Journal of Computer Science 8 (10): 1749-1758, 2012
ISSN 1549-3636

A Gaussian Process Regression Model for the Traveling Salesman Problem

Wanatchapong Kongkaew and Juta Pichitlamken
Department of Industrial Engineering,
Faculty of Engineering, Kasetsart University, Bangkok, 10900, Thailand

Abstract: Problem statement: Traveling Salesman Problem (TSP) is a famous NP hard problem. Many approaches have been proposed up to date for solving TSP. We consider a TSP tour as a dependent variable and its corresponding distance as an independent variable. If a predictive function can be formulated from arbitrary sample tours, the optimal tour may be predicted from this function. **Approach:** In this study, a combined procedure of the Nearest Neighbor (NN) method, Gaussian Process Regression (GPR) and the iterated local search is proposed to solve a deterministic symmetric TSP with a single salesman. The first tour in the sample is constructed by the nearest neighbor algorithm and it is used to construct other tours by the random 2-exchange swap. These tours and their total distances are training data for a Gaussian process regression model. A GPR solution is further improved with the iterated 2-opt method. In the numerical experiments, our algorithm is tested on many TSP instances and it is compared with the Genetic Algorithm (GA) and the Simulated Annealing (SA) algorithm. **Results:** The proposed method can find good TSP tours within a reasonable computational time for a wide range of TSP test problems. In some cases, it outperforms GA and SA. **Conclusion:** Our proposed algorithm is promising for solving the TSP.

Key words: Gaussian process regression, nearest neighbor, iterated 2-opt algorithm, genetic algorithm, simulated annealing, traveling salesman problem

INTRODUCTION

The Traveling Salesman Problem (TSP) is one of the most well-known NP-hard problems in the field of combinatorial optimization. The objective 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. Although its mathematical formulation is simple, the TSP is difficult because it is combinatorial and the number of solutions increases exponentially with the number of cities.

TSP applications can be found in many fields including job sequencing on a single machine or assignment problems (Gilmore and Gomory, 1964), material handling in a warehouse (Ratliff and Rosenthal, 1983), genome rearrangement (Sankoff and Blanchette, 1997), the drilling of printed circuits boards (Duman and Or, 2004), transportation and logistics problem (Rodriguez and Ruiz, 2012).

Several algorithms are designed to solve the TSP problem. The exact algorithm would be to try all possible permutations (order combinations), but the brute force method takes more computational time than the cutting plane method (Dantzig *et al*., 1954) or the branch and bound method (Little *et al*., 1963). Other heuristics and approximation algorithms include the nearest neighbor algorithm or the so-called greedy algorithm (Bellmore and Nemhauser, 1968), Lin-Kernighan heuristics (Lin and Kernighan, 1973; Helsgaun, 2000) and the *k*-opt heuristic (Helsgaun, 2009).

Moreover, many randomized approaches are shown to perform well on the TSP, e.g., the ant colony optimization (Dorigo and Gambardella, 1997), the tabu search (Gendreau *et al*., 1998), the genetic algorithm (Chatterjee *et al*., 1996; Moon *et al*., 2002), the cross entropy method (Boer *et al*., 2005), the particle swarm optimization (Shi *et al*., 2007) and the simulated annealing algorithm (Geng *et al*., 2011).

The key contributions of this study: We propose an algorithmic approach for solving a deterministic and symmetric TSP with a single salesman. The method integrates the Gaussian Process Regression (GPR) with the Nearest Neighbor algorithm (NN) and improves its solution by using the iterated 2-opt method. The numerical experiments show that our approach can

Corresponding Author: Wanatchapong Kongkaew, Department of Industrial Engineering, Faculty of Engineering, Kasetsart University, Bangkok, 10900, Thailand

yield TSP tours within 1-12% of the optimal solutions for the TSP with the size up to 2103 nodes.

The study is organized as follows: Firstly, the literature survey on GPR applications and a conceptual framework of GPR approach are described. Then, our GPR algorithm for solving TSP is explained and the comparing methods are also described in brief. Next, the experimental results of all algorithms and discussion are presented. Lastly, this work is summarized in conclusion.

MATERIALS AND METHODS

Literature survey on GPR applications: The Gaussian process regression is known as a probabilistic approach for a regression model due to its practical and theoretical simplicity and excellent generalization ability (Rasmussen and Williams, 2006). For applications, GPR is employed in many fields, particularly in machine learning, e.g., to estimate the depth of a point in space from observing its image position from two different cameras (Sinz *et al*., 2004), to learn motion and observation non-parametric system models for sequential state estimation and to apply its algorithm to the problem of tracking an autonomous micro-blimp (Ko *et al*., 2007), to find near optimal sensor placements in the task as an instance of the art-gallery problem (Krause *et al*., 2008). For traffic problems, GPR is applied to predict the traveling time for an arbitrary traffic path of downtown Kyoto in Japan (Ide and Kato, 2009).

The notations in this paper are as follows: Vectors are represented by small Roman symbols, such as x_s and all ones are assumed to be column vectors. The transpose of vectors or matrices is denoted by ^{T}. Matrices are represented by capitalized Roman symbols, such as K, and the (i, j) element of a matrix K is $K_{i,j}$. The identity matrix is I. Finally, the estimated value is denoted with a hat, e.g., $\hat{y}$.

Standard Linear Regression (SLR): The standard linear regression model with Gaussian noise is $y = f(x)+\varepsilon$ and Eq. 1:

$$f(x) = x^T w \quad (1)$$

where x is the input vector and w is a vector of weights (parameters to be estimated) of a linear model (Rasmussen and Williams, 2006). Noise ε follows an independent and identically distributed Gaussian distribution with zero mean and noise variance σ_n^2, that is $\varepsilon \sim N(0,\sigma_n^2)$, where n is the sample size.

Gaussian process: Gaussian Process (GP) is a collection of random variables, any finite number of which has a joint Gaussian distribution and it is specified by its mean function and covariance function (Rasmussen and Williams, 2006). The mean function m(x) and covariance function k(x,x') are defined a real process as $m(x) = E[f(x)] = 0$ and $k(x,x') = E[(f(x) - m(x))(f(x')-m(x'))]$ Thus, we can write the Gaussian process as $f(x) = GP[m(x), k(x,x')]$.

Gaussian process regression: Gaussian process regression is a model to estimate the value of a dependent variable or a response from some observations of dependent variables at certain values of independent variables (Rasmussen and Williams, 2006). A training set T_s of n observations is denoted as $T_s = \{(x_i,y_i)|i =1,..,n\}$, where x_i is an input vector (in our case, a traveling tour) of n_c cities and y_i is a response (the total distance of the tour). The vector inputs of all n observations are aggregated into the $n_c \times n$ matrix X and the total distances are aggregated into the column vector y, so we can write $T_s = (X, y)$. The graphical model of GPR is shown in Fig. 1 (Rasmussen and Williams, 2006).

In the GPR model, the Gaussian basis function $\phi(x_i)$ is specified and it maps a n_c city input vector x_i into N_c-dimensional feature space. Let the matrix $\Phi(X)$ be the aggregation of columns $\phi(x_i)$ for all inputs in training data set. Therefore, the function f(x) in the SLR model (Eq. 1) becomes $\phi(x_i)^T$ w, where w is the vector of weight parameters.

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

$$p(y \mid X, w) = \prod_{i=1}^{n} p(y_i \mid x_i, w) \sim N(X^T w, \sigma_n^2 I),$$

where w is a bias and $w \sim N(0, \sum_p)$. The posterior distribution over the weights based on the Bayesian linear model is computed by Bayes' rule. The form of posterior Gaussian distribution with mean $\bar{w}$ and covariance matrix A^{-1} is given as Eq. 2:

$$p(w \mid X, y) \sim N(\bar{w} = \sigma_n^{-2} A^{-1} X y, A^{-1}) \quad (2)$$

where $A = \sigma_n^{-2} X X^T + \sum_p^{-1}$ (Rasmussen and Williams, 2006).

To make prediction for a test case, all possible parameter values are averaged and weighted by posterior probability. The predictive distribution for the

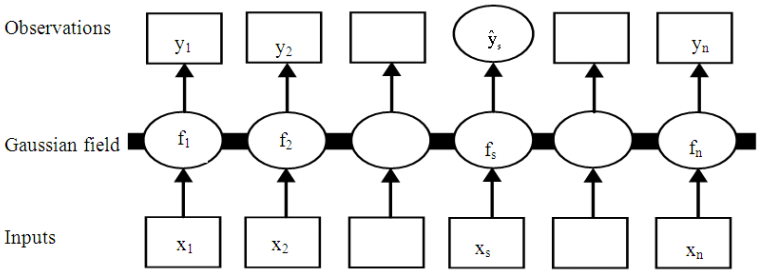

Fig. 1: Graphical model for GPR

function value $f_s \triangleq f(x_s)$ at a single test point x_s, is given by averaging the output of all possible linear models. Thus, the Gaussian posterior is written as Eq. 3:

$$p(f_s \mid x_s, X, y) \sim N\left(\sigma_n^{-2} x_s^T A^{-1} X y, x_s^T A^{-1} x_s\right). \quad (3)$$

The predictive distribution in (3) is also Gaussian distribution, with a posterior mean of the weights from (2) multiplied by the possible value x_s in a test case (Rasmussen and Williams, 2006). Moreover, the predictive variance is a quadratic form of the possible value x_s in a test case multiplied with the posterior covariance matrix. Hence, the predictive distribution becomes Eq. 4:

$$p(f_s \mid x_s, X, y) \sim N\left(\sigma_n^{-2} \phi_s^T A^{-1} \Phi y, \phi_s^T A^{-1} \phi_s\right) \quad (4)$$

With $\Phi = \Phi(X)$, $\phi_s = \phi(x_s)$ and $A = \sigma_n^{-2}\Phi\Phi^T + \Sigma_p^{-1}$. On the right-hand side of (4), the A^{-1} of size n×n is needed for making a prediction and it may be inconvenient if n is large. However, this term can be rewritten as Eq. 5:

$$N\left(\phi_s^T \Sigma_p \Phi\left(K + \sigma_n^2 I\right)^{-1} y, \; \phi_s^T \Sigma_p \phi_s - \phi_s^T \Sigma_p \Phi\left(K + \sigma_n^2 I\right)^{-1} \Phi^T \Sigma_p \phi_s\right) \quad (5)$$

where, the covariance matrix K is $\Phi^T \Sigma_p \Phi$.

Notice that in (5) the entries of matrices, in the form of $\Phi^T \Sigma_p \Phi$, $\phi_s^T \Sigma_p \Phi$, or $\phi_s^T \Sigma_p \phi_s$, are the form $\phi(x_i)^T \Sigma_p \phi(x_s)$ where x_i and x_s are in a training set and a test set, respectively. Let $k(x_i,x_s)$ be $\phi(x_i)^T \Sigma_p \phi(x_s)$ and it is called a covariance function or a kernel for prediction. Similarly, the covariance matrix K is defined as k(X,X) and each of its elements are determined by a covariance function of the pair of inputs in training set, $k(x_i,x_j)$.

The prior on the noisy observations, independent and identically distributed Gaussian noise ε with variance σ_n^2, becomes $K(X,X) + \sigma_n^2 I$. In addition, the joint distribution of the observed response values and the response at the test location is:

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

Thus, the predictive function value is Eq. 6:

$$f_s = K(X,x_s)^T \left(K + \sigma_n^2 I\right)^{-1} y \quad (6)$$

Covariance function: Many choices of covariance functions are available for GP, for example, Matérn class of covariance function, squared exponential covariance function, rational quadratic covariance function and radial basis covariance function (Rasmussen and Williams, 2006). In this study, the Squared Exponential (SE) covariance function is chosen because it is the most widely-used kernel; it is given by Eq. 7:

$$k(x_i,x_j) = \sigma_f^2 \exp\left[\frac{-(x_i - x_j)^2}{2\ell^2}\right] + \sigma_n^2 \delta(x_i,x_j) \quad (7)$$

where $\delta(x_i,x_j)$ is the Kronecker delta function which equals to 1 if and only if i = j and 0 otherwise and ℓ is the characteristic length-scale.

GPR parameter estimation: The hyperparameters of the covariance function (such as the characteristic length-scale ℓ, signal variance of function (σ_f^2) and noise variance σ_n^2) are determined by the maximum likelihood method. The log marginal likelihood function under the GP model is Eq. 8:

$$\log p(y \mid X,\theta) = -\frac{1}{2}y^{T}\alpha - \frac{1}{2}\log\left|K + \sigma_n^2 I\right| - \frac{n}{2}\log 2\pi \quad (8)$$

where $\alpha = K^{-1}y$, θ is a vector of ℓ, σ_f^2 and σ_n^2. The partial derivative of (8) with respect to θ is minimized until converging to zero. This gives a vector of optimized hyperparameters.

Tour construction and representation: The sample TSP tours are needed as an input to GPR. The nearest neighbor algorithm (Nuhoglu, 2007) constructs the first TSP tour which is subsequently used to construct other tours by the 2-exchange method. The nearest neighbor algorithm starts at a chosen starting city and then selects the next closest unvisited city until all cities are included in the tour. The total distance of a tour depends on a chosen starting city (Laporte, 1992). The 2-exchange swapping strategy randomly selects two cities in the tour and swaps them to get a new tour (Larranaga *et al*., 1999).

A tour can be represented in many ways, e.g., path representation, binary string representation, binary matrix representation, edge recombination crossover in binary representation (Larranaga *et al*., 1999). In this work, the binary string representation is selected to encode all tours for GPR input because it is simple and performs well when making test prediction. Each city in a tour is encoded as a string of $[\log_2 n]$ bits and then a complete tour becomes a string of $n[\log_2 n]$ bits (Larranaga *et al*., 1999); that is, a tour $4\rightarrow2\rightarrow1\rightarrow3$ is represented by (011001000010).

Tour improvement: Many algorithms have been proposed to improve a tour; for instance, *r*-opt algorithm, Lin-Kernighan heuristic, simulated annealing algorithm and tabu search (Laporte, 1992). We consider the iterated local search with the 2-exchange neighborhood, or the iterated 2-opt algorithm, because the 2-opt is one of the most famous simple local searches (Johnson and McGeoch, 1997; Lourenco *et al*., 2010). In the 2-opt algorithm, two edges are deleted from a given tour, breaking it into two paths and then those paths are reconnected with two other possible edges. If the new tour is shorter, it becomes the current tour. These steps are repeated until no improvement can be made (i.e., reaching a local optimum). However, the 2-opt algorithm moves with a neighborhood search by starting at the first node and it continues the search process with the next remaining nodes until all nodes are examined (Nuhoglu, 2007). This makes the solution boundary and it can be improved by iterating all search processes, leading to far better solutions (Lourenco *et al*., 2010). In this study, the iterated 2-opt procedure is modified from the 2-opt algorithm (Nuhoglu, 2007) by iterating the whole process until reaching another local optimum which is better.

Proposed GPR Algorithm: Our GPR algorithm for a deterministic TSP with a single salesman is divided into three phases: I, II and III.

Phase I: We prepare the training dataset T_s and the testing dataset.

Input: Distance matrix (D) where D_{ij} is the distance from city *i* to city *j*:

- Construct the first tour by using the nearest neighbor algorithm and other tours from the first tour by using the random 2-exchange swapping strategy
- Encode all tours as binary and aggregate them into X
- Calculate the total distance of each tour and aggregate them into y
- Find a vector of test tour that has the minimum total distance in X and encode it as binary vector of x_s in (3)

Output: Binary matrix of tours (X), vector of observed total distances (y) and binary vector of test tour (x_s)

Phase II: The GPR model is used to predict the length of an optimal tour.

Input: Binary matrix of tours (X), vector of observed total distances (y) and binary vector of test tour (x_s):

- Initialize the hyperparameters ($\ell, \sigma_f^2, \sigma_n^2$)
- Compute the square exponential covariance function of every possible pairs (x_i,x_j) by (7)
- Compute the log marginal likelihood for GPR as specified in (8), and then compute f_s by (6)
- Compute the squared difference between each value in y and value of f_s and identify the *i*th-index of minimum value; that is:

$$i = \arg\min_i \left[\left(y_i - f_s \right)^2 \right].$$

- Find the binary vector of tour (which belong to the *i*th-index in X) to be the binary vector of predictive tour and decode it to the vector of the predictive tour $\hat{x}^*$ (in cities' number).

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

Phase III: This phase implements the iterated 2-opt algorithm to improve $\hat{x}^*$ from Phase II.

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

- Start with $\hat{x}^*$ and its first edge
- Select an edge (a, b) and search for another edge (c, d) and then remove them to break the tour into two paths
- Calculate the sum of distances between these two edges and the sum of distances between edge (a, c) and edge (b, d)
- 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
- Set the obtained tour as an initial tour and repeat the whole process until no improvement (reach a local optimum)
- Keep the best tour and set it as an initial tour for next iteration
- Repeat Steps 2-6 until no improvement can be made. Return the resulting tour and its total distance

Output: Resulting tour $\hat{\hat{x}}^*$ and its total distance ($\hat{\hat{y}}^*$).

Comparing methods:

Classical genetic algorithm (GA): GA is a well-known search heuristic for solving optimization problems using techniques inspired by natural evolution (Chatterjee *et al*., 1996). It involves three basic steps: evaluation, crossover and mutation. In the first step, a population of individual chromosomes is reproduced and good chromosomes (based on their objective function or "fitness" value) are selected for the next generation with some probability. Next, 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 in the new generation completed by the crossover and mutates it at a particular point for a new population. The whole process is iterated until reaching the stopping criteria. In this study, we use GA from a MATLAB toolbox (Kirk, 2007), where 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.

Simulated Annealing Algorithm (SA): SA is a probabilistic method based on the process of material annealing in metallurgy (Laporte, 1992). For the TSP, SA starts from a given initial tour (when temperature is high) and a schedule for gradually decreasing temperature. It generates a new tour by randomly swapping two cities in a current tour and calculates the difference in the length of tour between the current tour and the new tour as ΔE. If the new tour is better than the current tour, it is accepted as current tour; otherwise, the new tour is accepted with a probability given by:

$$\exp\left(-\frac{\Delta E}{T}\right)$$

where, T is the current temperature, which is decreased by a cooling rate in each iteration. These steps are repeated until reaching the stopping criteria. In this paper, we use SA from a MATLAB toolbox (Seshadri, 2006), where the main parameters are the initial temperature (T_{int}), end temperature (T_{end}), cooling rate (T_{cool}) and the stopping criteria on the number of iteration.

Numerical experiments: Our GPR algorithm is modified from the GPML toolbox (Rasmussen and Nickisch, 2010) which is implemented in MATLAB. We experiment all algorithms with 60 TSP test problems, with the number of cities ranging from 16 to 2103 from the TSP library (Reinelt, 1995). The computation is done on PC running Intel(R) Pentium Dual CPU 2 GHz. processor with 1 GB of memory.

For the GPR algorithm, we use 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 algorithm needs a training data set which we create randomly, we repeat the GPR algorithm on each TSP instance for nine times. Then total distances are averaged. Our GPR algorithm integrates GPR with the Nearest Neighbor algorithm (NN) and the iterated 2-opt method, namely NN+GPR+Iterated 2-opt.

The GA is set with the following parameters: *pop_size* = 100, $p_c = 0.5$, $p_m = 0.8$ (Kirk, 2007) and the number of iteration = 10000 (by trial and error). SA parameters are: T_{int}= 1000 and T_{end} = 0.0025 (Geng *et al*., 2011), T_{cool} = 0.97 and the number of iteration = 20000 (Seshadri, 2006).

The performance of all algorithms is the deviation between the total distance of the resulting tour ($\hat{\hat{y}}^*$) and that of the actual optimal solution (y^*) for each instance, given by:

$$\text{deviation } (\%) = \frac{\hat{\hat{y}}^* - y^*}{y^*} \times 100,$$

and the computational time is also considered.

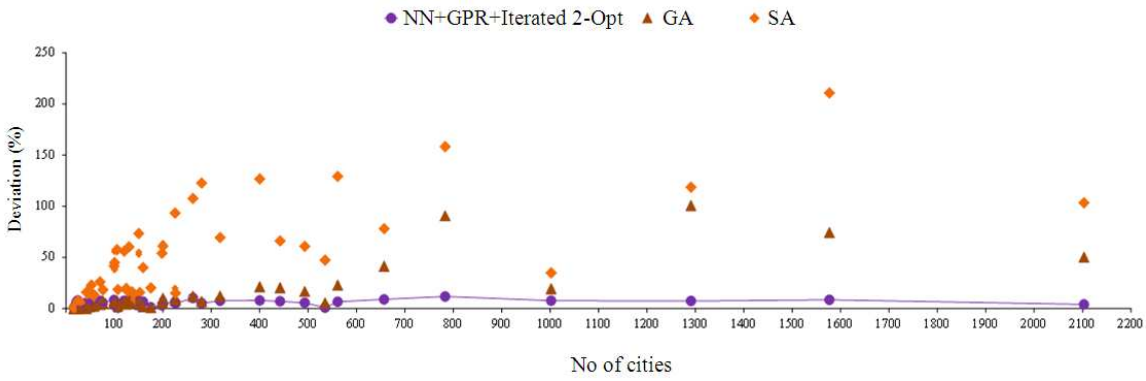

Fig. 2: Comparison of the deviation from optimal solutions among three algorithms

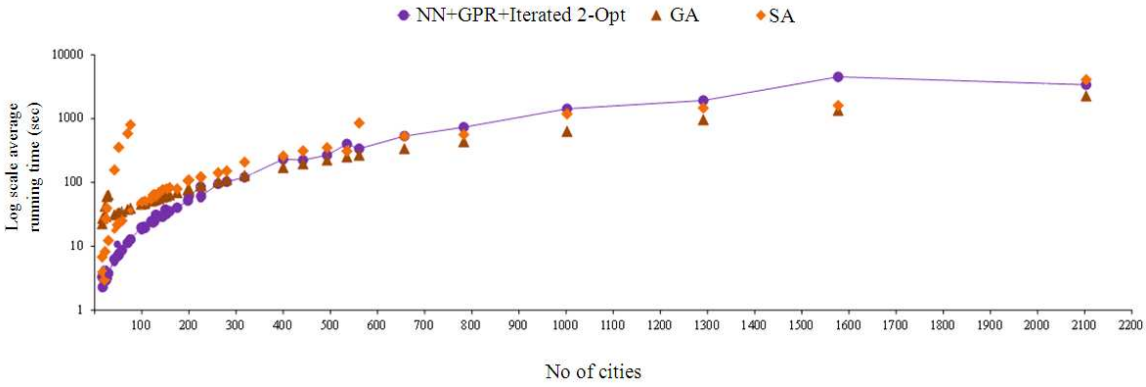

Fig. 3: Comparison of the log scale of average running time among three algorithms

RESULTS

The results of the proposed algorithm for each instance, including the best solution obtained from nine trials, the average performance of nine trials, the 95% confidence interval, the percentage of the deviation of its solution from the optimal solution, and the average running time, are shown in Table 1. The percentages of the deviation are less than 12% and the average running time is between 2 to 4, 520 sec. For the gr17 problem, the GPR algorithm can find an optimal solution within 3 sec. In addition, for ulysses16, ulysses22, bays29, swiss42, pr107 and si535 problems, the 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%.

The summary results of the proposed algorithm and the comparing algorithms for each instance, including the percentage of the deviation of their solutions from the optimal solutions and the average running time, are shown in Table 2. The comparison plots of the deviation of all algorithms' solutions from optimal solutions and the log scale of average running time among three algorithms are provided in Fig. 2 and 3, respectively. Comparing with GA and SA, the proposed algorithm can find better solutions for 39 TSP instances (out of 60). The average running time of 49 TSP instances are also less than one second. Although the proposed algorithm spends more run time than both comparing algorithms in the remaining 11 TSP instances, the solutions obtained are better.

When the size of the test problem is bigger, our proposed approach performs well while the performance of GA and SA deteriorates, as shown in Fig. 2. Figure 3 shows the average running time consumed by the NN+GPR+Iterated 2-opt algorithm spends more run time than GA and SA when the problem size is larger than 535 cities, but the solution's quality is better.

Table 1: Performance of our GPR algorithm

Test problem	# of nodes	Optimal solution (units)	Best solution out of 9 trials (units)	Average $\hat{y}^*$ (units)	95% Confidence interval Lower (units)	Upper (units)	Deviation (%)	Average running time (sec.)
Ulysses16	16	6859	6909	6909.0	-	-	0.7	3.3
gr17	17	2085	2085	2085.0	-	-	0.0	2.3
gr21	21	2707	2816	2870.2	2854.6	2885.9	6.0	2.9
ulysses22	22	7013	7013	7032.3	7002.8	7061.8	0.3	4.2
gr24	24	1272	1272	1371.6	1342.9	1400.3	7.8	2.9
fri26	26	937	961	961.0	-	-	2.6	3.2
bayg29	29	1610	1660	1666.2	1663.5	1668.9	3.5	3.8
bays29	29	2020	2035	2035.0	-	-	0.7	3.6
dantzig42	42	699	736	739.0	736.8	741.2	5.7	6.3
swiss42	42	1273	1274	1283.9	1267.0	1300.8	0.9	5.5
att48	48	10628	10954	11142.2	11060.2	11224.2	4.8	6.9
gr48	48	5046	5286	5359.2	5324.2	5394.2	6.2	10.8
hk48	48	11461	12003	12003.0	-	-	4.7	6.9
eil51	51	426	428	432.4	429.2	435.7	1.5	7.3
berlin52	52	7542	7596	7666.3	7504.1	7828.5	1.6	7.7
brazil58	58	25395	25699	25947.0	25875.5	26018.5	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	108159	112220	113981.2	112450.7	115511.7	5.4	13.4
rat99	99	1211	1278	1311.0	1288.9	1333.1	8.3	19.7
kroA100	100	21282	21768	21956.8	21834.1	22079.4	3.2	19.2
kroB100	100	22141	22755	23133.0	22882.8	23383.2	4.5	17.4
kroC100	100	20749	22005	22013.9	22011.3	22016.5	6.1	18.3
kroD100	100	21294	22857	23268.6	23086.2	23450.9	9.3	18.3
kroE100	100	22068	22596	22729.0	22531.1	22926.9	3.0	18.7
rd100	100	7910	8311	8596.8	8388.4	8805.2	8.7	18.9
lin105	105	14379	14984	15070.7	15008.8	15132.5	4.8	20.2
pr107	107	44303	44573	44666.9	44553.6	44780.1	0.8	19.4
gr120	120	6942	7305	7469.8	7372.9	7566.7	7.6	24.7
pr124	124	59030	61746	61746.0	-	-	4.6	23.7
bier127	127	118282	121772	124480.8	122390.8	126570.7	5.2	24.8
ch130	130	6110	6513	6630.2	6544.7	6715.7	8.5	31.2
pr136	136	96772	104093	105196.3	104454.0	105938.7	8.7	29.6
pr144	144	58537	60754	60754.0	-	-	3.8	29.1
ch150	150	6528	6759	6772.3	6764.6	6780.0	3.7	37.7
kroA150	150	26524	28372	28860.2	28695.6	29024.8	8.8	38.5
kroB150	150	26130	27484	27688.8	27622.2	27755.3	6.0	37.3
pr152	152	73682	75774	75970.3	75810.2	76130.5	3.1	32.1
u159	159	42080	44380	44899.7	44667.0	45132.4	6.7	35.5
si175	175	21407	21656	21671.0	21665.2	21676.8	1.2	40.2
d198	198	15780	16042	16271.1	16082.4	16459.9	3.1	52.1
kroA200	200	29368	30457	30689.1	30559.9	30818.3	4.5	61.3
kroB200	200	29437	31731	31954.6	31879.9	32029.2	8.6	57.3
tsp225	225	3919	4137	4176.1	4158.7	4193.6	6.6	85.5
ts225	225	126643	130742	132839.0	132197.2	133480.8	4.9	55.1
pr226	226	80369	83184	84415.9	83937.8	84894.0	5.0	61.1
gil262	262	2378	2580	2623.0	2606.0	2640.0	10.3	95.2
a280	280	2579	2668	2713.6	2689.4	2737.7	5.2	102.8
lin318	318	42029	44229	45241.8	44787.8	45695.8	7.6	120.6
rd400	400	15281	16354	16491.1	16385.8	16596.4	7.9	232.1
pcb442	442	50778	53799	54363.2	54100.8	54625.7	7.1	223.9
d493	493	35002	36553	36971.9	36706.1	37237.7	5.6	268.0
si535	535	48450	48771	48865.3	48827.1	48903.5	0.9	400.6
pa561	561	2763	2923	2946.8	2933.6	2959.9	6.7	338.2
d657	657	48912	52916	53281.7	53125.2	53438.1	8.9	533.0
rat783	783	8806	9779	9846.7	9809.8	9883.6	11.8	734.3
pr1002	1002	259045	276226	279077.1	277476.5	280677.8	7.7	1426.0
d1291	1291	50801	53924	54545.3	54066.6	55024.0	7.4	1924.3
fl1577	1577	[22204,22249]	23861	24099.7	23974.0	24225.3	8.5	4519.2
d2103	2103	[79952,80529]	82822	83161.7	83017.1	83306.2	4.0	3396.6

Table 2: Comparison of the search performance

		Deviation (%)			Average running time (sec.)		
Test problem	# of nodes	NN+GPR+ Iterated 2-opt	GA	SA	NN+GPR+ Iterated 2-opt	GA	SA
ulysses16	16	0.7	0.0	0.8	3.3	22.5	6.8
gr17	17	0.0	0.1	1.4	2.3	27.5	3.9
gr21	21	6.0	0.4	2.8	2.9	29.4	3.0
ulysses22	22	0.3	0.4	2.3	4.2	42.2	8.3
gr24	24	7.8	0.4	7.4	2.9	29.2	27.1
fri26	26	2.6	0.0	6.7	3.2	60.3	39.7
bayg29	29	3.5	0.8	5.9	3.8	63.6	12.3
bays29	29	0.7	0.4	5.5	3.6	63.6	12.2
dantzig42	42	5.7	0.0	15.9	6.3	31.1	157.4
swiss42	42	0.9	2.5	15.1	5.5	31.2	17.8
att48	48	4.8	1.7	16.8	6.9	32.4	21.7
gr48	48	6.2	0.8	21.6	10.8	32.8	20.8
hk48	48	4.7	2.4	17.9	6.9	32.4	21.9
eil51	51	1.5	3.8	10.0	7.3	33.4	356.0
berlin52	52	1.6	3.5	22.9	7.7	33.3	23.3
brazil58	58	2.2	2.3	13.4	8.7	35.0	25.2
st70	70	7.4	4.1	26.1	11.3	37.8	585.0
eil76	76	5.5	5.0	18.5	12.8	39.4	803.1
pr76	76	5.4	9.0	16.4	13.4	39.2	36.5
rat99	99	8.3	7.1	41.6	19.7	45.8	45.2
kroA100	100	3.2	6.1	44.9	19.2	46.0	48.7
kroB100	100	4.5	6.4	37.9	17.4	45.5	48.2
kroC100	100	6.1	7.5	45.0	18.3	45.8	48.4
kroD100	100	9.3	5.2	42.1	18.3	45.8	49.6
kroE100	100	3.0	4.7	38.8	18.7	45.7	48.4
rd100	100	8.7	7.4	55.8	18.9	45.6	47.3
lin105	105	4.8	7.5	57.6	20.2	47.2	50.0
pr107	107	0.8	2.2	18.7	19.4	47.5	50.4
gr120	120	7.6	6.6	56.1	24.7	51.3	55.8
pr124	124	4.6	4.3	19.8	23.7	52.5	63.6
bier127	127	5.2	6.2	17.0	24.8	53.4	64.5
ch130	130	8.5	9.1	60.3	31.2	54.7	61.4
pr136	136	8.7	8.4	16.3	29.6	56.3	69.4
pr144	144	3.8	5.3	8.9	29.1	58.8	77.1
ch150	150	3.7	9.6	73.3	37.7	61.1	76.5
kroA150	150	8.8	8.7	53.0	38.5	60.8	80.0
kroB150	150	6.0	7.6	55.3	37.3	60.8	79.9
pr152	152	3.1	5.3	15.7	32.1	61.3	79.2
u159	159	6.7	1.9	40.0	35.5	63.8	82.9
si175	175	1.2	1.0	20.2	40.2	69.8	79.7
d198	198	3.1	5.3	54.1	52.1	78.0	106.8
kroA200	200	4.5	10.0	61.1	61.3	78.8	109.5
kroB200	200	8.6	10.4	63.7	57.3	79.0	109.6
tsp225	225	6.6	8.0	93.3	85.5	88.7	122.4
ts225	225	4.9	6.8	19.8	55.1	88.8	129.8
pr226	226	5.0	9.2	15.1	61.1	89.2	120.8
gil262	262	10.3	12.0	107.6	95.2	103.8	141.6
a280	280	5.2	6.8	122.7	102.8	111.2	152.0
lin318	318	7.6	12.1	69.4	120.6	129.0	208.9
rd400	400	7.9	21.3	126.8	232.1	172.1	259.1
pcb442	442	7.1	20.4	66.1	223.9	195.9	312.9
d493	493	5.6	16.7	60.9	268.0	224.4	351.2
si535	535	0.9	5.5	47.4	400.6	252.0	309.6
pa561	561	6.7	22.8	129.2	338.2	268.4	855.2
d657	657	8.9	41.4	78.1	533.0	337.6	527.7
rat783	783	11.8	90.6	158.3	734.3	437.6	561.3
pr1002	1002	7.7	19.4	34.9	1426.0	632.9	1185.2
d1291	1291	7.4	100.6	118.6	1924.3	962.6	1469.1
fl1577	1577	8.5	74.3	210.8	4519.2	1343.6	1605.7
d2103	2103	4.0	50.2	103.4	3396.6	2263.7	4083.5

DISCUSSION

We consider the nearest neighbor method, Gaussian process regression and the iterated 2-opt method. It adopts the NN method to construct the first sample tour and uses it to construct other sample tours by the 2-exchange method, then they are treated as an input for the GPR for predicting the optimal tour. In addition, the solutions from this approach are not very good, far away from the optimal solutions. Thus, the improvement procedure is called for. The iterated 2-opt method is a local search and the combined approach is called "NN+GPR+Iterated 2-opt." The numerical experiments show that it performs well on a set of 60 TSP instances.

Moreover, we compare the proposed method with two well-known methods, i.e., Genetic Algorithm (GA) and Simulated Annealing Algorithm (SA). The experimental results show the NN+GPR+Iterated 2-opt algorithm yields better overall solution quality than GA and SA even though there are some TSP instances in which it is not the winner, comparing with GA and SA. Our algorithm also consumes less overall run time than GA and SA. Although there are some TSP instances that it spends more run time than both algorithms, it acquires the better solution quality. Thus, in this study, the NN+GPR+Iterated 2-opt algorithm is the best method, comparing among three approaches.

CONCLUSION

In this study, we propose an algorithm based on Gaussian Process Regression (GPR) for predicting the optimal tour of the deterministic Traveling Salesman Problem (TSP) with a single salesman. This algorithm formulates TSP as a GPR model where the response is the length of traveling tour while the predictor is the traveling tours with the cities' number. The NN+GPR embedded with the iterated 2-opt algorithm achieves a reasonable trade-off between computational time and solution quality. The results indicate that NN+GPR+Iterated 2-opt performs well on a set of 60 TSP instances. However, it consumes more running time than the two comparing algorithms (genetic algorithm and simulated annealing algorithm) for some TSP instances.

ACKNOWLEDGMENT

The researchers would like to thank Associate Professor Dr. Peerayuth Charnsethikul, Department of Industrial Engineering at Kasetsart University, for his constructive comments and suggestions. Also, the first author gratefully acknowledges the financial support from Prince of Songkla University for this study.

REFERENCES

Bellmore, M. and G.L. Nemhauser, 1968. The traveling salesman problem: A survey. Operat. Res., 16: 538-558. DOI: 10.1287/opre.16.3.538

Boer, P.D.T., D.P. Kroese, S. Mannor and R.Y. Rubinstein, 2005. A tutorial on the cross-entropy method. Ann. Operat. Res., 134: 19-67. DOI: 10.1007/s10479-005-5724-z

Chatterjee, S., C. Carrera and L.A. Lynch, 1996. Genetic algorithms and traveling salesman problems. Eur. J. Operat. Res., 93: 490-510. DOI: 10.1016/0377-2217(95)00077-1

Dantzig, G., R. Fulkerson and S. Johnson, 1954. Solution of a large-scale traveling-salesman problem. Operat. Res., 2: 393-410. DOI: 10.1287/opre.2.4.393

Dorigo, M. and L.M. Gambardella, 1997. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Trans. Evolut. Comput., 1: 53-66. DOI: 10.1109/4235.585892

Duman, E. and I. Or, 2004. Precedence constrained TSP arising in printed circuit board assembly. Int. J. Prod. Res., 42: 67-78. DOI: 10.1080/00207540310001601073

Gendreau, M., G. Laporte and F. Semet, 1998. A tabu search heuristic for the undirected selective travelling salesman problem. Eur. J. Operat. Res., 106: 539-545. DOI: 10.1016/S0377-2217(97)00289-0

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 Comput., 11: 3680-3689. DOI: 10.1016/j.asoc.2011.01.039

Gilmore, P.C. and R.E. Gomory, 1964. Sequencing a one state-variable machine: A solvable case of the traveling salesman problem. Operat. Res., 12: 655-679. DOI: 10.1287/opre.12.5.655

Helsgaun, K., 2000. An effective implementation of the Lin-Kernighan traveling salesman heuristic. Eur. J. Operat. Res., 126: 106-130. DOI: 10.1016/S0377-2217(99)00284-2

Helsgaun, K., 2009. General k-opt submoves for the Lin–Kernighan TSP heuristic. Math. Prog. Comput., 1: 119-163. DOI: 10.1007/s12532-009-0004-6

Ide, T. and S. Kato, 2009. Travel-time prediction using Gaussian process regression: A trajectory-based approach. Proceedings of the 9th SIAM International Conference on Data Mining, Apr. 30-May. 2, SIAM, Nevada, USA, pp. 1185-1196.

Johnson, D.S. and L.A. McGeoch, 1997. The Travelling Salesman Problem: A Case Study in Local Optimization. In: Local Search in Combinatorial Optimization, Aarts, E.H.L. and J.K. Lenstra, (Eds.). John Wiley and Sons, Chichester, UK., ISBN: 9780471948223, pp: 215-310.

Kirk, J., 2007. Traveling salesman problem-genetic algorithm.

Ko, J., D.J. Klein, D. Fox and D. Haehnel, 2007. Gaussian processes and reinforcement learning for identification and control of an autonomous blimp. Proceedings of the EEE Conference on Robotics and Automation, Apr. 10-14, IEEE Xplore Press, Roma, pp: 742-747. DOI: 10.1109/ROBOT.2007.363075

Krause, A., A. Singh and C. Guestrin, 2008. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. J. Mach. Learn. Res., 9: 235-284.

Laporte, G., 1992. The traveling salesman problem: an overview of exact and approximate algorithms. Eur. J. Operat. Res., 59: 231-247. DOI: 10.1016/0377-2217(92)90138-Y

Larranaga, 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. Artif. Intell. Rev., 13: 129-170. DOI: 10.1023/A:1006529012972

Lin, S. and B.W. Kernighan, 1973. An effective heuristic algorithm for the traveling-salesman problem. Operat. Res., 21: 498-516. DOI: 10.1287/opre.21.2.498

Little, J.D.C., K.G. Murty, D.W. Sweeney and C. Karel, 1963. An algorithm for the traveling salesman problem. Operat. Res., 11: 972-989. DOI: 10.1287/opre.11.6.972

Lourenco, H.R., O.C. Martin and T. Stutzle, 2010. Iterated Local Search: Framework and Applications. In: Handbook of Metaheuristics, Gendreau, M. and J.Y. Potvin, (Eds.). Springer, New York, ISBN: 9781441916631, pp: 363-397.

Moon, C., J. Kim, G. Choi and Y. Seo, 2002. An efficient genetic algorithm for the traveling salesman problem with precedence constraints. Eur. J. Operat. Res., 140: 606-617. DOI: 10.1016/S0377-2217(01)00227-2

Nuhoglu, M., 2007. Shortest path heuristics (nearest neighborhood, 2 opt, farthest and arbitrary insertion) for travelling salesman problem.

Rasmussen, C.E. and C.K.I. Williams, 2006. Gaussian Processes for Machine Learning. 1st Edn., MIT Press, Cambridge, ISBN-10: 026218253X, pp: 266.

Rasmussen, C.E. and H. Nickisch, 2010. Gaussian processes for machine learning (GPML) toolbox. J. Mach. Learn. Res., 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. Operat. Res., 31: 507-521. DOI: 10.1287/opre.31.3.507

Reinelt, G., 1995. The TSPLIB symmetric traveling salesman problem instances.

Rodriguez, A. and R. Ruiz, 2012. The effect of the asymmetry of road transportation networks on the traveling salesman problem. Comput. Operat. Res., 39: 1566-1576. DOI: 10.1016/j.cor.2011.09.005

Sankoff, D. and M. Blanchette, 1997. The median problem for breakpoints in comparative genomics. Proceedings of the 3rd Annual Conference on Computing and Combinatorics, (CC' 97), Springer, London, UK, pp: 251-264.

Seshadri, A., 2006. Traveling Salesman Problem (TSP) using simulated annealing.

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. Inform. Process. Lett., 103: 169-176. DOI: 10.1016/j.ipl.2007.03.010

Sinz, F.H., J.Q. Candela, G.H. Bakir, C.E. Rasmussen and M.O. Franz, 2004. Learning depth from stereo. Pattern Recognition, 3175: 245-252. DOI: 10.1007/978-3-540-28649-3_30