0.7	0.4	5.5	3.6	20.8	63.6	12.2
dantzig42	42	699	5.7	4.6	0.0	15.9	6.3	25.7	31.1	157.4
swiss42	42	1,273	0.9	0.3	2.5	15.1	5.5	26.8	31.2	17.8
att48	48	10,628	4.8	3.9	1.7	16.8	6.9	30.4	32.4	21.7
gr48	48	5,046	6.2	5.5	0.8	21.6	10.8	36.6	32.8	20.8
hk48	48	11,461	4.7	4.7	2.4	17.9	6.9	29.5	32.4	21.9
eil51	51	426	1.5	1.2	3.8	10.0	7.3	28.3	33.4	356.0

Copyright by Kasetsart University All rights reserved

Table 5 (Continued)

Instances	# of nodes	Optimal solution (units)	Average Deviation $\bar{D}$ (%)				Average running time (seconds)			
			GPRTSP-0	GPRTSP-1	GA	SA	GPRTSP-0	GPRTSP-1	GA	SA
berlin52	52	7,542	1.6	2.8	3.5	22.9	7.7	30.1	33.3	23.3
brazil58	58	25,395	2.2	2.9	2.3	13.4	8.7	36.0	35.0	25.2
st70	70	675	7.4	8.5	4.1	26.1	11.3	42.8	37.8	585.0
eil76	76	538	5.5	4.7	5.0	18.5	12.8	40.4	39.4	803.1
pr76	76	108,159	5.4	3.8	9.0	16.4	13.4	46.9	39.2	36.5
rat99	99	1,211	8.3	8.2	7.1	41.6	19.7	56.0	45.8	45.2
kroA100	100	21,282	3.2	4.3	6.1	44.9	19.2	58.5	46.0	48.7
kroB100	100	22,141	4.5	4.4	6.4	37.9	17.4	62.3	45.5	48.2
kroC100	100	20,749	6.1	5.9	7.5	45.0	18.3	57.1	45.8	48.4
kroD100	100	21,294	9.3	9.3	5.2	42.1	18.3	60.4	45.8	49.6
kroE100	100	22,068	3.0	3.1	4.7	38.8	18.7	57.9	45.7	48.4
rd100	100	7,910	8.7	6.8	7.4	55.8	18.9	60.1	45.6	47.3
lin105	105	14,379	4.8	4.9	7.5	57.6	20.2	63.9	47.2	50.0
pr107	107	44,303	0.8	0.9	2.2	18.7	19.4	67.3	47.5	50.4

Copyright by Kasetsart University All rights reserved

Table 5 (Continued)

Instances	# of nodes	Optimal solution (units)	Average Deviation $\bar{D}$ (%)				Average running time (seconds)			
			GPRTSP-0	GPRTSP-1	GA	SA	GPRTSP-0	GPRTSP-1	GA	SA
gr120	120	6,942	7.6	7.1	6.6	56.1	24.7	67.8	51.3	55.8
pr124	124	59,030	4.6	5.3	4.3	19.8	23.7	77.0	52.5	63.6
bier127	127	118,282	5.2	4.6	6.2	17.0	24.8	84.0	53.4	64.5
ch130	130	6,110	8.5	8.0	9.1	60.3	31.2	83.6	54.7	61.4
pr136	136	96,772	8.7	9.8	8.4	16.3	29.6	91.9	56.3	69.4
pr144	144	58,537	3.8	3.8	5.3	8.9	29.1	94.2	58.8	77.1
ch150	150	6,528	3.7	3.7	9.6	73.3	37.7	99.0	61.1	76.5
kroA150	150	26,524	8.8	9.1	8.7	53.0	38.5	101.9	60.8	80.0
kroB150	150	26,130	6.0	6.0	7.6	55.3	37.3	101.5	60.8	79.9
pr152	152	73,682	3.1	3.0	5.3	15.7	32.1	102.6	61.3	79.2
u159	159	42,080	6.7	7.4	1.9	40.0	35.5	97.9	63.8	82.9
si175	175	21,407	1.2	1.2	1.0	20.2	40.2	108.9	69.8	79.7
d198	198	15,780	3.1	3.2	5.3	54.1	52.1	129.4	78.0	106.8
kroA200	200	29,368	4.5	4.6	10.0	61.1	61.3	132.0	78.8	109.5

Copyright by Kasetsart University All rights reserved

Table 5 (Continued)

Instances	# of nodes	Optimal solution (units)	Average Deviation $\bar{D}$ (%)				Average running time (seconds)			
			GPRTSP-0	GPRTSP-1	GA	SA	GPRTSP-0	GPRTSP-1	GA	SA
kroB200	200	29,437	8.6	7.5	10.4	63.7	57.3	133.4	79.0	109.6
tsp225	225	3,916	6.6	7.2	8.1	93.5	85.5	191.6	88.7	122.4
ts225	225	126,643	4.9	5.2	6.8	19.8	55.1	157.5	88.8	129.8
pr226	226	80,369	5.0	4.4	9.2	15.1	61.1	160.9	89.2	120.8
gil262	262	2,378	10.3	9.2	12.0	107.6	95.2	226.3	103.8	141.6
a280	280	2,579	5.2	5.3	6.8	122.7	102.8	247.9	111.2	152.0
lin318	318	42,029	7.6	7.3	12.1	69.4	120.6	277.7	129.0	208.9
rd400	400	15,281	7.9	8.4	21.3	126.8	232.1	395.5	172.1	259.1
pcb442	442	50,778	7.1	6.8	20.4	66.1	223.9	465.2	195.9	312.9
d493	493	35,002	5.6	5.9	16.7	60.9	268.0	560.2	224.4	351.2
si535	535	48,450	0.9	0.9	5.5	47.4	400.6	655.3	252.0	309.6
pa561	561	2,763	6.7	6.9	22.8	129.2	338.2	768.4	268.4	855.2
d657	657	48,912	8.9	9.1	41.4	78.1	533.0	950.9	337.6	527.7
rat783	783	8,806	11.8	12.0	90.6	158.3	734.3	1,278.5	437.6	561.3

Copyright by Kasetsart University All rights reserved

Table 5 (Continued)

Instances	# of nodes	Optimal solution (units)	Average Deviation $\bar{D}$ (%)				Average running time (seconds)			
			GPRTSP-0	GPRTSP-1	GA	SA	GPRTSP-0	GPRTSP-1	GA	SA
pr1002	1,002	259,045	7.7	7.1	19.4	34.9	1,426.0	1,464.1	632.9	1,185.2
d1291	1,291	50,801	7.4	7.3	100.6	118.6	1,924.3	2,358.3	962.6	1,469.1
fl1577	1,577	22,249	8.3	9.7	74.0	210.2	4,519.2	3,467.9	1,343.6	1,605.7
d2103	2,103	80,450	3.4	3.3	49.3	102.1	3,396.6	6,000.0	2,263.7	4,083.5
Average			5.26	5.15	11.81	46.42	256.09	370.12	162.89	270.17

Copyright by Kasetsart University All rights reserved

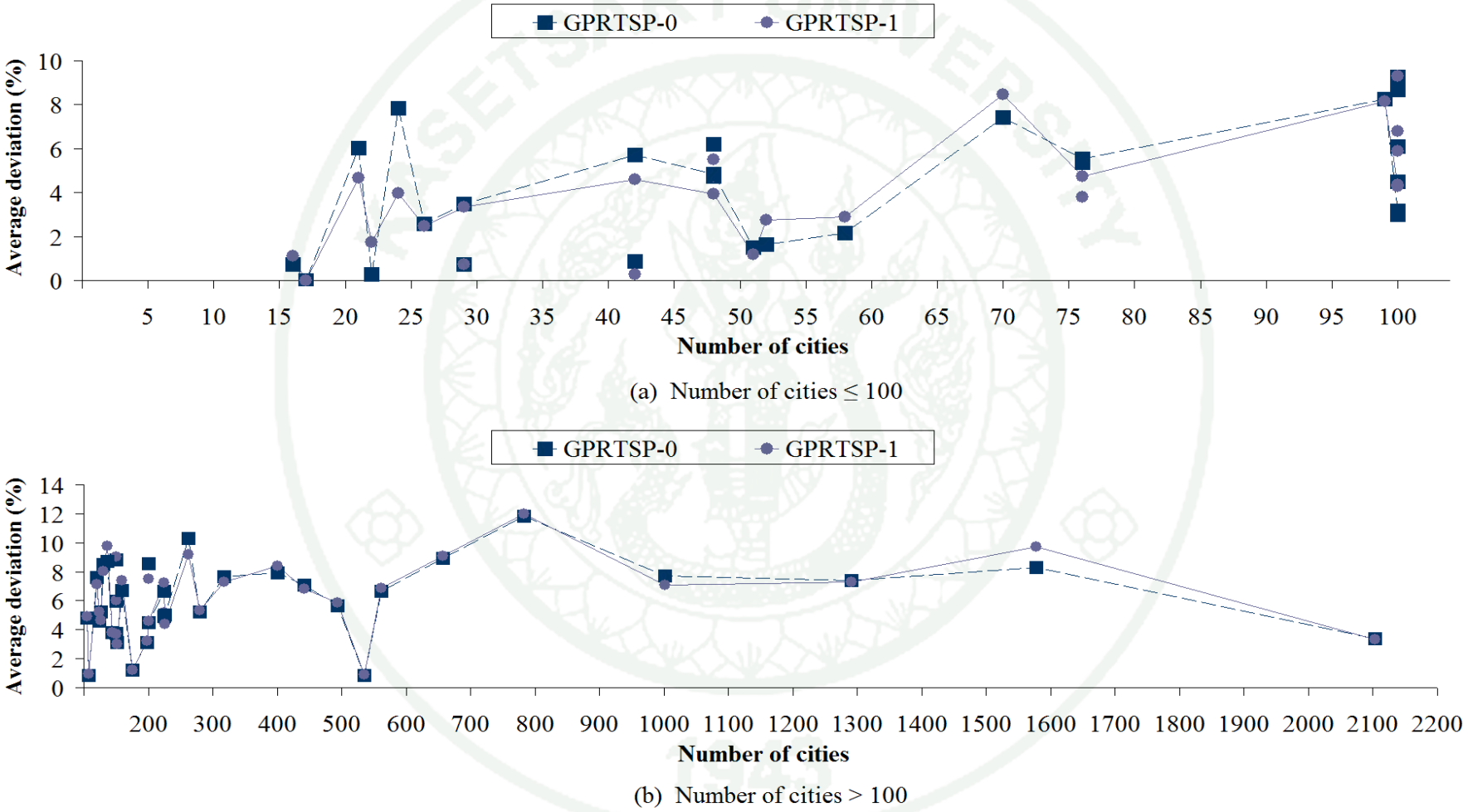


Figure 10 Comparisons of solution deviations between the two proposed algorithms

Copyright by Kasetsart University All rights reserved

3.1.2 Comparisons with genetic algorithm and simulated annealing algorithm

The percentages of average deviation of the solutions from the optimal solutions for the genetic algorithm (GA) and the simulated annealing (SA) algorithm are shown in Table 5. In addition, the solutions obtained (upon 9 trials) of GA and SA algorithms can be seen in Tables B3 and B4 in the Appendix B, respectively. In Table 5, both of our proposed methods (GPRTSP-0 and GPRTSP-1 algorithms) can find better solutions with the same number of TSP instances, i.e., for 39 instances (vs. GA) and for 58 instances (vs. SA), in comparison to GA and SA algorithms. Comparing with GA, the solutions quality of our two proposed methods are slightly different for the problem size up to 280 cities while those of GA are worse when a number of nodes are higher (as shown in Table 5).

On the overall, the SA algorithm yields extremely worse performance on solution quality than our two proposed algorithms and GA (as shown in Table 5). It is because of the stopping criterion that the SA algorithm terminates when reaching the maximum number of iterations. For all 60 test problems, our two proposed approaches achieve (roughly) two times and nine times less average percentage of average deviations from the optimal solutions than GA and SA algorithms (Table 5), respectively.

Figure 11 illustrates the comparison plot of the average deviation of all four algorithms' solutions from optimal solutions. This figure shows that both of our proposed methods perform well on 60 instances and accomplish similarly on solution quality, and they achieve remarkably better solution quality than GA and SA. However, our proposed GPRTSP-1 approach has a slightly better trend of percentage of average deviation than the previous one (Figure 10). When the size of the test instance is bigger, our two proposed approach still performs well while the performance of GA and SA deteriorates.

Copyright by Kasetsart University All rights reserved

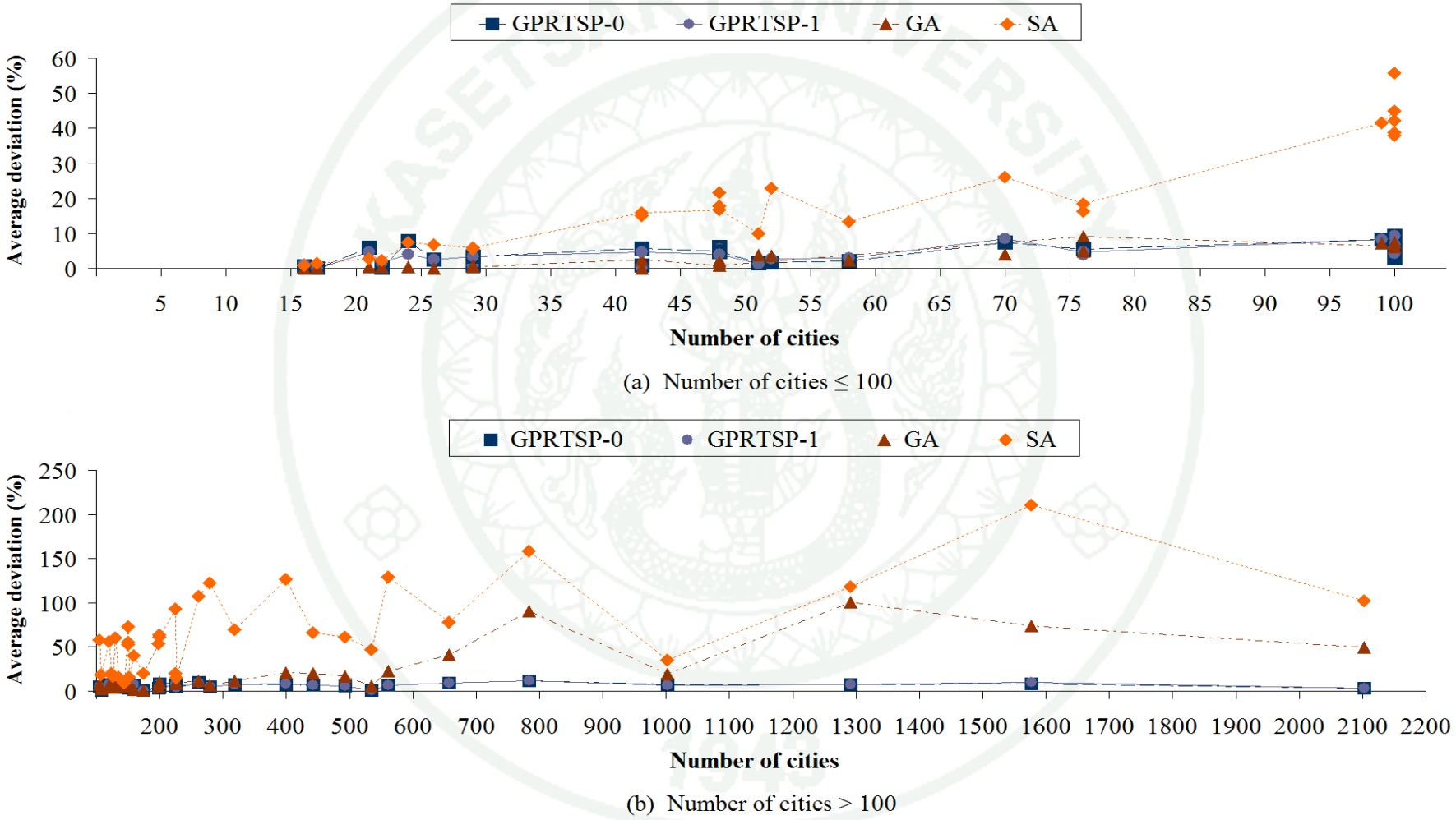


Figure 11 Comparisons of solution deviations among four algorithms

Copyright by Kasetsart University All rights reserved

3.2 Comparison of the running time

The first comparison is to compare the running time of the GPRTSP-0 algorithm to the GPRTSP-1 algorithm. The GPRTSP-1 algorithm consumes more average running time than the GPRTSP-0 algorithm on almost all 60 instances, except for the fl1577 instance (Table 5). This is because of the stopping criteria that are specified in Table 2. The GPRTSP-1 algorithm expands larger boundary of searching limit than the earlier one, so it takes a longer time for searching. Although the GPRTSP-1 algorithm spends more run time in the remaining 59 instances, overall solutions obtained are slightly better.

Comparing with the GA algorithm, the GPRTSP-0 algorithm consumes more average running time than the GA algorithm when the problem size is more than 318 cities (Table 5). The running time (upon 9 trials) of the GA algorithm can be seen in Table B7 in the Appendix B. The GPRTSP-0 algorithm spends less average running time on 49 TSP instances than the GA algorithm. Although the GPRTSP-0 algorithm spends more run time than the GA algorithm in the remaining 11 TSP instances, the solutions obtained are better (Table 5). For the GPRTSP-1 algorithm, this algorithm spends less average running time on 14 TSP instances than the GA algorithm. For the remaining 46 test problems, the GPRTSP-1 algorithm spends more average running time than the GA algorithm due to this algorithm expands larger boundary of searching limit that is specified by the stopping criteria. On the overall, the GPRTSP-1 algorithm achieves better solution quality (Table 5).

Finally, comparing with the SA algorithm, the GPRTSP-0 algorithm consumes less average running time than the SA algorithm on almost all 60 instances (Table 5), except for the si535, d657, rat783, pr1002, d1291, and fl1577 test problems. In addition, the running time (upon 9 trials) of the SA algorithm can be seen in Table B8 in the Appendix B. Although the GPRTSP-0 algorithm spends more average running time than the SA algorithm in those six instances, the solutions obtained are better (Table 5). For the GPRTSP-1 algorithm, this algorithm spends more average running time than the SA algorithm, except for the gr24, fri26,

Copyright by Kasetsart University All rights reserved

dantzig42, eil51, st70, eil76, and pa561 test problems (Table 5). This is because of the stopping criteria (as specified in Table 2), which it allows expanding larger boundary of searching limit in the GPRTSP-1 algorithm. In contrast to the stopping criterion of the GPRTSP-1 algorithm, the SA algorithm terminates when reaching the maximum number of iterations, and it is not allowed to repeat the whole procedure when the algorithm terminates. Although the GPRTSP-1 algorithm spends more average running time than the SA algorithm, the solution quality is better (Table 5).

The comparison plot of the average running time among four algorithms is illustrated in Figure 12. On the overall, the GPRTSP-1 algorithm significantly spends more average running time than the three benchmarking algorithms while the GPRTSP-0 algorithm spends less average running time than the GA and SA algorithms when the problem sizes are less than 400 and 600 cities, respectively. Lastly, for all 60 instances, the average running times consumed by the GPRTSP-0, GPRTSP-1, GA, and SA algorithms (Table 5) are: 256.09, 370.12, 162.89, and 270.17 seconds, respectively.

Copyright by Kasetsart University All rights reserved

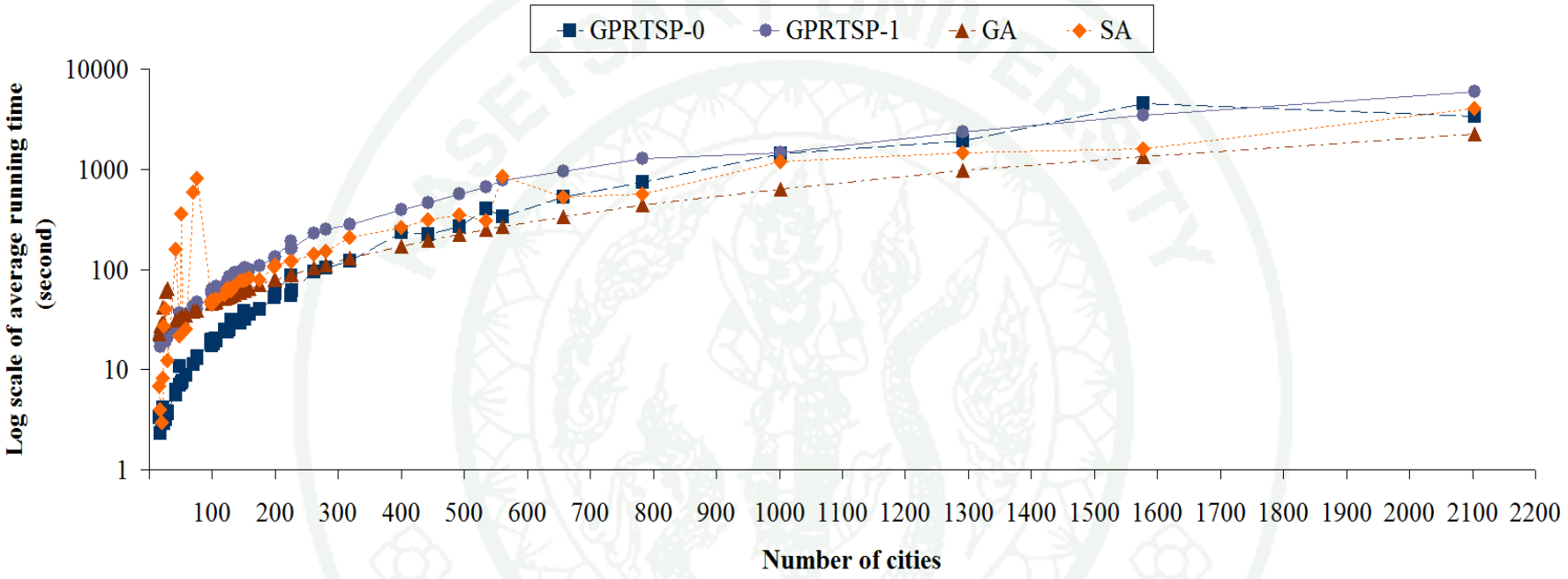


Figure 12 Comparison of average running time among four algorithms

Copyright by Kasetsart University All rights reserved

3.3 Comparison the solution quality with GA, SA, and other six recent algorithms

This comparison implements the GPRTSP-1, GA, and SA algorithms with fixed terminating time, and their average percentages of the deviation of their solutions from the optimal solutions are collected with two groups of runtime termination, depending on the instance size. In addition, their average percentages of the deviation of these methods are compared with the other six algorithms (Table 1). The first group consists of instances, whose size does not exceed 200 cities, and those algorithms terminate when the runtime reaches 500 seconds. The second group consists of instances, whose size is more than 200 cities. In this group, those algorithms terminate when the runtime reaches 1,000 seconds. Table 6 illustrates the comparison of the average percentages of the deviation of the solutions from the optimal solutions (or solution quality) of these nine methods. The bold number means that the given algorithm gives worse average percentage of the deviation of its solution from the optimal solution than the GPRTSP-1 algorithm, and the symbol "-" denotes that there is no result on the given method. Furthermore, the solutions obtained for the GPRTSP-1, GA, and SA algorithms with running time limitation (upon 9 trials) can be seen in Tables B9 through B11 in the Appendix B, respectively.

In Table 6, comparing with GA, there are 14 (out of 30) TSP instances that the average percentages of the average deviation from optimal solutions obtained by the GPRTSP-1 algorithm are better: eil51, berlin52, pr107, bier127, ch150, kroA200, lin318, pcb442, d657, rat783, pr1002, d1291, fl1577, and d2103. In addition, the average percentage of the average deviation of the GA solution from the optimal solution is definitely worse than that of the GPRTSP-1 solution when the instance size is high. For example, for the d1291, fl1577, and d2103 instances, the average percentages of the average deviation of the GA solution from the optimal solution are roughly 10, 6, 12 times worse than that of the GPRTSP-1 solution. Comparing with SA, our proposed algorithm performs better for all of 30 instances, and the average percentages of the average deviation of the SA solution from the optimal solution have an increasing trend when the instance size is large.

Copyright by Kasetsart University All rights reserved

Table 6 Comparison of solution quality of GPRTSP-1 with GA, SA, and other six recent algorithms

(Unit: percent)

Instances	Methods								
	KNIES-TSP	**Co-Adaptive Net**	**CONN**	**GCGA**	**Memetic SOM**	**Improved-EN**	**GA**	**SA**	**GPRTSP-1**
eil51	**2.86**	**2.89**	**2.58**	0.94	**2.14**	-	**2.16**	**10.54**	1.25
berlin52	-	**7.01**	**6.03**	-	**2.01**	-	**4.08**	**23.35**	1.82
st70	1.51	1.72	2.96	0.44	0.99	-	2.24	**27.79**	7.31
eil76	**4.98**	4.35	**5.02**	2.42	2.88	-	2.89	**16.48**	4.77
kroA100	-	1.31	2.57	1.23	1.14	2.09	2.86	**45.58**	3.64
kroB100	-	2.20	2.60	1.81	1.75	3.02	2.98	**41.31**	4.72
kroC100	-	1.70	1.53	1.33	0.71	2.33	2.88	**41.69**	5.94
kroD100	-	1.87	1.42	2.42	1.14	-	3.08	**52.43**	9.29
kroE100	-	2.56	1.97	1.41	1.99	**4.22**	2.63	**36.20**	3.02
rd100	2.09	3.64	3.59	1.53	2.65	-	4.82	**47.62**	6.61
lin105	1.98	1.08	0.38	1.15	0.34	-	4.56	**59.22**	5.13
pr107	**0.73**	**4.41**	**2.77**	**1.37**	**0.67**	-	**1.52**	**15.91**	0.64
pr124	0.07	2.93	1.74	0.19	1.52	-	2.41	**20.76**	5.62
bier127	2.76	3.00	2.45	1.80	2.78	-	**5.96**	**17.85**	4.01
ch130	-	2.82	4.66	-	2.83	-	5.53	**69.29**	7.51

Copyright by Kasetsart University All rights reserved

Table 6 (Continued)

(Unit: percent)

Instances	Methods								
	KNIES-TSP	**Co-Adaptive Net**	**CONN**	**GCGA**	**Memetic SOM**	**Improved-EN**	**GA**	**SA**	**GPRTSP-1**
pr136	4.53	4.65	2.27	2.82	3.10	-	3.10	**18.76**	9.00
ch150	-	3.23	3.29	-	2.95	-	**7.08**	**75.05**	3.69
kroA150	-	3.06	4.78	2.92	2.73	5.56	5.29	**53.81**	9.37
kroB150	-	2.60	3.09	2.11	1.61	3.80	5.60	**48.54**	6.04
pr152	0.97	2.06	0.79	1.22	2.60	-	2.00	**15.21**	2.89
kroA200	**5.72**	3.27	4.40	1.85	2.20	**5.53**	**7.69**	**59.88**	4.63
kroB200	-	2.31	4.24	4.04	3.92	5.31	7.91	**64.83**	8.19
lin318	-	4.31	-	5.14	5.51	**8.02**	**10.58**	**68.23**	7.41
pcb442	**11.07**	**7.58**	5.56	**8.99**	6.08	-	**12.51**	**61.77**	6.99
d657	-	5.52	7.58	-	5.02	-	**16.40**	**75.89**	9.02
rat783	-	6.57	7.59	-	5.95	9.83	**39.85**	**152.87**	12.26
pr1002	-	5.27	6.94	-	4.78	8.70	**17.13**	**32.24**	8.94
d1291	-	**11.98**	**10.80**	-	9.66	-	**104.30**	**118.44**	10.06
fl1577	-	14.36	8.76	-	**17.46**	-	**84.44**	**251.12**	15.24
d2103	-	**19.18**	2.67	-	**19.15**	-	**60.14**	**126.43**	4.99

Copyright by Kasetsart University All rights reserved

Our proposed GPRTSP-1 algorithm is compared with six recent methods proposed by other authors, in term of solution quality. For the eil51, eil76, pr107, kroA200, and pcb442 instances, our proposed GPRTSP-1 algorithm obtains better solution quality than the KNIES-TSP. Comparing with Co-Adaptive Net, the GPRTSP-1 algorithm can find better solutions on six instances (out of 30): the eil51, berlin52, pr107, pcb442, d1291, and d2103 instances. The GPRTSP-1 algorithm outperforms CONN on five instances (out of 29): the eil51, berlin52, eil76, pr107, and d1291 instances. However, comparing with GCGA, there are only two instances (pr107 and pcb442) that the GPRTSP-1 algorithm can do better. In addition, there are five (out of 30 TSP instances) that the GPRTSP-1 algorithm obtains better solution quality than the Memetic SOM. Lastly, comparing with the Improved-EN on 11 TSP instances, the GPRTSP-1 algorithm obtains better solution quality on three instances, i.e., kroE100, kroA200, and lin318 instances.

For the eil51 instance, our proposed GPRTSP-1 algorithm performs better than the comparing methods (except the GCGA algorithm). Comparing with all methods (excluding the algorithm with no result), for the berlin52 and pr107 instances, the GPRTSP-1 algorithm can find better solutions. Finally, for d2103 instance, the Co-Adaptive Net and the Memetic SOM methods obtain approximately four times higher percentage of the average deviation from the optimal solution than our proposed GPRTSP-1 algorithm.

The GPRTSP-1 algorithm obviously outperforms SA. Comparing with GA, the GPRTSP-1 algorithm achieves better solution quality on 14 instances, almost half of 30 TSP instances. Comparing with other six recent algorithms in literatures, on the overall, the GPRTSP-1 algorithm gives worse solution quality. However, there are some TSP instances where the GPRTSP-1 algorithm achieves better solution quality than those methods.

Copyright by Kasetsart University All rights reserved

CONCLUSION AND RECOMMENDATIONS

Conclusion

This thesis presents two approaches based on Gaussian process regression (GPR) for predicting the optimal tour of the symmetrically deterministic traveling salesman problem (TSP) with a single salesman. The response of the GPR models is the length of traveling tour while the predictor is the traveling tours with the cities' number. These two proposed methods adopt the nearest neighbor (NN) technique to construct the first sample tour, and use it to construct other sample tours by the 2-exchange neighborhood scheme; then they are treated as a training input of the GPR model for learning the prior distribution over this input. A single test input has the minimum length of traveling tour in the training set. Then it is used to make prediction by computing the posterior distribution; then the predictive function value is determined. The predictive function value is utilized to identify a tour whose length is close to this predictive value. Consequently, this tour becomes the estimated optimal tour.

The estimated optimal tour may not be optimal, and it can be improved further by a tour improvement procedure. This thesis investigates two types of the tour improvement procedure: the iterated 2-opt method and the iterated simulated annealing algorithm. The first one is embedded into the first proposed algorithm (called the GPRTSP-0 algorithm). The 2-opt algorithm executes on the estimated optimal tour with edges recombination by starting at the first node in a given tour, removing two edges for breaking the tour into two paths, and then reconnecting those paths with two other possible edges, respectively. If the new tour is shorter, it becomes the current tour; then this process is repeated with the next consecutive node until all nodes are selected. After that, the whole steps of the 2-opt algorithm is iterated until no improvement can be made in the two consecutive iterations.

Because of the optimality, the first algorithm can be stuck at a local optimum. Therefore, the second algorithm (called the GPRTSP-1 algorithm) is proposed in

Copyright by Kasetsart University All rights reserved

which it incorporates the second tour improvement procedure: the iterated simulated annealing (ISA) algorithm. The simulated annealing (SA) loop in this procedure executes from the estimated optimal tour, and then it starts by generating a candidate tour. Then the candidate tour is accepted if it is shorter; otherwise, it is accepted with some probability. This SA loop is repeated until the maximum number of iteration is reached, and it returns the best tour and its corresponding length. After that, the ISA loop is iterated by starting with the tour in previous loop until the maximum number of iterations is reached. However, during the ISA execution, the solution may be stuck at local optima. The restart method is implemented in the ISA procedure for which the search progression can continue on the next iteration. Thus, the ISA procedure terminates when reaching the maximum number of the restarting points (at 10 times) or reaching the maximum number of iterations.

The convergence of the GPRTSP-1 algorithm is examined through the discrete-time Markov chain theory in which its current solution depends only on the previous one. Furthermore, the numerical experiments are provided on not only the behavior analysis, but also the results executing on 60 TSP instances. The comparison is performed on the search performances of our two proposed algorithms, in terms of the solution quality and the running time. The performances of the two proposed fashions are compared with two well-known methods, i.e., the genetic algorithm (GA) and the simulated annealing (SA) algorithm. Moreover, the performance of the GPRTSP-1 algorithm is compared with the other six recent approaches, in term of solution quality, on 30 common TSP instances.

Recommendations

The performance of the GPRTSP-1 algorithm (as recommended in Section 2.1 in the results and discussion chapter) depends on the size of test problems and the choices of parameters of the ISA procedure. In order to reach one in the set of the globally optimal solutions, parameters should be fine tuned for each test problem; for example, the number of iterations allows searching for a longer time or the other stopping criteria (as defined in Table 2) should be adaptable. To enhance the

Copyright by Kasetsart University All rights reserved

performance of the GPRTSP-1 algorithm, in terms of solution quality and computation time, the recommendations contain three parts: the neighborhood generation, the elimination of accepting the recently visited solutions, and the adaptable cooling schedule, as follow:

The neighborhood generation relates to the construction step of the sample tour in the training set of the GPR and the candidate-generating step in the ISA procedure. The GPRTSP-1 algorithm uses the 2-exchange neighborhood function to generate solutions in both steps; however, some literatures (Tian *et al.*, 1999; Liu *et al.*, 2009) recommend that the sampling method (with reversing and/or moving a subsequence of cycle in a tour) has higher efficiency than the 2-exchange scheme. In addition, the multi-stage of distinct local search strategy can possibly be applied to exploit and intensify in the searching process for reaching the globally optimal solutions with less computation time. Therefore, these methods should be further investigated for improving the performance of the GPRTSP-1 algorithm.

The event of accepting the recently visited solutions is eliminated by implementing the memory list feature of the tabu search to check the solutions at the acceptance step of algorithm, in the case of turning back to the solutions visited in the previous iterations. Finally, the adaptable cooling schedule should be implemented with the adaptive function of decreasing the annealing temperature. For example, this function should be based on either the size of test problem or the multi-stage of function with having the distinct cooling rate.

Copyright by Kasetsart University All rights reserved

LITERATURE CITED

Aarts, E., J. Korst and W. Michiels. 2005. Simulated annealing, pp. 187-210. *In* E.K. Burke and G. Kendall, eds. **Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques.** Springer, New York.

Aarts, E.H.L., J.H.M. Korst and P.J.M. Van Laarhoven. 2003. Simulated annealing, pp. 91-120. *In* E. Aarts and J.K. Lenstra, eds. **Local Search in Combinatorial Optimization.** Princeton University Press, Princeton.

Alkhamis, T.M., M.A. Ahmed and V.K. Tuan. 1999. Simulated annealing for discrete optimization with estimation. **European Journal of Operational Research** 116 (3): 530-544.

Applegate, D.L., R.E. Bixby, V. Chvátal and W.J. Cook. 2006. **The traveling salesman problem: a computational study.** Princeton University Press, New Jersey.

Aras, N., B.J. Oommen and İ.K. Altınel. 1999. The Kohonen network incorporating explicit statistics and its application to the travelling salesman problem. **Neural Networks** 12 (9): 1273-1284.

Bagchi, T.P., J.N.D. Gupta and C. Sriskandarajah. 2006. A review of TSP based approaches for flowshop scheduling. **European Journal of Operational Research** 169 (3): 816-854.

Baldacci, R., E. Hadjiconstantinou and A. Mingozzi. 2003. An exact algorithm for the traveling salesman problem with deliveries and collections. **Networks** 42 (1): 26-41.

Copyright by Kasetsart University All rights reserved

Bellmore, M. and G.L. Nemhauser. 1968. The traveling salesman problem: a survey. **Operations Research** 16 (3): 538-558.

Blum, C. and A. Roli. 2003. Metaheuristics in combinatorial optimization: overview and conceptual comparison. **ACM Computing Surveys** 35 (3): 268-308.

Boyle, P. 2007. **Gaussian processes for regression and optimisation.** Ph.D. Thesis, Victoria University of Wellington.

Brest, J. and J. Žerovnik. 2005. A heuristic for the asymmetric traveling salesman problem, pp. 145-150. *In* **Proceedings of the 6th Metaheuristics International Conference (MIC2005).** INFORMS, Vienna, Austria.

Burke, E., P. Cowling and R. Keuthen. 2001. Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem, pp. 203-212. *In* E. Boers, ed. **Applications of Evolutionary Computing.** Springer, Berlin.

Chatterjee, S., C. Carrera and L.A. Lynch. 1996. Genetic algorithms and traveling salesman problems. **European Journal of Operational Research** 93 (3): 490-510.

Chen, P., H.K. Huang and X.Y. Dong. 2010. Iterated variable neighborhood descent algorithm for the capacitated vehicle routing problem. **Expert Systems with Applications** 37 (2): 1620-1627.

Chen, S.M. and C.Y. Chien. 2011. Solving the traveling salesman problem based on the genetic simulated annealing ant colony system with particle swarm optimization techniques. **Expert Systems with Applications** 38 (12): 14439-14450.

Copyright by Kasetsart University All rights reserved

Chen, T. and J. Ren. 2009. Bagging for Gaussian process regression. **Neurocomputing** 72 (7-9): 1605-1610.

Chernykh, I., A. Kononov and S. Sevastyanov. 2013. Efficient approximation algorithms for the routing open shop problem. **Computers and Operations Research** 40 (3): 841-847.

Choi, I.C., S.I. Kim and H.S. Kim. 2003. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. **Computers and Operations Research** 30 (5): 773-786.

Cochrane, E.M. and J.E. Beasley. 2003. The co-adaptive neural network approach to the Euclidean travelling salesman problem. **Neural Networks** 16 (10): 1499-1525.

Cox, D.R., M. Burmeister, E.R. Price, S. Kim and R.M. Myers. 1990. Radiation hybrid mapping: a somatic cell genetic method for constructing high-resolution maps of mammalian chromosomes. **Science** 250 (4978): 245-250.

Créput, J.C. and A. Koukam. 2009. A memetic neural network for the Euclidean traveling salesman problem. **Neurocomputing** 72 (4-6): 1250-1264.

Dantzig, G., R. Fulkerson and S. Johnson. 1954. Solution of a large-scale traveling-salesman problem. **Operations Research** 2 (4): 393-410.

Dawid, H. 1994. A Markov chain analysis of genetic algorithms with a state dependent fitness function. **Complex Systems** 8 (6): 407-417.

De Boer, P.T., D.P. Kroese, S. Mannor and R.Y. Rubinstein. 2005. A tutorial on the cross-entropy method. **Annals of Operations Research** 134 (1): 19-67.

Copyright by Kasetsart University All rights reserved

Derbel, H., B. Jarboui, S. Hanafi and H. Chabchoub. 2012. Genetic algorithm with iterated local search for solving a location-routing problem. **Expert Systems with Applications** 39 (3): 2865-2871.

Dong, G., W.W. Guo and K. Tickle. 2012. Solving the traveling salesman problem using cooperative genetic ant systems. **Expert Systems with Applications** 39 (5): 5006-5011.

Dong, M.G. and N. Wang. 2012. A novel hybrid differential evolution approach to scheduling of large-scale zero-wait batch processes with setup times. **Computers and Chemical Engineering** 45: 72-83.

Dong, X., H. Huang and P. Chen. 2011. Study on iterated local search algorithm for permutation flowshop problem with total flowtime objective, pp. 236-245. *In* D. Zeng, ed. **Applied Informatics and Communication.** Springer, Heidelberg.

_____, P. Chen, H. Huang and M. Nowak. 2013. A multi-restart iterated local search algorithm for the permutation flow shop problem minimizing total flow time. **Computers and Operations Research** 40 (2): 627-632.

Dorea, C.C.Y., J.A. Guerra, R. Morgado and A.G.C. Pereira. 2010. Multistage Markov chain modeling of the genetic algorithm and convergence results. **Numerical Functional Analysis and Optimization** 31 (2): 164-171.

Dorigo, M. and L.M. Gambardella. 1997. Ant colony system: a cooperative learning approach to the traveling salesman problem. **IEEE Transactions on Evolutionary Computation** 1 (1): 53-66.

_____, V. Maniezzo and A. Colorni. 1996. Ant system: optimization by a colony of cooperating agents. **IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics** 26 (1): 29-41.

Copyright by Kasetsart University All rights reserved

Duman, E. and I. Or. 2004. Precedence constrained TSP arising in printed circuit board assembly. **International Journal of Production Research** 42 (1): 67-78.

Durrett, R. 2012. **Essentials of stochastic processes.** 2nd ed. Springer, New York.

Eastman, W.L. 1958. **Linear programming with pattern constraints.** Ph.D. Thesis, Harvard University, Massachusetts, USA. *Cited* D.L. Applegate, R.E. Bixby, V. Chvátal, and W.J. Cook. 2006. **The traveling salesman problem: a computational study.** Princeton University Press, New Jersey.

Erdoğan, G., J.F. Cordeau and G. Laporte. 2009. The pickup and delivery traveling salesman problem with first-in-first-out loading. **Computers and Operations Research** 36 (6): 1800-1808.

Eshragh, A., J.A. Filar and M. Haythorpe. 2009. A hybrid simulation-optimization algorithm for the Hamiltonian cycle problem. **Annals of Operations Research** 189 (1): 103-125.

Fanjul-Peyro, L. and R. Ruiz. 2010. Iterated greedy local search methods for unrelated parallel machine scheduling. **European Journal of Operational Research** 207 (1): 55-69.

Felipe, A., M. Ortuño and G. Tirado. 2012. An adapted heuristic approach for a clustered traveling salesman problem with loading constraints. **4OR: A Quarterly Journal of Operations Research** 10 (3): 245-265.

Freisleben, B. and P. Merz. 1996. A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems, pp. 616-621. *In* **Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC '96).** IEEE Press, Nagoya, Japan.

Copyright by Kasetsart University All rights reserved

Gendreau, M., G. Laporte and F. Semet. 1998. A tabu search heuristic for the undirected selective travelling salesman problem. **European Journal of Operational Research** 106 (2-3): 539-545.

Geng, X., Z. Chen, W. Yang, D. Shi and K. Zhao. 2011. Solving the traveling salesman problem based on an adaptive simulated annealing algorithm with greedy search. **Applied Soft Computing** 11 (4): 3680-3689.

Ghate, A. and R.L. Smith. 2008. Adaptive search with stochastic acceptance probabilities for global optimization. **Operations Research Letters** 36 (3): 285-290.

Grötschel, M., M. Jünger and G. Reinelt. 1991. Optimal control of plotting and drilling machines: A case study. **Zeitschrift für Operations Research** 35 (1): 61-84.

Gutjahr, W.J. 2002. ACO algorithms with guaranteed convergence to the optimal solution. **Information Processing Letters** 82 (3): 145-153.

_____. 2008. First steps to the runtime complexity analysis of ant colony optimization. **Computers and Operations Research** 35 (9): 2711-2727.

Hanafi, S. 2001. On the convergence of tabu search. **Journal of Heuristics** 7 (1): 47-58.

Hansen, P. and N. Mladenović. 2001. Variable neighborhood search: principles and applications. **European Journal of Operational Research** 130 (3): 449-467.

_____, _____ and J.A. Moreno Pérez. 2008. Variable neighbourhood search: methods and applications. **4OR: A Quarterly Journal of Operations Research** 6 (4): 319-360.

Copyright by Kasetsart University All rights reserved

Helsgaun, K. 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. **European Journal of Operational Research** 126 (1): 106-130.

_____. 2009. General k-opt submoves for the Lin-Kernighan TSP heuristic. **Mathematical Programming Computation** 1 (2-3): 119-163.

Henderson, D. 2001. **Assessing the finite-time performance of local search algorithms.** Ph.D. Thesis, Virginia Polytechnic Institute and State University.

Herault, L. 2000. Rescaled simulated annealing - accelerating convergence of simulated annealing by rescaling the states energies. **Journal of Heuristics** 6 (2): 215-252.

Hernández-Pérez, H. and J.J. Salazar-González. 2004. Heuristics for the one-commodity pickup-and-delivery traveling salesman problem. **Transportation Science** 38 (2): 245-255.

Idé, T. and S. Kato. 2009. Travel-time prediction using Gaussian process regression: a trajectory-based approach, pp. 1185-1196. *In* **Proceedings of the 9th SIAM International Conference on Data Mining (SDM).** SIAM, Nevada, USA.

Jacobson, S.H. and E. Yücesan. 2004a. Analyzing the performance of generalized hill climbing algorithms. **Journal of Heuristics** 10 (4): 387-405.

_____, _____. 2004b. Global optimization performance measures for generalized hill climbing algorithms. **Journal of Global Optimization** 29 (2): 173-190.

_____, S.N. Hall, L.A. McLay and J.E. Orosz. 2005. Performance analysis of cyclical simulated annealing algorithms. **Methodology and Computing in Applied Probability** 7 (2): 183-201.

Copyright by Kasetsart University All rights reserved

Jeong, S.J., K.S. Kim and Y.H. Lee. 2009. The efficient search method of simulated annealing using fuzzy logic controller. **Expert Systems with Applications** 36 (3): 7099-7103.

Jiang, M., Y.P. Luo and S.Y. Yang. 2007. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. **Information Processing Letters** 102 (1): 8-16.

Johnson, A.W. and S.H. Jacobson. 2002a. A class of convergent generalized hill climbing algorithms. **Applied Mathematics and Computation** 125 (2-3): 359-373.

_____, _____. 2002b. On the convergence of generalized hill climbing algorithms. **Discrete Applied Mathematics** 119 (1-2): 37-57.

Johnson, D.S. and L.A. McGeoch. 2003. The travelling salesman problem: a case study, pp. 215-310. *In* E. Aarts and J.K. Lenstra, eds. **Local Search in Combinatorial Optimization.** Princeton University Press, Princeton.

Karapetyan, D. and G. Gutin. 2011. Lin-Kernighan heuristic adaptations for the generalized traveling salesman problem. **European Journal of Operational Research** 208 (3): 221-232.

Kirk, J. 2007. **Traveling salesman problem - genetic algorithm**. Available Source: http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm, January 10, 2012.

Ko, J., D.J. Klein, D. Fox and D. Haehnel. 2007. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp, pp. 742-747. *In* **Proceedings of the 2007 IEEE International Conference on Robotics and Automation.** IEEE Press, Roma, Italy.

Copyright by Kasetsart University All rights reserved

Krause, A., A. Singh and C. Guestrin. 2008. Near-optimal sensor placements in Gaussian processes: theory, efficient algorithms and empirical studies. **Journal of Machine Learning Research** 9 (2): 235-284.

Laporte, G. 1992. The traveling salesman problem: an overview of exact and approximate algorithms. **European Journal of Operational Research** 59 (2): 231-247.

Larrañaga, P., C.M.H. Kuijpers, R.H. Murga, I. Inza and S. Dizdarevic. 1999. Genetic algorithms for the travelling salesman problem: a review of representations and operators. **Artificial Intelligence Review** 13 (2): 129-170.

Lasdon, L.S. 2002. **Optimization theory for large systems.** Dover Publications, Inc., New York.

Liang, Y.C., Y.M. Hsiao and C.Y. Tien. 2013. Metaheuristics for drilling operation scheduling in Taiwan PCB industries. **International Journal of Production Economics** 141 (1): 189-198.

Liao, Y.F., D.H. Yau and C.L. Chen. 2012. Evolutionary algorithm to traveling salesman problems. **Computers and Mathematics with Applications** 64 (5): 788-797.

Lin, S. and B.W. Kernighan. 1973. An effective heuristic algorithm for the traveling-salesman problem. **Operations Research** 21 (2): 498-516.

Little, J.D.C., K.G. Murty, D.W. Sweeney and C. Karel. 1963. An algorithm for the traveling salesman problem. **Operations Research** 11 (6): 972-989.

Copyright by Kasetsart University All rights reserved

Liu, Y., S. Xiong and H. Liu. 2009. Hybrid simulated annealing algorithm based on adaptive cooling schedule for TSP, pp. 895-898. *In* L. Xu, E.D. Goodman, G. Chen, D. Whitley and Y. Ding, eds. **Proceedings of the First ACM/SIGEVO Summit on Genetic and Evolutionary Computation (GEC '09).** Association for Computing Machinery, New York.

Lizotte, D., T. Wang, M. Bowling and D. Schuurmans. 2007. Automatic gait optimization with Gaussian process regression, pp. 944-949. *In* **Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI'07).** Morgan Kaufmann Publishers, Inc., Hyderabad, India.

Lo, C.C. and C.C. Hsu. 1998. An annealing framework with learning memory. **IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans** 28 (5): 648-661.

Lourenço, H.R., O.C. Martin and T. Stützle. 2010. Iterated local search: framework and applications, pp. 363-397. *In* M. Gendreau and J.Y. Potvin, eds. **Handbook of Metaheuristics.** Springer, New York.

Marinakis, Y., M. Marinaki and G. Dounias. 2011. Honey bees mating optimization algorithm for the euclidean traveling salesman problem. **Information Sciences** 181 (20): 4684-4698.

Martí, R., M.G.C. Resende and C.C. Ribeiro. 2013. Multi-start methods for combinatorial optimization. **European Journal of Operational Research** 226 (1): 1-8.

Martin, O., S.W. Otto and E.W. Felten. 1991. Large-step Markov chains for the traveling salesman problem. **Complex Systems** 5: 299-326.

Copyright by Kasetsart University All rights reserved

Martin, O., S.W. Otto and E.W. Felten. 1992. Large-step Markov chains for the TSP incorporating local search heuristics. **Operations Research Letters** 11 (4): 219-224.

Martin, O.C. and S.W. Otto. 1996. Combining simulated annealing with local search heuristics. **Annals of Operations Research** 63 (1): 57-75.

Mladenović, N., D. Urošević, S. Hanafi and A. Ilić. 2012. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. **European Journal of Operational Research** 220 (1): 270-285.

Moon, C., J. Kim, G. Choi and Y. Seo. 2002. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. **European Journal of Operational Research** 140 (3): 606-617.

Nagata, Y. and D. Soler. 2012. A new genetic algorithm for the asymmetric traveling salesman problem. **Expert Systems with Applications** 39 (10): 8947-8953.

Nikolaev, A.G. and S.H. Jacobson. 2011. Using Markov chains to analyze the effectiveness of local search algorithms. **Discrete Optimization** 8 (2): 160-173.

Nolte, A. and R. Schrader. 2000. A note on the finite time behavior of simulated annealing. **Mathematics of Operations Research** 25 (3): 476-484.

Nuhoglu, M. 2007. **Shortest path heuristics (nearest neighborhood, 2 opt, farthest and arbitrary insertion) for travelling salesman problem**. Available Source: http://icodesnip.com/snippet/matlab/shortest-path-heuristics-nearest-neighborhood-2-opt-farthest-and-arbitrary-insertion-for-travelling-salesman-problem, December 23, 2011.

Copyright by Kasetsart University All rights reserved

Orosz, J.E. and S.H. Jacobson. 2002a. Analysis of static simulated annealing algorithms. **Journal of Optimization Theory and Applications** 115 (1): 165-182.

_____, _____. 2002b. Finite-time performance analysis of static simulated annealing algorithms. **Computational Optimization and Applications** 21 (1): 21-53.

Pal, M. and S. Deswal. 2010. Modelling pile capacity using Gaussian process regression. **Computers and Geotechnics** 37 (7-8): 942-947.

Platt, J.C., C.J.C. Burges, S. Swenson, C. Weare and A. Zheng. 2002. Learning a Gaussian process prior for automatically generating music playlists, pp. 1425-1432. *In* T.G. Dietterich, S. Becker and Z. Ghahramani, eds. **Advances in Neural Information Processing Systems 14.** MIT Press, Cambridge.

Potvin, J.Y. 1996. Genetic algorithms for the traveling salesman problem. **Annals of Operations Research** 63 (3): 337-370.

Press, W.H., S.A. Teukolsky, W.T. Vetterling and B.P. Flannery. 2007. **Numerical recipes: the art of scientific computing.** 3rd ed. Cambridge University Press, New York.

Qi, C. 2007. An ant colony system hybridized with randomized algorithm for TSP, pp. 461-465. *In* W. Feng and F. Gao, eds. **Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD2007).** IEEE Press, Qingdao, China.

Quiñonero-Candela, J. and C.E. Rasmussen. 2005. A unifying view of sparse approximate Gaussian process regression. **Journal of Machine Learning Research** 6: 1939-1959.

Copyright by Kasetsart University All rights reserved

Rasmussen, C.E. and C.K.I. Williams. 2006. **Gaussian processes for machine learning.** MIT Press, Cambridge.

_____ and H. Nickisch. 2010. Gaussian processes for machine learning (GPML) toolbox. **Journal of Machine Learning Research** 11 (11): 3011-3015.

Ratliff, H.D. and A.S. Rosenthal. 1983. Order-picking in a rectangular warehouse: a solvable case of the traveling salesman problem. **Operations Research** 31 (3): 507-521.

Rego, C., D. Gamboa, F. Glover and C. Osterman. 2011. Traveling salesman problem heuristics: leading methods, implementations and latest advances. **European Journal of Operational Research** 211 (3): 427-441.

Reinelt, G. 1995. **The TSPLIB symmetric traveling salesman problem instances**. Available Source: http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/, December 21, 2011.

Ross, S.M. 2007. **Introduction to Probability Models.** 9th ed. Academic Press, San Diego.

Rubinstein, R.Y. 2006. How many needles are in a haystack, or how to solve #P-complete counting problems fast. **Methodology and Computing in Applied Probability** 8 (1): 5-51.

Rudolph, G. 1994. Convergence analysis of canonical genetic algorithms. **IEEE Transactions on Neural Networks** 5 (1): 96-101.

Ruland, K.S. and E.Y. Rodin. 1997. The pickup and delivery problem: faces and branch-and-cut algorithm. **Computers and Mathematics with Applications** 33 (12): 1-13.

Copyright by Kasetsart University All rights reserved

Saadatmand-Tarzjan, M., M. Khademi, M.R. Akbarzadeh-T and H.A. Moghaddam. 2007. A novel constructive-optimizer neural network for the traveling salesman problem. **IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics** 37 (4): 754-770.

Sankoff, D. and M. Blanchette. 1997. The median problem for breakpoints in comparative genomics, pp. 251-263. *In* T. Jiang and D.T. Lee, eds. **Computing and Combinatorics.** Springer, Heidelberg.

Schmid, V., K.F. Doerner and G. Laporte. 2013. Rich routing problems arising in supply chain management. **European Journal of Operational Research** 224 (3): 435-448.

Schmitt, L.M. 2004. Theory of genetic algorithms II: models for genetic operators over the string-tensor representation of populations and convergence to global optima for arbitrary fitness function under scaling. **Theoretical Computer Science** 310 (1-3): 181-231.

Seshadri, A. 2006. **Traveling salesman problem (TSP) using simulated annealing**. Available Source: http://www.mathworks.com/matlabcentral/fileexchange/9612-traveling-salesman-problem-tsp-using-simulated-annealing, January 10, 2012.

Shi, X.H., Y.C. Liang, H.P. Lee, C. Lu and Q.X. Wang. 2007. Particle swarm optimization-based algorithms for TSP and generalized TSP. **Information Processing Letters** 103 (5): 169-176.

Sinz, F.H., J. Quiñonero-Candela, G.H. Bakır, C.E. Rasmussen and M.O. Franz. 2004. Learning depth from stereo, pp. 245-252. *In* C.E. Rasmussen, H.H. Bülthoff, B. Schölkopf and M.A. Giese, eds. **Pattern Recognition.** Springer, Heidelberg.

Copyright by Kasetsart University All rights reserved

Snelson, E. and Z. Ghahramani. 2006. Sparse Gaussian processes using pseudo-inputs, pp. 1257-1264. *In* Y. Weiss, B. Schölkopf and J. Platt, eds. **Advances in Neural Information Processing Systems 18.** MIT Press, Cambridge.

Stützle, T. 2006. Iterated local search for the quadratic assignment problem. **European Journal of Operational Research** 174 (3): 1519-1539.

_____ and H.H. Hoos. 2002. Analysing the run-time behaviour of iterated local search for the travelling salesman problem, pp. 589-611. *In* P. Hansen and C. Ribeiro, eds. **Essays and Surveys on Metaheuristics.** Kluwer Academic Publishers, Boston.

Suzuki, J. 1995. A Markov chain analysis on simple genetic algorithms. **IEEE Transactions on Systems, Man and Cybernetics** 25 (4): 655-659.

_____. 2010. A Markov chain analysis of genetic algorithms: large deviation principle approach. **Journal of Applied Probability** 47 (4): 967-975.

Theys, C., O. Bräysy, W. Dullaert and B. Raa. 2010. Using a TSP heuristic for routing order pickers in warehouses. **European Journal of Operational Research** 200 (3): 755-763.

Tian, P., J. Ma and D.M. Zhang. 1999. Application of the simulated annealing algorithm to the combinatorial optimisation problem with permutation property: an investigation of generation mechanism. **European Journal of Operational Research** 118 (1): 81-94.

Trelea, I.C. 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. **Information Processing Letters** 85 (6): 317-325.

Copyright by Kasetsart University All rights reserved

Vaughan, D.E. and S.H. Jacobson. 2004. Tabu guided generalized hill climbing algorithms. **Methodology and Computing in Applied Probability** 6 (3): 343-354.

_____, _____ and H. Kaul. 2007. Analyzing the performance of simultaneous generalized hill climbing algorithms. **Computational Optimization and Applications** 37 (1): 103-119.

Verhoeven, M.G.A., E.H.L. Aarts and P.C.J. Swinkels. 1995. A parallel 2-opt algorithm for the traveling salesman problem. **Future Generation Computer Systems** 11 (2): 175-182.

Voudouris, C. and E. Tsang. 1999. Guided local search and its application to the traveling salesman problem. **European Journal of Operational Research** 113 (2): 469-499.

Williams, C.K.I. and M. Seeger. 2001. Using the Nyström method to speed up kernel machines, pp. 682-688. *In* T.K. Leen, T.G. Dietterich and V. Tresp, eds. **Advances in Neural Information Processing Systems 13.** MIT Press, Cambridge.

_____, C.E. Rasmussen, A. Schwaighofer and V. Tresp. (2002) **Observations on the Nyström method for Gaussian process prediction**, University of Edinburgh and University College London, London, UK. pp. 9.

Wu, C., Y. Liang, H.P. Lee and C. Lu. 2004. Generalized chromosome genetic algorithm for generalized traveling salesman problems and its applications for machining. **Physical Review E** 70 (1): 016701-1–016701-13.

Yang, J., C. Wu, H.P. Lee and Y. Liang. 2008. Solving traveling salesman problems using generalized chromosome genetic algorithm. **Progress in Natural Science** 18 (7): 887-892.

Copyright by Kasetsart University All rights reserved

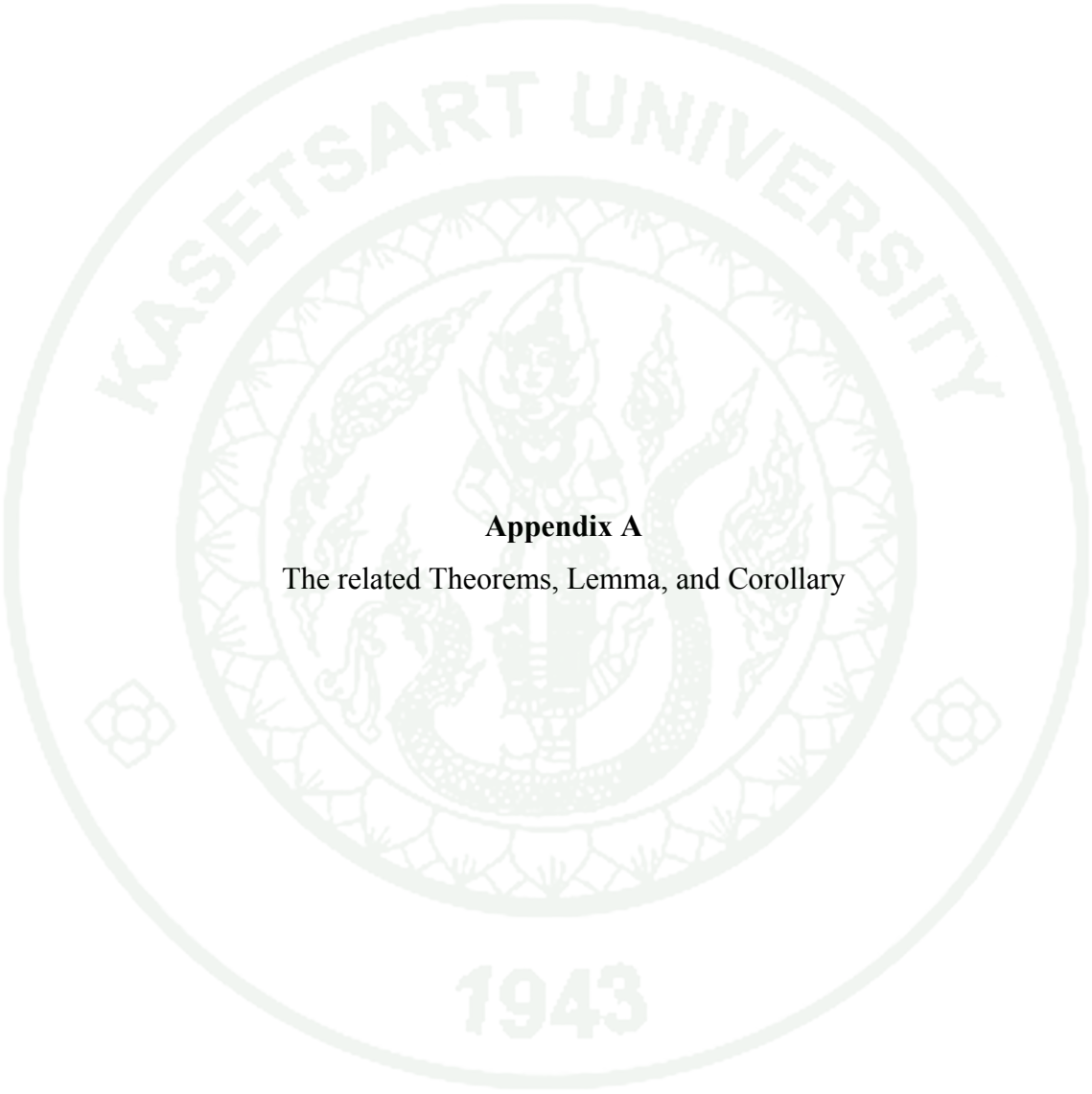
Yi, J., G. Yang, Z. Zhang and Z. Tang. 2009. An improved elastic net method for traveling salesman problem. **Neurocomputing** 72 (4-6): 1329-1335.

Yuan, J., K. Wang, T. Yu and M. Fang. 2008. Reliable multi-objective optimization of high-speed WEDM process based on Gaussian process regression. **International Journal of Machine Tools and Manufacture** 48 (1): 47-60.

Copyright by Kasetsart University All rights reserved

APPENDICES

Copyright by Kasetsart University All rights reserved



Appendix A

The related Theorems, Lemma, and Corollary

Copyright by Kasetsart University All rights reserved

Appendix Theorem A1 (Theorem 1 in Johnson and Jacobson (2002a), p. 362) Consider a homogeneous Markov chain defined by a transition matrix $\boldsymbol{P}(k)$, $k \in \mathbb{Z}^+$ fixed, where $P_{i,j}(k)$ is the probability of going from state i to state j in one step, and $P_{i,j}^n(k)$ is the probability of going from state i state j in n steps. If the Markov chain is irreducible and aperiodic with finitely many states, then $\lim_{n\to\infty} P_{i,j}^n(k) = \pi_j(k)$ exists for all $i, j \in \Omega$. Furthermore, $\pi_j(k)$ is the unique strictly positive solution of

$$\pi_j(k) = \sum_{i\in\Omega} \pi_i(k) P_{i,j}(k), \quad j \in \Omega,$$

and

$$\sum_{j\in\Omega} \pi_j(k) = 1.$$

Appendix Theorem A2 (Theorem 2 in Johnson and Jacobson (2002a), p. 363) Let (Ω, c) denote an instance of a discrete optimization problem. Let η be a neighborhood function defined on Ω. Let the generalized hill climbing (GHC) transition probability $P_{i,j}(k)$ be defined by (1), where the acceptance probabilities are defined by (6). Let the generation probabilities $g_{i,j}(k)$ satisfy (2) and the conditions

(a) for all $i, j \in \Omega$ and all iterations k, there exists an integer $d \geq 1$, and a corresponding sequence of states $l_0, l_1, l_2, ..., l_d \in \Omega$, with $l_0 = i$, $l_d = j$, and $g_{l_n, l_{n+1}}(k) > 0$, $n = 0,1,2,...,d-1$,

(b) for all $i, j \in \Omega$, $g_{i,j}(k) = g_{j,i}(k)$, and $g_{i,j}(k)$ is independent of k (i.e., $g_{i,j}(k) = g_{j,i}(k) = g_{i,j} = g_{j,i}$. Moreover, let the acceptance probabilities satisfy

(c) $\Pr\{R_k(i,j) \geq \Delta_{i,j}\} > 0$ for all $i, j \in \Omega$ and all iterations $k \in \mathbb{Z}^+$.

Copyright by Kasetsart University All rights reserved

Then the stationary distribution for all $i \in \Omega$, and for each outer loop iteration k, is

$$\pi_i(k) = \frac{\Pr\left\{\tilde{R}_k(\text{opt}, i) \geq \Delta_{\text{opt},i}\right\}}{\sum_{n \in \Omega} \Pr\left\{\tilde{R}_k(\text{opt}, n) \geq \Delta_{\text{opt},n}\right\}},$$

where $\tilde{R}_k(\text{opt}, i)$ and $\tilde{R}_k(\text{opt}, n)$ are random variables.

Appendix Theorem A3 (Theorem 3 in Johnson and Jacobson (2002a), p. 364) Under the conditions and assumptions of Theorem A2 if, for all $i, j \in \Omega$, and for all iterations $k \in \mathbb{Z}^+$,

$$c_i < c_j \quad \Rightarrow \quad \lim_{k \to \infty} \Pr\left\{R_k(i, j) \geq \Delta_{i,j}\right\} = 0,$$

then as k approaches infinity, the unique stationary distributions $\pi(k)$ of Theorem A2 approach the limiting form

$$\lim_{k \to \infty} \pi_i(k) = \begin{cases} \left|\Omega^{\text{opt}}\right|^{-1} & \text{if } i \in \Omega^{\text{opt}}, \\ 0 & \text{otherwise}. \end{cases}$$

Appendix Theorem A4 (Theorem 3 in Aarts *et al.* (2003), p. 102) Let $(\mathcal{S}, f)$ be an instance of a combinatorial optimization problem, $\mathcal{N}$ a neighborhood function, and $\boldsymbol{P}(k)$ the transition matrix of the homogeneous Markov chain associated with the simulated annealing algorithm defined by (4), with the simulated annealing temperature $c_k = c$ for all k. Furthermore, let the following conditions hold:

(G1) $\forall c > 0 \; \forall i, j \in \mathcal{S} \; \exists p \geq 1 \; \exists l_0, l_1, \ldots, l_p \in \mathcal{S}$ *with* $l_0 = i, l_p = j,$

and $G_{l_k l_{k+1}}(c) > 0, \; k = 0, 1, \ldots, p-1,$

(G2) $\forall c > 0 \; \forall i, j \in \mathcal{S}: \quad G_{ij}(c) = G_{ji}(c),$

Copyright by Kasetsart University All rights reserved

(A1) $\forall c > 0 \; \forall i, j \in \mathscr{S}: \quad A_{ij}(c) = 1 \qquad if \;\; f(i) \geq f(j),$

$$A_{ij}(c) \in (0,1) \quad if \;\; f(i) < f(j),$$

(A2) $\forall c > 0 \; \forall i, j, k \in \mathscr{S}: A_{ij}(c) A_{jk}(c) A_{ki}(c) = A_{ik}(c) A_{kj}(c) A_{ji}(c),$

(A3) $\forall i, j \in \mathscr{S} \; with \; f(i) < f(j): \lim_{c \downarrow 0} A_{ij}(c) = 0,$

where $G_{ij}(c)$ denotes the probability of generating a solution j from a solution i, $A_{ij}(c)$ denotes the probability of accepting a solution j that is generated from solution i. Then the Markov chain has a unique stationary distribution $q(c)$, whose components are given by

$$q_i(c) = \frac{1}{\sum_{j \in \mathscr{S}} \left(A_{ij}(c) / A_{ji}(c) \right)} \quad \text{for all } i \in \mathscr{S},$$

and

$$\lim_{c \downarrow 0} \; q_i(c) = q_i^*,$$

where the q_i^* is the stationary distribution that the simulated annealing algorithm finds solution i belonging to the set of globally optimal solutions.

Appendix Corollary A1 (Corollary 7.2 in Aarts *et al.* (2005), p. 194) Given an instance (S, f) of a combinatorial optimization problem and a suitable neighborhood function, and, furthermore, let the probability $q_i(c)$ that the simulated annealing finds solution i after an infinite number of trials at value c of the control parameter be given by (7.5), then

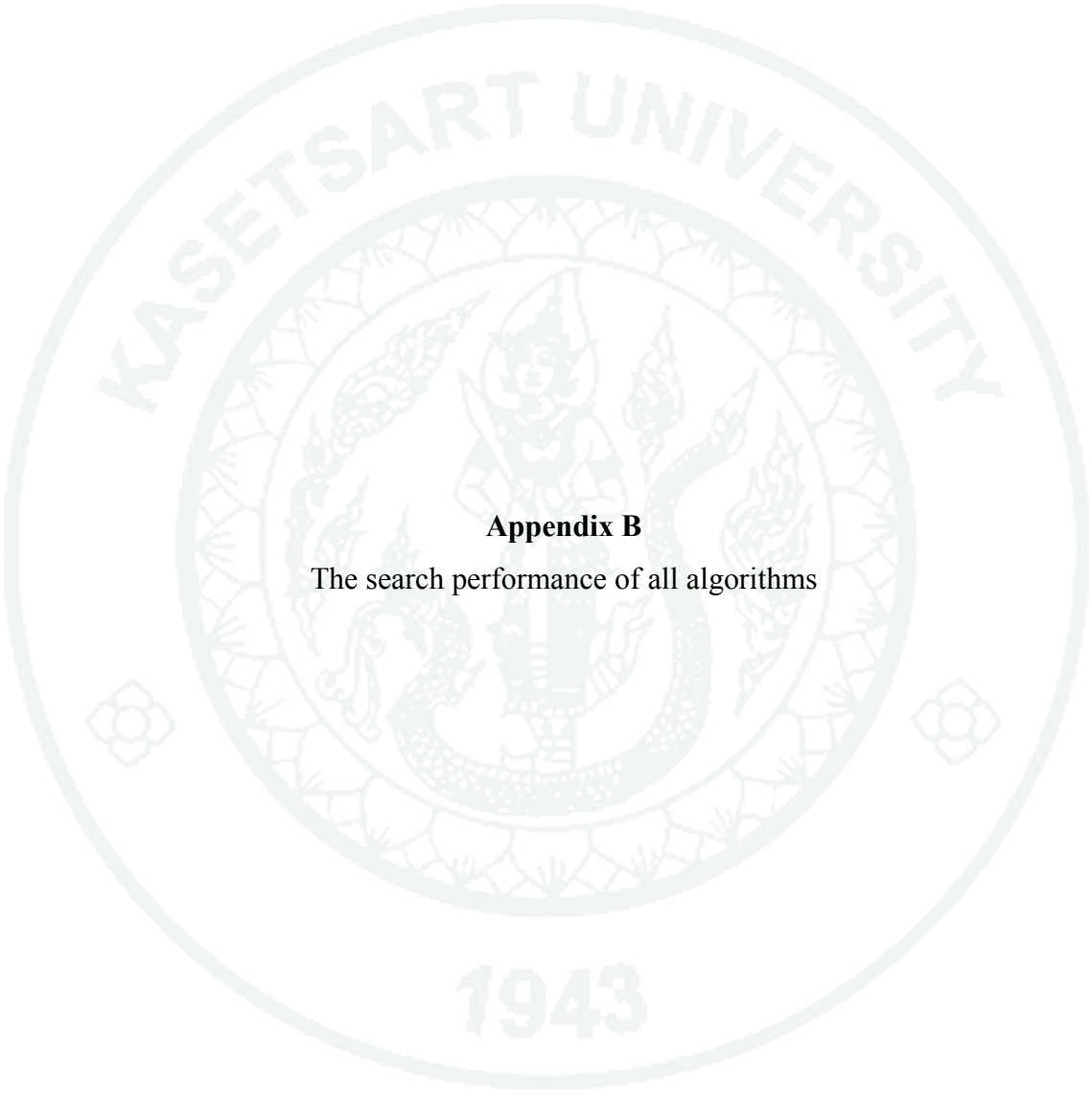
$$\lim_{c \downarrow 0} \; q_i(c) \triangleq q_i^* = \frac{1}{|S^*|} \chi_{(S^*)}(i),$$

Copyright by Kasetsart University All rights reserved

where S^* denotes the set of globally optimal solutions, and the characteristic function $\chi_{(S^*)}: S \to \{0,1\}$ of the set S^* is defined as $\chi_{(S^*)}(i) = 1$ if $i \in S^*$, and $\chi_{(S^*)}(i) = 0$ otherwise.

Appendix Lemma A1 (Lemma 1.4 in Durrett (2012), p. 16) If state i is accessible from state s and state s is accessible from state j, then state i is accessible from state j.

Copyright by Kasetsart University All rights reserved



Appendix B

The search performance of all algorithms

Copyright by Kasetsart University All rights reserved

Appendix Table B1 The solutions of GPRTSP-0 algorithm

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
ulysses16	16	6,859	6,909	6,909	6,909	6,909	6,909	6,909	6,909	6,909	6,909	6,909	6,909.0	-
gr17	17	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085.0	-
gr21	21	2,707	2,877	2,877	2,816	2,877	2,877	2,877	2,877	2,877	2,877	2,816	2,870.2	2,877
ulysses22	22	7,013	7,100	7,013	7,100	7,013	7,013	7,013	7,013	7,013	7,013	7,013	7,032.3	7,100
gr24	24	1,272	1,272	1,384	1,384	1,384	1,384	1,384	1,384	1,384	1,384	1,272	1,371.6	1,384
fri26	26	937	961	961	961	961	961	961	961	961	961	961	961.0	-
bayg29	29	1,610	1,668	1,668	1,668	1,660	1,668	1,668	1,668	1,668	1,660	1,660	1,666.2	1,668
bays29	29	2,020	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035.0	-
dantzig42	42	699	736	741	741	736	736	743	741	741	736	736	739.0	743
swiss42	42	1,273	1,285	1,274	1,341	1,274	1,274	1,274	1,274	1,274	1,285	1,274	1,283.9	1,341
att48	48	10,628	11,196	11,196	11,196	11,196	11,196	10,954	10,954	11,196	11,196	10,954	11,142.2	11,196
gr48	48	5,046	5,286	5,372	5,372	5,372	5,286	5,372	5,372	5,429	5,372	5,286	5,359.2	5,429
hk48	48	11,461	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003.0	-
eil51	51	426	436	428	428	436	436	436	428	436	428	428	432.4	436
berlin52	52	7,542	8,229	7,596	7,596	7,596	7,596	7,596	7,596	7,596	7,596	7,596	7,666.3	8,229
brazil58	58	25,395	25,978	25,978	25,978	25,978	25,978	25,978	25,978	25,978	25,699	25,699	25,947.0	25,978

Copyright by Kasetsart University All rights reserved

Appendix Table B1 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
st70	70	675	728	728	728	728	728	735	711	711	728	711	725.0	735
eil76	76	538	558	571	571	569	571	569	558	558	585	558	567.8	585
pr76	76	108,159	114,363	114,363	112,389	114,363	112,389	112,220	118,616	114,363	112,765	112,220	113,981.2	118,616
rat99	99	1,211	1,290	1,278	1,335	1,335	1,335	1,278	1,278	1,335	1,335	1,278	1,311.0	1,335
kroA100	100	21,282	22,042	22,042	21,768	21,768	22,042	21,969	22,229	21,768	21,983	21,768	21,956.8	22,229
kroB100	100	22,141	23,320	23,264	23,674	22,876	23,320	23,357	22,755	22,755	22,876	22,755	23,133.0	23,674
kroC100	100	20,749	22,015	22,015	22,005	22,015	22,015	22,015	22,015	22,015	22,015	22,005	22,013.9	22,015
kroD100	100	21,294	22,857	22,857	23,379	23,416	23,416	23,278	23,416	23,399	23,399	22,857	23,268.6	23,416
kroE100	100	22,068	22,596	22,619	22,596	22,596	22,596	23,183	22,596	23,183	22,596	22,596	22,729.0	23,183
rd100	100	7,910	8,826	8,311	8,826	8,311	8,311	8,823	8,826	8,311	8,826	8,311	8,596.8	8,826
lin105	105	14,379	15,065	14,984	15,199	14,984	15,065	15,065	15,010	15,199	15,065	14,984	15,070.7	15,199
pr107	107	44,303	44,573	44,814	44,577	44,992	44,706	44,577	44,613	44,577	44,573	44,573	44,666.9	44,992
gr120	120	6,942	7,313	7,568	7,305	7,568	7,568	7,465	7,305	7,568	7,568	7,305	7,469.8	7,568
pr124	124	59,030	61,746	61,746	61,746	61,746	61,746	61,746	61,746	61,746	61,746	61,746	61,746.0	-
bier127	127	118,282	127,785	122,612	121,772	122,612	122,612	121,772	125,592	127,785	127,785	121,772	124,480.8	127,785
ch130	130	6,110	6,724	6,724	6,724	6,513	6,513	6,724	6,513	6,724	6,513	6,513	6,630.2	6,724

Copyright by Kasetsart University All rights reserved

Appendix Table B1 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
pr136	136	96,772	105,986	104,257	104,093	105,932	106,155	106,033	104,122	104,257	105,932	104,093	105,196.3	106,155
pr144	144	58,537	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754.0	-
ch150	150	6,528	6,779	6,779	6,779	6,779	6,759	6,779	6,759	6,779	6,759	6,759	6,772.3	6,779
kroA150	150	26,524	28,845	28,980	28,845	28,980	28,980	28,693	29,067	28,980	28,372	28,372	28,860.2	29,067
kroB150	150	26,130	27,484	27,686	27,743	27,686	27,686	27,799	27,686	27,686	27,743	27,484	27,688.8	27,799
pr152	152	73,682	75,778	75,778	75,778	75,972	76,227	76,227	76,227	75,972	75,774	75,774	75,970.3	76,227
u159	159	42,080	44,380	45,164	44,989	44,380	44,989	45,021	45,021	44,989	45,164	44,380	44,899.7	45,164
si175	175	21,407	21,675	21,656	21,675	21,667	21,680	21,678	21,667	21,666	21,675	21,656	21,671.0	21,680
d198	198	15,780	16,677	16,042	16,042	16,151	16,285	16,096	16,677	16,235	16,235	16,042	16,271.1	16,677
kroA200	200	29,368	31,011	30,684	30,684	30,457	30,839	30,475	30,684	30,684	30,684	30,457	30,689.1	31,011
kroB200	200	29,437	31,981	31,731	31,943	32,063	31,943	31,943	31,943	31,981	32,063	31,731	31,954.6	32,063
tsp225	225	3,916	4,191	4,191	4,151	4,191	4,191	4,191	4,137	4,191	4,151	4,137	4,176.1	4,191
ts225	225	126,643	132,995	132,995	132,995	132,995	132,995	130,742	132,995	133,844	132,995	130,742	132,839.0	133,844
pr226	226	80,369	84,515	85,148	84,515	83,184	85,184	83,998	83,998	84,515	84,686	83,184	84,415.9	85,184
gil262	262	2,378	2,624	2,645	2,623	2,623	2,580	2,645	2,624	2,645	2,598	2,580	2,623.0	2,645
a280	280	2,579	2,689	2,694	2,731	2,689	2,708	2,731	2,756	2,756	2,668	2,668	2,713.6	2,756

Copyright by Kasetsart University All rights reserved

Appendix Table B1 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
lin318	318	42,029	45,083	45,654	44,464	45,148	44,229	46,028	45,331	45,769	45,470	44,229	45,241.8	46,028
rd400	400	15,281	16,384	16,522	16,501	16,719	16,384	16,680	16,354	16,522	16,354	16,354	16,491.1	16,719
pcb442	442	50,778	53,799	54,620	54,620	54,776	54,544	54,417	54,417	53,862	54,214	53,799	54,363.2	54,776
d493	493	35,002	36,884	37,365	37,198	36,553	36,553	37,078	36,553	37,198	37,365	36,553	36,971.9	37,365
si535	535	48,450	48,930	48,899	48,771	48,820	48,881	48,845	48,845	48,909	48,888	48,771	48,865.3	48,930
pa561	561	2,763	2,969	2,950	2,950	2,967	2,932	2,923	2,924	2,957	2,949	2,923	2,946.8	2,969
d657	657	48,912	53,364	53,408	52,989	53,476	53,198	53,458	53,371	53,355	52,916	52,916	53,281.7	53,476
rat783	783	8,806	9,837	9,838	9,838	9,878	9,837	9,948	9,779	9,863	9,802	9,779	9,846.7	9,948
pr1002	1,002	259,045	281,515	279,781	276,226	278,818	281,515	277,739	281,515	277,228	277,357	276,226	279,077.1	281,515
d1291	1,291	50,801	54,615	54,136	55,839	53,926	54,615	54,136	53,924	54,615	55,102	53,924	54,545.3	55,839
fl1577	1,577	22,249	24,285	24,011	24,125	24,337	24,125	23,884	24,071	24,198	23,861	23,861	24,099.7	24,337
d2103	2,103	80,450	83,070	82,968	83,070	83,341	83,233	83,361	82,822	83,276	83,314	82,822	83,161.7	83,361

Copyright by Kasetsart University. All rights reserved

Appendix Table B2 The solutions of GPRTSP-1 algorithm

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
ulysses16	16	6859	6,909	7,005	6,909	6,875	6,875	6,982	7,076	6,870	6,913	6,870	6,934.9	7,076
gr17	17	2085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085	2,085.0	-
gr21	21	2707	2,877	2,801	2,707	2,877	2,803	2,877	2,877	2,877	2,801	2,707	2,833.0	2,877
ulysses22	22	7013	7,013	7,083	7,198	7,239	7,100	7,239	7,239	7,013	7,083	7,013	7,134.1	7,239
gr24	24	1272	1,272	1,332	1,345	1,332	1,332	1,352	1,332	1,332	1,272	1,272	1,322.3	1,352
fri26	26	937	961	955	961	961	961	961	961	961	961	955	960.3	961
bayg29	29	1610	1,668	1,660	1,676	1,743	1,649	1,628	1,635	1,660	1,654	1,628	1,663.7	1,743
bays29	29	2020	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035	2,035.0	-
dantzig42	42	699	743	741	743	699	741	736	741	701	736	699	731.2	743
swiss42	42	1273	1,274	1,274	1,274	1,273	1,285	1,274	1,274	1,285	1,274	1,273	1,276.3	1,285
att48	48	10628	11,196	10,918	11,057	11,196	10,918	11,196	10,954	11,021	10,954	10,918	11,045.6	11,196
gr48	48	5046	5,286	5,286	5,372	5,372	5,373	5,372	5,286	5,286	5,286	5,286	5,324.3	5,373
hk48	48	11461	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003	12,003.0	-
eil51	51	426	428	434	434	436	428	428	436	428	428	428	431.1	436
berlin52	52	7542	7,596	7,596	7,596	7,856	8,229	8,085	7,596	7,596	7,596	7,596	7,749.6	8,229
brazil58	58	25395	26,447	25,978	25,983	26,447	25,983	25,978	25,978	26,397	25,978	25,978	26,129.9	26,447

Copyright by Kasetsart University All rights reserved

Appendix Table B2 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
st70	70	675	740	724	729	728	729	735	735	735	735	724	732.2	740
eil76	76	538	558	569	558	580	558	558	558	561	571	558	563.4	580
pr76	76	108159	116,959	110,915	113,204	111,952	111,827	110,915	110,915	112,220	111,525	110,915	112,270.2	116,959
rat99	99	1211	1,278	1,335	1,278	1,345	1,335	1,278	1,335	1,327	1,278	1,278	1,309.9	1,345
kroA100	100	21282	22,391	21,873	22,450	22,599	22,042	22,413	22,670	21,572	21,768	21,572	22,197.6	22,670
kroB100	100	22141	22,778	23,582	22,808	23,243	23,243	23,320	22,792	23,297	22,876	22,778	23,104.3	23,582
kroC100	100	20749	22,015	22,015	22,015	22,015	22,015	21,613	22,015	22,015	22,015	21,613	21,970.3	22,015
kroD100	100	21294	22,857	23,416	23,416	22,911	23,416	23,416	23,399	23,399	23,278	22,857	23,278.7	23,416
kroE100	100	22068	22,596	23,236	22,596	23,348	22,570	22,596	22,709	22,596	22,596	22,570	22,760.3	23,348
rd100	100	7910	8,311	8,483	8,826	8,311	8,311	8,823	8,401	8,253	8,311	8,253	8,447.8	8,826
lin105	105	14379	15,369	15,045	15,199	15,065	14,897	15,065	15,369	14,897	14,837	14,837	15,082.6	15,369
pr107	107	44303	44,674	44,940	44,895	44,638	44,908	44,577	44,573	44,674	44,573	44,573	44,716.9	44,940
gr120	120	6942	7,435	7,305	7,324	7,305	7,296	7,568	7,568	7,568	7,568	7,296	7,437.4	7,568
pr124	124	59030	61,396	62,756	61,551	61,720	61,396	62,756	63,106	61,746	62,756	61,396	62,131.4	63,106
bier127	127	118282	124,327	121,576	122,345	124,898	125,823	126,626	123,153	122,952	122,110	121,576	123,756.7	126,626
ch130	130	6110	6,687	6,513	6,499	6,513	6,513	6,724	6,513	6,724	6,724	6,499	6,601.1	6,724

Copyright by Kasetsart University All rights reserved

Appendix Table B2 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
pr136	136	96772	106,748	106,494	105,989	105,386	106,519	105,005	106,589	106,973	106,214	105,005	106,213.0	106,973
pr144	144	58537	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754	60,754.0	-
ch150	150	6528	6,779	6,759	6,779	6,759	6,759	6,780	6,759	6,779	6,759	6,759	6,768.0	6,780
kroA150	150	26524	29,170	28,942	29,067	28,372	28,845	28,980	28,980	28,989	28,980	28,372	28,925.0	29,170
kroB150	150	26130	27,799	27,604	27,639	28,130	27,686	27,078	27,502	27,743	28,186	27,078	27,707.4	28,186
pr152	152	73682	76,302	75,774	75,774	76,304	76,227	75,402	75,679	75,774	75,774	75,402	75,890.0	76,304
u159	159	42080	45,164	45,581	45,792	44,989	45,164	44,380	45,375	44,989	45,375	44,380	45,201.0	45,792
si175	175	21407	21,667	21,675	21,648	21,666	21,675	21,675	21,666	21,667	21,675	21,648	21,668.2	21,675
d198	198	15780	16,671	16,151	16,133	16,677	16,235	16,165	16,208	16,235	16,151	16,133	16,291.8	16,677
kroA200	200	29368	31,094	30,693	30,684	30,684	30,773	30,477	30,693	30,695	30,659	30,477	30,716.9	31,094
kroB200	200	29437	32,260	31,803	31,324	31,603	31,800	31,713	31,380	31,294	31,603	31,294	31,642.2	32,260
tsp225	225	3916	4,191	4,199	4,199	4,211	4,191	4,211	4,211	4,187	4,191	4,187	4,199.0	4,211
ts225	225	126643	133,494	132,806	134,026	132,958	132,645	133,494	132,958	133,288	133,288	132,645	133,217.4	134,026
pr226	226	80369	83,580	84,015	82,782	85,054	83,878	84,958	83,111	84,936	82,594	82,594	83,878.7	85,054
gil262	262	2378	2,580	2,624	2,580	2,547	2,580	2,580	2,626	2,624	2,624	2,547	2,596.1	2,626
a280	280	2579	2,704	2,731	2,731	2,695	2,734	2,734	2,706	2,731	2,681	2,681	2,716.3	2,734

Copyright by Kasetsart University All rights reserved

Appendix Table B2 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
lin318	318	42029	45,947	45,148	45,660	44,719	44,719	44,563	44,464	45,194	45,506	44,464	45,102.2	45,947
rd400	400	15281	16,719	16,672	16,354	16,522	16,535	16,680	16,534	16,522	16,522	16,354	16,562.2	16,719
pcb442	442	50778	54,842	53,862	54,486	54,469	53,628	54,639	54,695	53,701	53,738	53,628	54,228.9	54,842
d493	493	35002	36,553	36,884	37,186	37,365	37,198	37,395	36,553	37,158	37,186	36,553	37,053.1	37,395
si535	535	48450	48,890	48,842	48,888	48,896	48,896	48,888	48,888	48,845	48,899	48,842	48,881.3	48,899
pa561	561	2763	2,962	2,924	2,928	2,969	2,955	2,982	2,949	2,949	2,960	2,924	2,953.1	2,982
d657	657	48912	53,198	53,411	53,198	53,411	52,916	53,723	53,603	53,318	53,408	52,916	53,354.0	53,723
rat783	783	8806	9,878	9,799	9,870	9,849	9,839	9,878	9,859	9,837	9,931	9,799	9,860.0	9,931
pr1002	1,002	259045	278,262	278,750	275,087	277,739	276,999	278,218	279,578	276,076	275,384	275,087	277,343.7	279,578
d1291	1,291	50801	54,615	54,506	53,776	55,142	54,800	53,987	54,800	54,247	54,615	53,776	54,498.7	55,142
fl1577	1,577	22249	24,741	24,605	24,057	24,017	24,205	24,795	24,358	24,482	24,394	24,017	24,406.0	24,795
d2103	2,103	80450	82,822	83,079	83,377	83,240	83,341	83,416	82,822	83,202	82,822	82,822	83,124.6	83,416

Copyright by Kasetsart University All rights reserved

Appendix Table B3 The solutions of genetic algorithm

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
ulysses16	16	6,859	6,859	6,859	6,870	6,870	6,859	6,859	6,859	6,859	6,859	6,859	6,861.4	6,870
gr17	17	2,085	2,090	2,085	2,085	2,085	2,090	2,085	2,090	2,090	2,090	2,085	2,087.8	2,090
gr21	21	2,707	2,707	2,707	2,707	2,707	2,801	2,707	2,707	2,707	2,707	2,707	2,717.4	2,801
ulysses22	22	7,013	7,013	7,013	7,114	7,013	7,013	7,013	7,092	7,013	7,092	7,013	7,041.8	7,114
gr24	24	1,272	1,286	1,272	1,272	1,286	1,272	1,272	1,286	1,272	1,272	1,272	1,276.7	1,286
fri26	26	937	937	937	937	937	937	937	937	937	937	937	937.0	-
bayg29	29	1,610	1,628	1,626	1,626	1,610	1,626	1,610	1,626	1,643	1,610	1,610	1,622.8	1,643
swiss42	29	2,020	2,026	2,026	2,034	2,026	2,026	2,026	2,034	2,026	2,026	2,026	2,027.8	2,034
att48	42	699	699	699	699	699	699	699	699	699	699	699	699.0	-
gr48	42	1,273	1,315	1,273	1,339	1,301	1,273	1,339	1,310	1,315	1,273	1,273	1,304.2	1,339
att48	48	10,628	10,841	10,753	10,792	10,953	10,628	10,653	11,015	10,904	10,725	10,628	10,807.1	11,015
gr48	48	5,046	5,116	5,055	5,072	5,072	5,046	5,114	5,055	5,074	5,153	5,046	5,084.1	5,153
hk48	48	11,461	11,461	11,824	11,470	11,830	11,941	11,879	11,722	11,906	11,574	11,461	11,734.1	11,941
eil51	51	426	439	435	448	441	459	443	436	434	444	434	442.1	459
berlin52	52	7,542	7,841	8,143	7,991	7,543	7,716	7,543	7,543	7,886	8,065	7,543	7,807.9	8,143
brazil58	58	25,395	25,946	27,132	25,895	25,722	25,643	25,475	25,643	25,946	26,428	25,475	25,981.1	27,132

Copyright by Kasetsart University All rights reserved

Appendix Table B3 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
st70	70	675	691	722	689	685	726	710	695	691	715	685	702.7	726
eil76	76	538	568	551	564	584	573	564	559	565	555	551	564.8	584
pr76	76	108,159	115,717	115,725	127,267	115,368	119,452	119,741	116,137	117,361	114,443	114,443	117,912.3	127,267
rat99	99	1,211	1,285	1,266	1,314	1,294	1,318	1,281	1,339	1,280	1,298	1,266	1,297.2	1,339
kroA100	100	21,282	21,828	22,837	24,451	21,758	22,081	22,108	22,799	22,944	22,475	21,758	22,586.8	24,451
kroB100	100	22,141	23,476	23,341	23,659	23,376	23,583	24,049	24,292	22,987	23,288	22,987	23,561.2	24,292
kroC100	100	20,749	22,099	22,270	22,816	22,431	21,644	22,698	22,217	22,352	22,242	21,644	22,307.7	22,816
kroD100	100	21,294	22,039	23,635	21,852	22,575	22,272	21,548	22,131	22,956	22,544	21,548	22,394.7	23,635
kroE100	100	22,068	22,715	23,415	22,805	23,094	23,204	22,500	23,914	22,907	23,379	22,500	23,103.7	23,914
rd100	100	7,910	8,545	8,466	8,403	8,459	8,511	8,668	8,305	8,470	8,617	8,305	8,493.8	8,668
lin105	105	14,379	15,731	15,816	15,354	14,755	15,426	15,771	15,403	15,479	15,389	14,755	15,458.2	15,816
pr107	107	44,303	45,040	46,295	45,517	45,262	44,899	45,692	45,365	45,113	44,509	44,509	45,299.1	46,295
gr120	120	6,942	7,374	7,352	7,231	7,137	7,450	7,365	7,637	7,616	7,415	7,137	7,397.4	7,637
pr124	124	59,030	61,257	60,616	61,486	62,432	64,192	61,637	61,246	61,295	59,928	59,928	61,565.4	64,192
bier127	127	118,282	123,427	124,016	131,673	124,322	123,397	126,045	129,596	123,026	124,642	123,026	125,571.6	131,673
ch130	130	6,110	6,605	6,651	6,685	6,788	6,346	6,663	6,742	6,553	6,949	6,346	6,664.7	6,949

Copyright by Kasetsart University All rights reserved

Appendix Table B3 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
pr136	136	96,772	104,548	105,528	105,903	102,477	106,986	109,677	103,651	101,425	103,523	101,425	104,857.6	109,677
pr144	144	58,537	61,470	65,498	59,179	65,371	60,128	62,391	61,338	59,758	59,780	59,179	61,657.0	65,498
ch150	150	6,528	6,909	7,090	7,188	7,088	7,162	7,408	7,258	7,160	7,143	6,909	7,156.2	7,408
kroA150	150	26,524	28,372	28,547	28,541	28,209	29,127	28,271	29,212	29,737	29,509	28,209	28,836.1	29,737
kroB150	150	26,130	28,143	28,229	28,242	28,308	27,973	28,148	27,971	27,589	28,520	27,589	28,124.8	28,520
pr152	152	73,682	78,162	76,375	77,626	76,842	77,911	77,581	77,784	78,025	77,913	76,375	77,579.9	78,162
u159	159	42,080	42,645	42,939	42,970	42,981	42,970	42,712	42,981	42,939	42,603	42,603	42,860.0	42,981
si175	175	21,407	21,482	21,834	21,587	21,693	21,628	21,642	21,693	21,492	21,605	21,482	21,628.4	21,834
d198	198	15,780	16,679	16,639	16,562	16,577	16,561	16,683	16,689	16,725	16,470	16,470	16,620.6	16,725
kroA200	200	29,368	32,673	32,496	31,829	32,617	30,735	32,486	32,157	33,302	32,578	30,735	32,319.2	33,302
kroB200	200	29,437	31,791	33,194	33,579	32,560	32,423	31,126	31,396	33,857	32,582	31,126	32,500.9	33,857
tsp225	225	3,916	4,316	4,115	4,226	4,179	4,273	4,409	4,294	4,069	4,221	4,069	4,233.6	4,409
ts225	225	126,643	138,736	134,174	132,902	133,624	137,536	134,210	138,298	131,972	135,569	131,972	135,224.6	138,736
pr226	226	80,369	85,848	88,169	85,986	91,139	87,566	88,135	88,345	87,794	87,012	85,848	87,777.1	91,139
gil262	262	2,378	2,705	2,636	2,666	2,648	2,673	2,721	2,657	2,664	2,596	2,596	2,662.9	2,721
a280	280	2,579	2,707	2,761	2,763	2,772	2,759	2,748	2,765	2,754	2,751	2,707	2,753.3	2,772

Copyright by Kasetsart University. All rights reserved

Appendix Table B3 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
lin318	318	42,029	47,552	46,839	46,417	47,573	46,852	46,442	48,050	47,137	47,193	46,417	47,117.2	48,050
rd400	400	15,281	18,862	18,760	18,077	18,119	18,837	18,815	18,569	18,319	18,474	18,077	18,536.9	18,862
pcb442	442	50,778	61,890	61,281	61,494	62,563	60,308	61,429	60,500	60,669	60,230	60,230	61,151.6	62,563
d493	493	35,002	40,651	40,368	40,624	41,209	40,276	41,742	40,893	41,304	40,501	40,276	40,840.9	41,742
si535	535	48,450	50,823	50,987	51,138	50,740	50,789	51,455	51,231	51,582	51,282	50,740	51,114.1	51,582
pa561	561	2,763	3,388	3,335	3,386	3,385	3,398	3,447	3,363	3,432	3,414	3,335	3,394.2	3,447
d657	657	48,912	69,699	69,563	68,001	70,129	67,541	68,244	69,386	69,362	70,616	67,541	69,171.2	70,616
rat783	783	8,806	17,091	16,753	16,847	16,694	17,020	16,706	16,364	16,678	16,929	16,364	16,786.9	17,091
pr1002	1,002	259,045	306,341	310,216	314,254	304,451	308,247	309,163	309,790	311,752	309,249	304,451	309,273.7	314,254
d1291	1,291	50,801	101,381	100,505	102,562	103,592	101,203	98,158	102,281	102,551	105,027	98,158	101,917.8	105,027
fl1577	1,577	22,249	40,309	39,215	36,621	40,149	37,167	38,702	40,304	38,272	37,624	36,621	38,707.0	40,309
d2103	2,103	80,450	116,715	119,709	118,272	118,605	120,549	121,799	126,158	121,857	117,386	116,715	120,116.7	126,158

Copyright by Kasetsart University All rights reserved

Appendix Table B4 The solutions of simulated annealing algorithm

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
ulysses16	16	6,859	6,978	6,875	6,895	6,972	6,895	6,911	6,870	6,875	6,972	6,870	6,915.9	6,978
gr17	17	2,085	2,090	2,085	2,120	2,085	2,154	2,154	2,085	2,085	2,177	2,085	2,115.0	2,177
gr21	21	2,707	2,707	2,707	2,803	2,707	3,000	2,707	2,997	2,707	2,707	2,707	2,782.4	3,000
ulysses22	22	7,013	7,377	7,120	7,013	7,294	7,087	7,105	7,294	7,174	7,100	7,013	7,173.8	7,377
gr24	24	1,272	1,317	1,364	1,376	1,445	1,402	1,317	1,355	1,373	1,345	1,317	1,366.0	1,445
fri26	26	937	937	970	1,079	965	1,044	1,052	1,008	960	985	937	1,000.0	1,079
bayg29	29	1,610	1,723	1,751	1,628	1,761	1,626	1,676	1,747	1,779	1,649	1,626	1,704.4	1,779
swiss42	29	2,020	2,151	2,108	2,137	2,144	2,098	2,150	2,031	2,179	2,190	2,031	2,132.0	2,190
att48	42	699	825	800	736	906	872	832	699	778	845	699	810.3	906
gr48	42	1,273	1,573	1,444	1,626	1,340	1,438	1,462	1,440	1,394	1,475	1,340	1,465.8	1,626
att48	48	10,628	11,686	11,952	13,261	11,688	11,771	13,575	12,490	11,323	13,991	11,323	12,415.2	13,991
gr48	48	5,046	6,223	6,289	6,408	6,017	6,169	5,917	6,228	6,209	5,756	5,756	6,135.1	6,408
hk48	48	11,461	14,314	12,143	13,342	14,118	14,472	12,991	13,219	12,636	14,375	12,143	13,512.2	14,472
eil51	51	426	466	472	463	458	470	492	459	469	470	458	468.8	492
berlin52	52	7,542	9,810	9,916	8,552	9,127	9,504	8,948	9,480	9,333	8,776	8,552	9,271.8	9,916
brazil58	58	25,395	28,913	27,582	31,113	31,690	28,187	26,947	27,175	28,138	29,503	26,947	28,805.3	31,690

Copyright by Kasetsart University All rights reserved

Appendix Table B4 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
st70	70	675	805	900	819	764	1,021	848	874	830	801	764	851.3	1,021
eil76	76	538	616	624	629	609	664	625	685	644	642	609	637.6	685
pr76	76	108,159	115,214	129,070	130,029	119,804	137,464	115,580	118,693	138,132	129,128	115,214	125,901.6	138,132
rat99	99	1,211	1,714	1,606	1,762	1,904	1,658	1,668	1,652	1,784	1,680	1,606	1,714.2	1,904
kroA100	100	21,282	31,112	30,824	32,827	31,875	30,963	29,254	27,579	31,668	31,496	27,579	30,844.2	32,827
kroB100	100	22,141	31,704	30,545	32,381	31,567	29,932	28,683	31,271	28,933	29,856	28,683	30,541.3	32,381
kroC100	100	20,749	27,831	29,615	30,261	32,832	28,644	28,328	32,930	30,096	30,260	27,831	30,088.6	32,930
kroD100	100	21,294	32,949	28,309	28,440	30,537	31,750	30,358	30,065	30,049	29,856	28,309	30,257.0	32,949
kroE100	100	22,068	29,714	26,893	30,408	32,954	34,237	30,819	29,008	30,383	31,272	26,893	30,632.0	34,237
rd100	100	7,910	12,248	13,404	12,157	13,080	11,580	13,984	12,265	11,531	10,634	10,634	12,320.3	13,984
lin105	105	14,379	23,454	22,231	21,974	23,252	25,690	24,557	20,958	21,658	20,138	20,138	22,656.9	25,690
pr107	107	44,303	52,847	51,091	48,301	48,698	52,697	52,786	58,679	54,735	53,270	48,301	52,567.1	58,679
gr120	120	6,942	11,636	11,091	10,132	10,475	10,492	10,517	11,079	11,354	10,728	10,132	10,833.8	11,636
pr124	124	59,030	69,146	73,846	68,632	73,453	69,238	74,089	71,632	71,186	65,280	65,280	70,722.4	74,089
bier127	127	118,282	136,350	140,903	134,580	142,283	133,823	139,535	136,867	144,475	136,380	133,823	138,355.1	144,475
ch130	130	6,110	10,566	9,501	9,154	9,808	10,352	10,811	9,431	9,214	9,326	9,154	9,795.9	10,811

Copyright by Kasetsart University All rights reserved

Appendix Table B4 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
pr136	136	96,772	111,319	115,814	107,468	113,231	108,636	117,748	116,840	110,147	111,942	107,468	112,571.7	117,748
pr144	144	58,537	63,000	64,488	65,261	67,602	61,486	61,954	63,849	63,202	63,043	61,486	63,765.0	67,602
ch150	150	6,528	12,136	11,700	11,538	12,298	10,984	11,577	10,789	10,188	10,635	10,188	11,316.1	12,298
kroA150	150	26,524	42,305	40,525	39,365	38,210	40,886	37,600	43,728	40,421	42,099	37,600	40,571.0	43,728
kroB150	150	26,130	40,459	38,824	40,847	38,504	44,321	37,932	39,120	43,140	41,974	37,932	40,569.0	44,321
pr152	152	73,682	86,987	85,789	80,300	80,035	86,642	88,812	85,861	84,999	87,866	80,035	85,254.6	88,812
u159	159	42,080	57,710	58,429	61,601	59,874	59,561	56,500	59,739	58,861	57,983	56,500	58,917.6	61,601
si175	175	21,407	25,537	25,694	26,106	25,634	26,054	26,092	24,839	26,227	25,306	24,839	25,721.0	26,227
d198	198	15,780	23,458	26,364	25,494	21,515	25,245	24,785	23,781	24,087	24,081	21,515	24,312.2	26,364
kroA200	200	29,368	50,664	47,469	50,733	46,383	49,653	43,255	44,291	47,383	45,847	43,255	47,297.6	50,733
kroB200	200	29,437	46,403	48,935	46,998	49,885	46,621	48,793	47,789	51,253	47,057	46,403	48,192.7	51,253
tsp225	225	3,916	7,740	7,012	8,250	7,825	7,009	8,286	8,046	6,855	7,169	6,855	7,576.9	8,286
ts225	225	126,643	145,811	155,995	146,939	148,571	157,559	159,309	156,807	149,027	145,998	145,811	151,779.6	159,309
pr226	226	80,369	91,213	96,902	92,588	89,338	99,716	91,197	92,588	90,826	87,909	87,909	92,475.2	99,716
gil262	262	2,378	4,666	4,963	5,117	5,169	4,716	4,702	4,771	5,232	5,098	4,666	4,937.1	5,232
a280	280	2,579	5,425	5,496	5,838	5,713	5,666	6,293	5,107	6,149	6,011	5,107	5,744.2	6,293

Copyright by Kasetsart University All rights reserved

Appendix Table B4 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Worst solution (units)
			1	2	3	4	5	6	7	8	9			
lin318	318	42,029	68,815	74,590	71,337	74,112	71,530	71,126	73,354	66,135	69,822	66,135	71,202.3	74,590
rd400	400	15,281	34,502	35,283	36,255	32,935	32,948	36,201	35,694	35,930	32,102	32,102	34,650.0	36,255
pcb442	442	50,778	84,341	82,987	79,162	85,453	80,031	83,689	84,290	90,671	88,536	79,162	84,351.1	90,671
d493	493	35,002	53,844	57,155	55,951	58,915	56,016	54,742	54,985	60,553	54,604	53,844	56,307.2	60,553
si535	535	48,450	71,813	70,571	71,571	71,871	70,763	72,851	70,274	71,903	70,959	70,274	71,397.3	72,851
pa561	561	2,763	6,342	6,041	6,110	6,204	6,057	5,935	7,024	6,572	6,710	5,935	6,332.8	7,024
d657	657	48,912	89,702	91,591	86,898	85,550	86,895	89,202	83,863	86,106	84,250	83,863	87,117.4	91,591
rat783	783	8,806	21,753	23,733	22,930	22,838	23,714	22,206	22,458	22,321	22,762	21,753	22,746.1	23,733
pr1002	1,002	259,045	354,302	337,891	365,545	358,093	345,295	339,815	352,628	347,908	343,725	337,891	349,466.9	365,545
d1291	1,291	50,801	112,400	110,488	112,124	110,147	109,083	115,966	112,804	114,027	102,560	102,560	111,066.6	115,966
fl1577	1,577	22,249	71,643	77,418	70,676	66,198	66,833	64,642	67,156	59,523	77,093	59,523	69,020.2	77,418
d2103	2,103	80,450	161,069	160,112	165,630	158,145	163,925	165,525	152,130	167,419	169,289	152,130	162,582.7	169,289

Copyright by Kasetsart University All rights reserved

Appendix Table B5 The running time of GPRTSP-0 algorithm

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
ulysses16	16	3.55	3.53	3.16	3.13	2.78	3.29	3.62	3.63	3.07	3.31
gr17	17	2.30	2.01	2.51	2.38	2.58	1.97	2.32	2.33	2.21	2.29
gr21	21	3.01	3.44	2.76	2.60	2.81	3.02	2.63	3.40	2.60	2.92
ulysses22	22	4.39	4.23	3.91	3.57	4.87	4.59	4.24	4.02	3.74	4.17
gr24	24	3.87	2.64	2.67	3.06	2.70	3.13	2.44	3.13	2.66	2.92
fri26	26	3.00	3.21	2.83	3.54	2.79	3.22	3.15	3.51	3.28	3.17
bayg29	29	3.71	4.11	3.59	3.86	3.90	3.31	3.70	3.67	4.08	3.77
bays29	29	3.67	3.33	3.51	3.89	3.46	3.37	3.66	3.90	3.62	3.60
dantzig42	42	5.51	6.07	5.44	6.70	6.85	6.67	6.37	6.44	6.43	6.27
swiss42	42	5.14	5.60	4.80	6.09	6.36	5.94	5.63	5.52	4.86	5.55
att48	48	6.90	6.85	6.90	6.90	6.92	7.10	7.12	6.85	6.93	6.94
gr48	48	10.89	9.80	11.19	10.26	10.51	12.51	10.67	9.84	11.09	10.75
hk48	48	6.87	6.88	6.97	6.91	7.18	6.85	6.90	6.87	6.93	6.93
eil51	51	7.09	6.60	6.48	7.10	8.83	6.40	7.85	7.04	8.20	7.29
berlin52	52	7.47	7.71	7.72	7.70	7.80	7.80	7.80	7.71	7.71	7.71
brazil58	58	8.66	8.74	8.76	8.71	8.65	8.63	8.69	8.75	8.77	8.71

Copyright by Kasetsart University All rights reserved

Appendix Table B5 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
st70	70	10.74	11.17	13.56	10.21	12.63	10.81	10.44	11.89	10.20	11.29
eil76	76	10.74	10.81	15.58	15.26	11.56	11.60	12.58	11.70	15.74	12.84
pr76	76	13.38	13.33	13.31	13.36	13.40	13.38	13.33	13.39	13.46	13.37
rat99	99	21.90	25.33	20.23	17.30	16.12	20.19	22.42	16.90	16.58	19.66
kroA100	100	18.38	18.37	20.60	20.72	18.29	18.34	19.39	20.64	18.37	19.23
kroB100	100	17.14	18.16	17.06	17.19	17.29	18.15	17.13	17.09	17.22	17.38
kroC100	100	18.30	18.49	17.24	18.48	18.37	18.55	18.49	18.47	18.40	18.31
kroD100	100	18.33	18.32	18.28	18.16	18.37	18.31	18.43	18.27	18.45	18.33
kroE100	100	18.39	19.49	18.13	18.28	18.26	19.44	18.21	19.45	18.27	18.66
rd100	100	18.36	19.37	18.23	19.76	19.69	18.14	18.30	19.52	18.35	18.86
lin105	105	19.35	22.19	19.59	22.12	19.53	19.27	20.70	19.47	19.57	20.20
pr107	107	18.61	18.33	19.77	19.92	19.84	19.91	20.06	19.73	18.77	19.44
gr120	120	24.62	24.60	24.45	24.49	24.44	24.51	24.61	26.31	24.48	24.72
pr124	124	23.59	23.63	23.79	23.62	23.64	23.61	23.73	23.70	23.62	23.66
bier127	127	24.47	24.54	26.43	24.29	24.24	26.34	24.16	24.39	24.52	24.82
ch130	130	27.48	27.62	27.65	31.77	31.62	43.45	31.71	27.82	31.66	31.20

Copyright by Kasetsart University. All rights reserved

Appendix Table B5 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
pr136	136	31.28	29.15	29.11	29.09	29.08	31.44	29.14	29.16	29.07	29.61
pr144	144	28.78	28.76	28.96	28.86	28.89	28.72	28.75	31.23	29.01	29.11
ch150	150	38.93	38.70	38.41	38.46	35.67	38.65	36.00	38.76	35.71	37.70
kroA150	150	41.19	38.32	41.45	38.61	38.56	35.57	38.39	38.64	35.83	38.51
kroB150	150	41.22	35.83	38.40	35.58	35.51	38.60	35.91	35.91	38.39	37.26
pr152	152	30.53	30.46	33.29	30.54	33.25	33.40	33.62	30.57	33.39	32.12
u159	159	35.46	35.49	35.82	35.87	35.65	35.18	35.44	35.24	35.51	35.52
si175	175	40.32	40.25	40.07	40.07	40.28	40.27	40.29	40.01	40.08	40.18
d198	198	52.23	57.06	56.84	47.06	51.78	56.44	52.32	47.28	47.45	52.05
kroA200	200	62.32	62.40	62.36	57.57	62.23	57.42	62.29	62.38	62.31	61.26
kroB200	200	57.53	57.21	57.16	57.23	57.43	57.44	56.89	57.34	57.20	57.27
tsp225	225	95.38	88.48	92.91	74.99	80.18	77.58	90.82	80.36	88.47	85.46
ts225	225	55.68	55.54	55.89	56.03	55.86	49.83	55.69	55.67	55.80	55.11
pr226	226	62.72	62.44	62.91	62.13	56.07	62.84	62.46	62.67	56.04	61.14
gil262	262	95.08	102.97	104.44	86.73	85.98	105.09	96.94	85.17	94.07	95.16
a280	280	102.87	101.58	101.10	102.95	105.79	124.03	94.62	94.35	97.58	102.76

Copyright by Kasetsart University. All rights reserved

Appendix Table B5 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
lin318	318	108.60	120.42	133.08	120.13	133.13	120.65	121.16	108.11	120.43	120.63
rd400	400	251.92	251.63	213.04	252.11	251.97	231.55	192.15	251.48	193.03	232.10
pcb442	442	251.05	226.26	228.09	177.61	249.69	227.04	226.16	202.49	226.28	223.85
d493	493	240.72	241.72	302.02	301.20	301.50	181.21	301.09	301.17	241.12	267.97
si535	535	392.72	392.08	427.35	429.14	465.58	357.18	356.99	391.59	392.62	400.58
pa561	561	392.30	372.30	315.92	317.67	277.49	379.51	322.11	330.82	335.95	338.23
d657	657	615.22	509.70	511.05	510.59	508.13	558.77	508.90	617.64	456.84	532.98
rat783	783	845.42	616.25	615.73	692.27	842.01	616.30	769.24	767.11	844.35	734.30
pr1002	1,002	1,701.02	1,584.10	1,330.53	955.73	1,703.72	1,201.04	1,702.57	1,328.97	1,326.12	1,425.98
d1291	1,291	2,365.29	1,525.60	1,530.98	1,743.05	2,355.76	1,532.49	2,156.80	2,362.58	1,746.58	1,924.35
fl1577	1,577	4,388.63	3,439.76	4,067.25	4,994.96	4,064.18	4,379.47	5,306.87	5,014.88	5,017.00	4,519.22
d2103	2,103	3,273.41	2,722.51	3,272.29	4,374.49	3,836.47	3,277.03	2,722.80	3,816.45	3,273.93	3,396.60

Copyright by Kasetsart University. All rights reserved

Appendix Table B6 The running time of GPRTSP-1 algorithm

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
ulysses16	16	21.57	18.43	18.64	21.55	21.59	18.55	19.20	21.92	20.27	20.19
gr17	17	17.61	16.38	17.45	18.07	17.36	16.86	17.26	16.00	16.67	17.07
gr21	21	18.37	21.58	21.97	19.66	20.84	18.36	19.58	18.86	18.49	19.75
ulysses22	22	22.57	21.02	21.60	23.36	22.05	22.12	22.46	21.20	21.41	21.98
gr24	24	18.58	19.31	19.82	17.96	19.13	18.24	18.73	18.62	19.90	18.92
fri26	26	20.27	20.11	18.45	18.45	18.19	18.90	19.23	20.03	18.58	19.13
bayg29	29	22.33	20.99	22.32	20.83	20.29	20.30	19.93	21.59	22.40	21.22
bays29	29	19.12	22.50	20.49	21.10	20.03	20.22	21.68	20.08	21.73	20.77
dantzig42	42	24.11	24.80	27.83	23.34	24.58	27.90	24.63	28.89	25.24	25.70
swiss42	42	26.47	24.07	25.60	25.62	27.05	26.47	27.53	29.27	28.80	26.77
att48	48	28.99	29.84	31.41	28.50	30.87	28.51	34.30	29.22	31.69	30.37
gr48	48	34.40	35.92	35.84	37.83	38.74	37.21	36.03	35.81	37.68	36.61
hk48	48	29.37	31.10	29.91	29.24	28.85	28.94	30.01	29.17	29.15	29.53
eil51	51	27.36	27.22	26.59	27.52	29.11	33.19	28.32	27.98	27.58	28.32
berlin52	52	30.42	29.02	29.47	29.46	31.40	30.77	29.86	29.63	30.94	30.11
brazil58	58	36.60	34.33	35.52	35.01	37.95	36.26	33.51	36.71	37.81	35.96

Copyright by Kasetsart University All rights reserved

Appendix Table B6 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
st70	70	42.42	42.55	43.71	46.63	38.41	37.41	47.19	38.27	48.44	42.78
eil76	76	47.62	46.31	36.72	40.98	39.65	41.42	36.80	35.89	37.84	40.36
pr76	76	46.24	45.90	46.72	47.78	46.86	48.73	46.02	45.61	48.00	46.87
rat99	99	58.59	52.39	51.45	57.85	52.03	60.38	62.23	55.56	53.37	55.98
kroA100	100	59.58	56.58	59.02	56.74	62.36	58.03	59.76	59.09	55.68	58.54
kroB100	100	64.79	64.00	62.66	62.12	62.38	61.13	61.69	60.52	61.57	62.32
kroC100	100	58.04	57.03	57.48	56.37	56.93	59.57	56.18	56.24	55.65	57.05
kroD100	100	60.23	62.26	59.60	59.96	60.86	60.34	61.13	59.62	59.38	60.38
kroE100	100	57.37	56.58	54.32	59.16	62.81	60.78	59.36	54.98	55.88	57.92
rd100	100	60.21	60.86	59.48	58.32	61.08	60.79	60.19	61.26	58.50	60.08
lin105	105	61.35	64.87	65.68	63.54	63.99	65.99	61.93	62.77	64.75	63.87
pr107	107	71.49	66.63	67.33	68.93	69.38	65.55	68.35	63.31	64.36	67.26
gr120	120	69.65	68.20	67.21	66.74	68.29	67.34	68.16	67.21	67.61	67.82
pr124	124	72.80	77.65	75.48	75.88	76.15	82.05	79.15	76.21	77.69	77.01
bier127	127	89.27	89.33	84.75	84.86	81.90	82.99	82.93	79.71	80.20	83.99
ch130	130	82.86	84.08	82.29	83.02	83.28	84.30	83.08	85.24	84.14	83.59

Copyright by Kasetsart University All rights reserved

Appendix Table B6 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
pr136	136	91.47	91.81	92.92	92.85	86.91	90.50	91.53	97.09	92.04	91.90
pr144	144	92.33	94.98	95.59	94.38	93.61	94.23	92.48	94.54	95.61	94.19
ch150	150	98.30	99.90	98.91	100.32	98.65	100.45	98.11	96.52	100.25	99.05
kroA150	150	101.78	105.60	103.86	98.84	100.78	102.57	103.19	100.50	99.82	101.88
kroB150	150	101.05	103.96	102.66	99.85	99.44	101.24	102.55	98.62	104.07	101.49
pr152	152	104.40	101.68	102.08	103.97	102.39	103.57	101.60	102.59	100.92	102.58
u159	159	97.49	101.37	97.82	99.63	96.30	98.11	96.38	98.09	96.24	97.94
si175	175	109.74	109.25	108.56	108.46	109.12	109.06	108.03	109.21	108.83	108.92
d198	198	128.96	128.06	130.52	131.87	129.25	130.11	128.74	128.48	128.52	129.39
kroA200	200	133.06	132.48	132.91	130.42	132.48	133.11	131.02	129.75	132.59	131.98
kroB200	200	133.02	131.96	131.73	131.90	133.50	133.84	133.29	134.84	136.49	133.40
tsp225	225	183.34	180.78	175.47	181.39	200.26	220.86	172.19	204.80	205.28	191.60
ts225	225	154.68	155.65	157.80	157.57	155.91	158.16	161.46	158.12	158.38	157.53
pr226	226	159.06	162.07	162.18	159.36	162.26	156.70	163.87	159.52	163.42	160.94
gil262	262	228.90	233.36	200.87	232.10	207.44	231.08	210.66	239.93	252.04	226.27
a280	280	246.78	262.86	241.63	273.42	220.65	285.93	231.04	244.18	224.43	247.88

Copyright by Kasetsart University All rights reserved

Appendix Table B6 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
lin318	318	300.22	297.83	272.59	272.37	268.83	273.44	270.46	272.01	271.47	277.69
rd400	400	395.81	394.03	399.00	391.44	398.13	392.84	398.91	395.77	393.94	395.54
pcb442	442	457.98	460.97	466.15	469.46	465.46	466.90	465.14	466.63	467.78	465.16
d493	493	559.32	558.33	561.01	560.48	559.84	561.33	557.45	562.54	561.26	560.17
si535	535	660.37	655.29	654.82	651.43	650.59	657.60	655.94	653.12	658.65	655.31
pa561	561	721.88	824.92	720.01	756.07	761.74	791.03	780.84	794.81	764.72	768.45
d657	657	996.99	939.08	937.30	936.68	997.81	940.76	935.38	938.40	935.69	950.90
rat783	783	1,281.10	1,286.09	1,287.20	1,286.62	1,363.02	1,285.21	1,283.87	1,280.14	1,153.14	1,278.49
pr1002	1,002	1,487.97	1,460.39	1,456.67	1,459.50	1,459.94	1,463.78	1,459.63	1,468.22	1,460.57	1,464.08
d1291	1,291	2,362.71	2,358.99	2,355.92	2,355.20	2,365.91	2,363.72	2,352.71	2,354.21	2,355.09	2,358.27
fl1577	1,577	3,434.32	3,481.43	3,468.96	3,468.21	3,473.82	3,471.82	3,472.25	3,468.80	3,471.86	3,467.94
d2103	2,103	5,989.25	5,993.23	5,990.95	5,978.71	5,974.18	6,130.33	6,003.11	5,972.55	5,967.82	6,000.01

Copyright by Kasetsart University All rights reserved

Appendix Table B7 The running time of genetic algorithm

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
ulysses16	16	22.87	22.50	22.47	22.47	22.49	22.42	22.55	22.48	22.44	22.52
gr17	17	22.56	24.71	28.62	28.52	28.55	28.65	28.53	28.56	28.52	27.47
gr21	21	29.36	29.40	29.48	29.32	29.40	29.58	29.34	29.45	29.20	29.39
ulysses22	22	40.83	42.18	42.32	42.27	42.35	42.57	42.35	42.40	42.43	42.19
gr24	24	29.11	29.21	29.22	29.24	29.17	29.20	29.08	29.30	29.23	29.20
fri26	26	60.34	60.21	60.30	60.41	60.33	60.22	60.27	60.21	60.38	60.30
bayg29	29	63.53	63.61	63.67	63.74	63.64	63.74	63.64	63.63	63.57	63.64
bays29	29	63.61	63.63	63.60	63.62	63.81	63.58	63.59	63.47	63.73	63.63
dantzig42	42	31.17	31.11	31.11	31.19	31.17	31.19	31.13	31.07	31.12	31.14
swiss42	42	31.43	31.12	31.28	31.20	31.14	31.14	31.23	31.23	31.18	31.22
att48	48	32.62	32.25	32.40	32.44	32.21	32.35	32.43	32.52	32.42	32.41
gr48	48	32.69	32.86	32.91	32.61	32.68	32.72	32.91	32.65	32.76	32.75
hk48	48	32.58	32.32	32.28	32.33	32.47	32.36	32.55	32.45	32.63	32.44
eil51	51	33.39	33.53	33.51	33.44	33.24	33.34	33.31	33.38	33.33	33.39
berlin52	52	33.38	33.30	33.38	33.37	33.30	33.38	33.25	33.35	33.35	33.34
brazil58	58	34.97	34.91	34.85	34.88	35.02	34.97	35.02	35.14	35.20	35.00

Copyright by Kasetsart University All rights reserved

Appendix Table B7 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
st70	70	38.02	37.89	37.77	37.82	37.89	37.70	37.85	37.81	37.82	37.84
eil76	76	39.60	39.42	39.39	39.46	39.40	39.34	39.43	39.50	39.41	39.44
pr76	76	39.27	39.08	39.06	39.15	39.22	39.12	39.10	39.26	39.27	39.17
rat99	99	46.30	45.73	45.58	45.67	45.66	46.30	45.68	45.87	45.57	45.82
kroA100	100	46.00	46.19	45.95	45.97	46.03	45.90	45.97	45.97	45.96	45.99
kroB100	100	45.54	45.50	45.45	45.48	45.38	45.53	45.59	45.56	45.53	45.51
kroC100	100	45.90	45.55	45.29	45.73	45.93	45.89	45.98	46.29	45.44	45.78
kroD100	100	45.78	46.26	45.85	45.55	45.82	45.78	45.79	45.87	45.82	45.84
kroE100	100	45.48	45.86	45.65	45.94	45.74	45.46	45.48	45.64	45.94	45.69
rd100	100	45.48	45.70	45.69	45.68	45.49	45.61	45.55	45.77	45.63	45.62
lin105	105	47.56	47.14	47.14	47.12	47.13	47.09	47.02	47.12	47.08	47.15
pr107	107	47.39	47.66	47.18	47.74	47.58	47.25	47.65	47.74	47.60	47.53
gr120	120	51.33	51.25	51.33	51.26	51.31	51.30	51.28	51.42	51.33	51.31
pr124	124	52.37	52.36	52.26	52.64	53.02	52.24	52.30	52.34	52.65	52.47
bier127	127	53.60	53.70	53.50	53.39	53.20	53.51	53.27	53.31	53.48	53.44
ch130	130	54.38	55.31	54.56	55.17	54.42	54.41	54.36	54.57	54.82	54.67

Copyright by Kasetsart University All rights reserved

Appendix Table B7 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
pr136	136	56.75	56.03	56.40	56.72	56.25	56.16	56.60	55.90	56.17	56.33
pr144	144	59.56	58.60	58.80	58.68	58.68	58.66	58.79	58.61	58.59	58.77
ch150	150	61.18	61.60	61.06	60.71	60.84	61.10	60.96	61.36	61.49	61.14
kroA150	150	60.91	60.81	60.78	60.72	60.61	60.74	60.81	60.92	60.86	60.80
kroB150	150	60.66	60.63	60.46	60.65	61.13	61.01	61.27	60.88	60.77	60.83
pr152	152	61.41	61.36	61.26	61.41	61.27	61.40	61.38	61.34	61.27	61.34
u159	159	63.99	63.74	63.78	63.76	63.75	63.68	63.86	63.61	63.83	63.78
si175	175	69.74	69.49	69.48	69.75	69.95	69.58	69.98	69.97	70.24	69.80
d198	198	77.98	77.89	78.12	77.90	77.87	78.05	77.98	78.00	78.15	77.99
kroA200	200	78.72	78.76	78.74	78.80	79.02	78.51	79.24	78.74	78.53	78.79
kroB200	200	78.40	78.72	79.27	78.97	78.82	79.42	78.94	79.53	78.88	78.99
tsp225	225	88.83	88.79	88.66	88.85	88.58	89.02	88.83	88.18	88.93	88.74
ts225	225	88.85	88.79	88.78	88.79	89.08	88.33	88.57	88.65	88.98	88.76
pr226	226	89.27	89.10	89.65	88.95	89.72	89.34	89.16	89.13	88.74	89.23
gil262	262	103.97	103.92	104.20	104.04	103.70	103.79	103.39	103.27	104.02	103.81
a280	280	111.34	111.16	111.02	111.11	111.16	111.44	111.21	111.18	111.20	111.20

Copyright by Kasetsart University All rights reserved

Appendix Table B7 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
lin318	318	129.24	128.96	128.81	129.25	128.95	129.00	129.01	128.77	129.03	129.00
rd400	400	171.89	172.12	172.36	172.32	171.98	172.02	171.65	171.50	172.74	172.07
pcb442	442	195.35	196.42	196.32	196.13	195.72	195.87	195.90	195.64	196.05	195.93
d493	493	223.95	224.75	224.44	224.81	223.84	224.21	223.92	224.84	224.45	224.36
si535	535	252.06	252.19	251.74	251.98	252.62	251.38	252.32	252.04	251.86	252.02
pa561	561	268.45	268.71	267.85	268.67	268.32	268.74	268.15	267.94	268.67	268.39
d657	657	337.93	337.26	337.43	337.32	338.11	337.53	337.85	337.30	337.96	337.63
rat783	783	436.92	437.45	437.91	437.30	438.11	438.40	437.36	438.14	437.17	437.64
pr1002	1,002	633.31	632.87	631.41	632.22	633.11	633.50	633.50	632.53	633.47	632.88
d1291	1,291	964.19	962.06	962.76	962.71	964.95	961.85	961.33	960.37	963.29	962.61
fl1577	1,577	1,343.10	1,343.61	1,345.15	1,343.91	1,344.26	1,342.43	1,343.63	1,343.20	1,343.26	1,343.62
d2103	2,103	2,263.23	2,256.53	2,267.25	2,261.53	2,269.13	2,259.06	2,265.25	2,263.86	2,267.13	2,263.66

Copyright by Kasetsart University All rights reserved

Appendix Table B8 The running time of simulated annealing algorithm

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
ulysses16	16	6.93	6.70	6.75	6.84	6.79	6.83	6.71	6.78	6.78	6.79
gr17	17	7.17	0.15	6.91	0.16	6.96	6.94	0.08	0.14	6.99	3.94
gr21	21	0.10	0.11	8.10	0.14	8.96	0.14	8.82	0.17	0.25	2.98
ulysses22	22	9.43	9.31	0.19	9.21	9.19	9.42	9.09	9.37	9.15	8.26
gr24	24	10.05	89.47	10.04	10.10	9.40	10.00	10.08	9.95	84.43	27.06
fri26	26	1.67	9.90	10.76	10.04	103.15	10.70	103.26	97.40	10.65	39.72
bayg29	29	12.42	12.23	12.48	12.13	12.36	12.36	12.24	12.44	12.37	12.34
bays29	29	12.40	12.52	12.41	12.32	12.17	12.23	12.14	12.38	11.37	12.22
dantzig42	42	18.36	266.18	274.84	16.81	264.25	267.59	14.12	277.06	17.53	157.42
swiss42	42	18.32	18.73	18.26	18.00	17.55	18.40	17.27	16.08	17.35	17.77
att48	48	21.71	21.40	22.14	21.97	21.79	21.74	21.36	21.56	21.48	21.68
gr48	48	18.47	21.91	18.78	21.85	21.82	21.63	19.89	20.86	21.60	20.76
hk48	48	21.68	22.03	21.95	21.95	21.62	22.01	22.12	21.48	21.98	21.87
eil51	51	402.82	377.74	398.03	20.91	395.45	403.29	399.61	399.22	406.82	355.99
berlin52	52	23.51	23.02	23.70	22.56	23.08	23.50	23.54	23.18	23.35	23.27
brazil58	58	25.81	25.39	23.77	25.96	25.06	24.63	26.07	25.25	25.12	25.23

Copyright by Kasetsart University All rights reserved

Appendix Table B8 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
st70	70	758.83	735.09	737.08	736.94	25.40	737.96	30.09	755.86	747.56	584.98
eil76	76	917.47	891.39	893.36	885.44	872.88	920.72	31.93	925.14	889.34	803.07
pr76	76	36.97	36.47	36.21	36.61	36.46	36.03	36.84	36.70	36.55	36.54
rat99	99	46.11	43.81	45.18	43.72	44.03	45.74	47.23	44.68	45.88	45.15
kroA100	100	48.26	47.89	48.57	49.01	49.44	49.16	49.02	48.56	48.34	48.69
kroB100	100	48.45	48.15	48.19	48.90	47.30	48.49	46.85	49.58	48.26	48.24
kroC100	100	48.27	47.69	48.62	49.34	48.18	48.48	48.37	48.68	48.15	48.42
kroD100	100	49.57	49.72	48.29	49.97	50.09	50.11	50.27	49.08	48.91	49.56
kroE100	100	48.07	49.10	48.58	48.38	48.71	49.05	48.13	47.43	48.54	48.44
rd100	100	47.57	46.96	48.20	46.19	46.56	48.06	47.69	48.11	46.34	47.30
lin105	105	51.04	50.08	49.43	50.06	50.24	50.05	49.11	50.83	48.82	49.96
pr107	107	50.62	47.07	50.95	50.07	51.51	50.60	50.57	51.06	51.52	50.44
gr120	120	56.23	56.59	54.31	54.34	56.58	57.62	55.91	56.97	53.29	55.76
pr124	124	62.68	65.48	64.15	63.84	62.43	62.85	62.76	64.09	64.46	63.64
bier127	127	65.84	63.84	64.96	64.79	64.23	65.01	63.76	63.94	63.74	64.46
ch130	130	62.69	62.15	61.69	61.49	59.50	62.38	60.67	62.67	59.66	61.43

Copyright by Kasetsart University. All rights reserved

Appendix Table B8 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
pr136	136	71.40	69.89	68.75	69.64	69.31	68.17	69.67	68.69	69.51	69.45
pr144	144	77.58	76.80	76.70	76.85	78.27	74.97	78.61	77.13	77.13	77.12
ch150	150	75.62	76.82	75.52	76.84	78.35	74.90	76.63	77.30	76.07	76.45
kroA150	150	79.51	82.06	78.91	80.36	79.55	80.66	78.15	80.39	80.81	80.05
kroB150	150	79.94	79.82	80.09	80.81	78.71	81.17	79.88	79.58	79.43	79.94
pr152	152	78.79	79.42	79.69	78.54	79.95	79.46	79.73	78.78	78.26	79.18
u159	159	84.16	83.24	84.81	79.51	82.48	82.27	82.71	84.14	82.66	82.89
si175	175	79.28	81.09	77.05	78.25	79.06	76.96	81.86	82.05	81.56	79.68
d198	198	108.02	107.08	105.24	103.80	106.13	107.72	107.35	106.86	108.62	106.76
kroA200	200	111.03	108.67	111.46	109.41	108.10	110.78	108.05	108.87	109.18	109.51
kroB200	200	110.35	109.47	109.12	110.51	107.44	109.18	111.23	109.63	109.73	109.63
tsp225	225	125.94	120.32	123.72	122.26	125.41	120.70	121.16	120.44	121.41	122.37
ts225	225	127.76	134.66	132.18	135.99	130.13	124.72	127.58	124.70	130.91	129.85
pr226	226	120.62	129.53	121.72	119.46	121.25	113.69	124.48	117.61	118.39	120.75
gil262	262	146.78	139.37	139.70	138.49	142.86	139.89	142.94	144.55	140.01	141.62
a280	280	152.74	155.02	150.21	154.89	150.87	147.35	145.69	153.29	158.01	152.01

Copyright by Kasetsart University All rights reserved

Appendix Table B8 (Continued)

Instances	# of nodes	Trial									Average running time (seconds)
		1	2	3	4	5	6	7	8	9	
lin318	318	201.43	206.50	212.62	212.81	206.61	205.68	210.17	213.25	211.13	208.91
rd400	400	260.87	258.51	259.80	263.33	253.07	252.09	262.99	264.71	256.34	259.08
pcb442	442	311.39	326.15	305.02	315.45	307.15	320.95	310.34	305.53	313.75	312.86
d493	493	350.94	340.90	353.34	358.59	347.01	350.63	347.48	348.22	363.35	351.16
si535	535	327.40	313.52	292.55	306.63	305.69	303.53	315.85	310.45	310.36	309.55
pa561	561	307.50	3,684.10	280.80	291.72	2,005.08	279.27	287.35	286.26	274.89	855.22
d657	657	522.44	530.20	522.17	533.30	538.76	520.66	528.45	520.53	532.79	527.70
rat783	783	561.11	557.36	549.52	580.78	548.56	557.78	547.81	568.95	580.20	561.34
pr1002	1,002	1,228.48	1,197.01	1,163.01	1,176.97	1,195.06	1,179.19	1,188.30	1,164.07	1,174.93	1,185.22
d1291	1,291	1,543.54	1,435.85	1,466.93	1,406.09	1,493.38	1,481.78	1,512.92	1,458.38	1,422.93	1,469.09
fl1577	1,577	1,625.34	1,523.70	1,541.11	1,572.47	1,703.80	1,624.99	1,622.22	1,610.44	1,626.95	1,605.67
d2103	2,103	4,180.33	3,968.79	4,189.99	4,160.46	3,939.96	4,115.69	4,131.03	4,097.59	3,967.73	4,083.51

Copyright by Kasetsart University All rights reserved

Appendix Table B9 The solutions of GPRTSP-1 algorithm with running time limitation

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Average Deviation $\bar{D}$ (%)
			1	2	3	4	5	6	7	8	9			
eil51	51	426	438	434	436	428	428	428	434	428	428	428	431	1.25
berlin52	52	7,542	7,596	7,596	7,596	7,856	7,596	8,085	7,596	7,596	7,596	7,596	7,679	1.82
st70	70	675	728	729	728	728	728	711	728	728	711	711	724	7.31
eil76	76	538	558	558	558	558	571	558	571	571	570	558	564	4.77
kroA100	100	21,282	21,884	22,024	21,988	22,083	22,143	22,472	21,873	21,577	22,472	21,577	22,057	3.64
kroB100	100	22,141	23,295	22,850	23,674	23,330	23,093	23,175	23,240	23,217	22,808	22,808	23,187	4.72
kroC100	100	20,749	22,015	22,015	21,917	22,015	22,015	22,015	21,917	21,907	22,015	21,907	21,981	5.94
kroD100	100	21,294	23,416	23,416	23,278	23,416	23,386	22,857	23,416	23,416	22,857	22,857	23,273	9.29
kroE100	100	22,068	22,897	22,596	22,906	22,472	22,614	22,472	22,897	23,154	22,596	22,472	22,734	3.02
rd100	100	7,910	8,311	8,401	8,311	8,311	8,826	8,311	8,311	8,804	8,311	8,311	8,433	6.61
lin105	105	14,379	15,469	14,897	15,199	15,065	14,897	15,065	15,065	15,199	15,199	14,897	15,117	5.13
pr107	107	44,303	44,577	44,577	44,577	44,577	44,625	44,613	44,577	44,682	44,479	44,479	44,587	0.64
pr124	124	59,030	61,551	63,080	63,080	61,396	62,756	63,080	61,720	63,080	61,396	61,396	62,349	5.62
bier127	127	118,282	121,014	124,604	122,998	121,720	122,916	121,576	124,176	125,259	122,966	121,014	123,025	4.01
ch130	130	6,110	6,513	6,724	6,513	6,724	6,414	6,513	6,724	6,513	6,484	6,414	6,569	7.51
pr136	136	96,772	104,158	105,346	106,414	105,253	106,155	105,483	105,919	104,477	106,085	104,158	105,477	9.00

Copyright by Kasetsart University All rights reserved

Appendix Table B9 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Average Deviation $\bar{D}$ (%)
			1	2	3	4	5	6	7	8	9			
ch150	150	6,528	6,767	6,768	6,767	6,767	6,759	6,779	6,767	6,779	6,767	6,759	6,769	3.69
kroA150	150	26,524	28,980	28,927	29,086	28,845	29,032	29,170	28,980	29,086	28,980	28,845	29,010	9.37
kroB150	150	26,130	27,639	27,492	27,662	27,716	27,659	27,685	28,047	27,826	27,658	27,492	27,709	6.04
pr152	152	73,682	75,774	75,488	75,679	75,774	76,340	75,774	75,488	75,774	76,227	75,488	75,813	2.89
kroA200	200	29,368	30,684	30,693	30,741	30,610	30,471	30,306	30,811	30,910	31,329	30,306	30,728	4.63
kroB200	200	29,437	31,324	31,909	31,757	32,038	31,943	31,880	31,468	32,260	32,063	31,324	31,849	8.19
lin318	318	42,029	45,506	45,148	45,948	44,464	45,493	44,908	45,194	45,148	44,464	44,464	45,141	7.41
pcb442	442	50,778	53,862	53,134	54,438	55,097	54,806	54,417	54,417	54,580	54,201	53,134	54,328	6.99
d657	657	48,912	53,627	53,576	53,600	52,717	53,603	53,476	52,952	53,118	53,232	52,717	53,322	9.02
rat783	783	8,806	9,898	9,910	9,810	9,958	9,870	9,963	9,870	9,863	9,827	9,810	9,885	12.26
pr1002	1,002	259,045	281,449	275,945	274,527	283,492	281,894	280,458	292,029	284,847	285,212	274,527	282,206	8.94
d1291	1,291	50,801	55,678	57,904	56,359	54,640	56,309	54,640	55,647	55,640	56,370	54,640	55,910	10.06
fl1577	1,577	22,249	25,965	25,334	25,836	26,014	25,648	25,389	25,648	25,668	25,251	25,251	25,639	15.24
d2103	2,103	80,450	84,610	84,740	84,947	84,300	84,339	84,039	84,599	84,319	84,319	84,039	84,468	4.99

Copyright by Kasetsart University. All rights reserved

Appendix Table B10 The solutions of genetic algorithm with running time limitation

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Average Deviation $\bar{D}$ (%)
			1	2	3	4	5	6	7	8	9			
eil51	51	426	440	439	433	431	428	439	442	437	428	428	435	2.16
berlin52	52	7,542	7,866	7,960	7,741	7,776	8,018	7,997	7,972	7,543	7,776	7,543	7,850	4.08
st70	70	675	690	684	682	699	691	700	676	688	701	676	690	2.24
eil76	76	538	549	560	556	556	544	568	552	547	550	544	554	2.89
kroA100	100	21,282	21,960	21,673	22,518	21,749	22,023	21,661	21,978	21,609	21,848	21,609	21,891	2.86
kroB100	100	22,141	22,495	22,377	23,379	22,502	22,492	22,613	23,122	22,916	23,304	22,377	22,800	2.98
kroC100	100	20,749	21,234	21,200	21,260	21,305	21,716	21,035	21,001	22,159	21,214	21,001	21,347	2.88
kroD100	100	21,294	22,518	22,045	21,294	21,614	21,558	22,535	22,063	22,493	21,425	21,294	21,949	3.08
kroE100	100	22,068	22,602	22,739	22,547	22,475	22,279	22,796	22,302	23,137	22,967	22,279	22,649	2.63
rd100	100	7,910	8,598	8,632	8,112	8,049	8,338	8,475	7,979	8,349	8,090	7,979	8,291	4.82
lin105	105	14,379	14,751	14,812	15,266	15,100	15,435	15,015	15,416	14,507	15,014	14,507	15,035	4.56
pr107	107	44,303	45,062	44,888	45,209	44,902	44,640	44,797	44,844	45,105	45,338	44,640	44,976	1.52
pr124	124	59,030	60,838	60,547	59,850	60,212	60,038	60,396	60,049	61,468	60,690	59,850	60,454	2.41
bier127	127	118,282	126,894	126,978	124,619	127,679	120,238	124,886	122,354	127,059	127,294	120,238	125,333	5.96
ch130	130	6,110	6,381	6,591	6,608	6,448	6,413	6,331	6,294	6,287	6,678	6,287	6,448	5.53
pr136	136	96,772	99,953	100,514	99,731	99,277	99,769	99,430	98,382	102,620	98,311	98,311	99,776	3.10

Copyright by Kasetsart University All rights reserved

Appendix Table B10 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Average Deviation $\bar{D}$ (%)
			1	2	3	4	5	6	7	8	9			
ch150	150	6,528	7,028	7,070	6,945	6,851	7,207	6,781	7,167	6,861	7,004	6,781	6,990	7.08
kroA150	150	26,524	28,733	27,638	27,252	27,914	27,549	27,424	28,481	28,026	28,336	27,252	27,928	5.29
kroB150	150	26,130	28,356	27,306	27,358	27,645	27,043	28,281	27,532	27,485	27,328	27,043	27,593	5.60
pr152	152	73,682	75,925	74,709	74,355	75,853	76,574	75,126	74,866	74,699	74,298	74,298	75,156	2.00
kroA200	200	29,368	30,504	32,460	31,057	31,975	32,678	32,088	31,293	31,036	31,543	30,504	31,626	7.69
kroB200	200	29,437	30,941	31,481	32,130	32,091	31,716	32,142	31,621	32,558	31,213	30,941	31,766	7.91
lin318	318	42,029	45,667	47,220	47,205	46,026	46,401	46,088	46,080	47,293	46,304	45,667	46,476	10.58
pcb442	442	50,778	57,703	56,075	57,036	59,673	56,057	57,337	57,384	57,096	55,834	55,834	57,133	12.51
d657	657	48,912	56,460	56,027	57,110	56,214	58,104	58,782	57,294	56,654	55,755	55,755	56,933	16.40
rat783	783	8,806	12,358	12,210	12,382	12,388	12,291	12,265	12,164	12,210	12,572	12,164	12,316	39.85
pr1002	1,002	259,045	313,278	301,431	296,829	304,477	299,552	304,169	302,790	306,358	301,839	296,829	303,414	17.13
d1291	1,291	50,801	104,311	100,514	100,664	104,798	108,163	104,314	102,630	104,961	103,713	100,514	103,785	104.30
fl1577	1,577	22,249	40,269	40,239	40,689	40,994	43,178	39,793	41,801	40,983	41,372	39,793	41,035	84.44
d2103	2,103	80,450	124,753	126,761	127,754	131,444	129,649	126,257	131,923	130,905	130,061	124,753	128,834	60.14

Copyright by Kasetsart University. All rights reserved

Appendix Table B11 The solutions of simulated annealing algorithm with running time limitation

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Average Deviation $\bar{D}$ (%)
			1	2	3	4	5	6	7	8	9			
eil51	51	426	467	479	494	475	448	472	468	471	464	448	471	10.54
berlin52	52	7,542	9,580	8,903	9,223	8,449	9,797	9,770	8,994	9,887	9,127	8,449	9,303	23.35
st70	70	675	891	794	842	883	855	849	935	848	866	794	863	27.79
eil76	76	538	662	630	613	635	578	613	630	626	653	578	627	16.48
kroA100	100	21,282	32,635	32,230	33,005	29,257	28,278	28,838	28,707	31,759	34,139	28,278	30,983	45.58
kroB100	100	22,141	33,124	31,358	32,405	31,660	28,910	31,478	29,460	30,240	32,953	28,910	31,288	41.31
kroC100	100	20,749	29,640	28,322	30,008	28,076	30,508	27,556	29,760	32,067	28,647	27,556	29,398	41.69
kroD100	100	21,294	34,865	31,569	32,957	30,305	31,361	31,437	32,204	34,728	32,708	30,305	32,459	52.43
kroE100	100	22,068	32,551	29,194	29,239	32,300	30,326	32,633	28,296	28,491	27,484	27,484	30,057	36.20
rd100	100	7,910	12,016	11,886	10,672	11,001	11,827	12,484	11,661	12,199	11,343	10,672	11,677	47.62
lin105	105	14,379	21,837	23,520	25,185	22,174	22,091	21,358	21,098	24,549	24,237	21,098	22,894	59.22
pr107	107	44,303	52,349	47,229	52,693	49,934	50,999	50,602	51,715	56,590	50,057	47,229	51,352	15.91
pr124	124	59,030	76,644	70,112	74,214	68,014	66,797	70,365	75,691	71,392	68,313	66,797	71,282	20.76
bier127	127	118,282	135,158	139,448	143,666	140,634	140,482	140,944	141,170	134,602	138,461	134,602	139,396	17.85
ch130	130	6,110	10,269	9,494	9,271	10,356	11,014	10,844	11,033	10,672	10,137	9,271	10,343	69.29
pr136	136	96,772	113,387	116,240	113,773	113,551	125,371	115,755	111,256	111,982	112,997	111,256	114,924	18.76

Copyright by Kasetsart University All rights reserved

Appendix Table B11 (Continued)

Instances	# of nodes	Optimal solution (units)	Trial									Best solution (units)	Average $\hat{\hat{y}}^*$ (units)	Average Deviation $\bar{D}$ (%)
			1	2	3	4	5	6	7	8	9			
ch150	150	6,528	10,948	12,023	10,535	10,966	11,937	11,241	11,445	11,870	11,878	10,535	11,427	75.05
kroA150	150	26,524	39,162	40,762	41,643	43,031	40,357	41,210	39,797	40,885	40,334	39,162	40,798	53.81
kroB150	150	26,130	40,043	37,432	40,328	38,802	41,216	38,284	36,790	35,847	40,585	35,847	38,814	48.54
pr152	152	73,682	79,993	88,110	86,917	83,357	86,270	85,717	87,954	77,758	87,952	77,758	84,892	15.21
kroA200	200	29,368	45,111	50,682	46,311	43,944	46,281	49,029	45,487	51,055	44,681	43,944	46,953	59.88
kroB200	200	29,437	47,972	48,760	48,531	52,041	46,035	48,723	47,908	46,574	50,143	46,035	48,521	64.83
lin318	318	42,029	69,000	67,675	75,425	70,426	73,293	64,435	74,772	69,532	71,788	64,435	70,705	68.23
pcb442	442	50,778	80,091	83,764	89,723	78,516	81,681	79,150	80,336	87,333	78,708	78,516	82,145	61.77
d657	657	48,912	87,819	87,492	88,315	83,961	87,876	84,077	88,097	84,696	81,942	81,942	86,031	75.89
rat783	783	8,806	22,653	21,714	23,236	23,224	21,989	21,477	21,190	22,638	22,290	21,190	22,268	152.87
pr1002	1,002	259,045	343,878	350,890	328,973	341,898	347,883	331,201	351,900	351,562	334,908	328,973	342,566	32.24
d1291	1,291	50,801	111,510	106,582	111,710	113,287	112,269	111,140	117,374	110,058	104,792	104,792	110,969	118.44
fl1577	1,577	22,249	81,904	83,990	77,791	80,764	77,891	68,896	84,122	78,667	69,061	68,896	78,121	251.12
d2103	2,103	80,450	184,258	176,044	177,314	173,918	186,005	181,776	192,460	179,707	187,999	173,918	182,165	126.43

Copyright by Kasetsart University All rights reserved

CURRICULUM VITAE

NAME : Mr. Wanatchapong Kongkaew

BIRTH DATE : April 19, 1982

BIRTH PLACE : Trang, Thailand

EDUCATION :

YEAR	INSTITUTE	DEGREE
2004	Kasetsart Univ.	B.Eng. (Industrial Engineering)
2007	Prince of Songkla Univ.	M.Eng. (Industrial and Systems Engineering)

POSITION : Lecturer

WORK PLACE : Faculty of Engineering, Prince of Songkla University

SCHOLARSHIP : Ph.D. 50% Scholarship, Prince of Songkla University 2009-2012
Faculty of Engineering, Kasetsart University 2013

PUBLICATIONS:

Kongkaew, W. and J. Pichitlamken. 2012. A Gaussian process regression model for the traveling salesman problem. **Journal of Computer Science** 8 (10): 1749-1758.

Kongkaew, W. and J. Pichitlamken. 2012. A combined procedure based on Gaussian process regression for the traveling salesman problem, pp. 2354-2363. *In* **Proceedings of the IE Network Conference 2012.** Sripatum University, Bangkok, Thailand.

Kongkaew, W. and J. Pichitlamken. 2012. Empirical study of optimal prediction of traveling salesman problem using Gaussian process regression, pp. 268-274. *In* **Proceedings of the 8th International Conference on Intelligent Manufacturing and Logistics Systems.** Ubon Ratchathani University, Ubon Ratchatani, Thailand.

Copyright by Kasetsart University All rights reserved