THESIS APPROVAL

GRADUATE SCHOOL, KASETSART UNIVERSITY

Master of Engineering (Industrial Engineering)

DEGREE

Industrial Engineering | Industrial Engineering

FIELD | **DEPARTMENT**

TITLE: Solving the Traveling Salesman Problem with Gaussian Process Regression

NAME: Miss Jarumas Chantapanich

THIS THESIS HAS BEEN ACCEPTED BY

_____ **THESIS ADVISOR**

(Assistant Professor Juta Pichitlamken, Ph.D.)

_____ **THESIS CO-ADVISOR**

(Ms. Suwitchaporn Witchakul, D.Eng.)

_____ **DEPARTMENT HEAD**

(Associate Professor Kongkiti Phusavat, Ph.D.)

APPROVED BY THE GRADUATE SCHOOL ON _____

_____ **DEAN**

(Associate Professor Gunjana Theeragool, D.Agr.)

THESIS

SOLVING THE TRAVELING SALESMAN PROBLEM WITH GAUSSIAN PROCESS REGRESSION

JARUMAS CHANTAPANICH

A Thesis Submitted in Partial Fulfillment of
the Requirements for the Degree of
Master of Engineering (Industrial Engineering)
Graduate School, Kasetsart University
2011

Jarumas Chantapanich 2011: Solving the Traveling Salesman Problem with Gaussian Process Regression. Master of Engineering (Industrial Engineering), Major Field: Industrial Engineering , Department of Industrial Engineering. Thesis Advisor: Assistant Professor Juta Pichitlamken, Ph.D. 102 pages.

The traveling salesman problem (TSP) is a generalized form of the simple problem to find the smallest closed loop or distance from route that connects a number of points in a plane. We present a new heuristics method for solving TSP which is NP-hard. Given a small set of data, we first fit a Gaussian process regression function and then find a route that minimizes this regression function. The route is further transformed into a TSP tour. The numerical experiment shows that our approach can find a reasonably good solution. This method can predict an optimal solution which is higher than the optimal value by 1.4-13% when being experimented on test problems from TSPLIB (Bixby and Reinelt 1995). We expect our heuristics to improve if we use a more effective method for a tour construction.

_____ _____ ____/ ____/___

Student's signature Thesis Advisor's signature

ACKNOWLEDGEMENTS

This research project would not have been possible without the support of many people. The author wishes to express her gratitude to her supervisor, Assistant Professor Juta Pichitlamken, Ph.D. who was abundantly helpful and offered invaluable assistance, support and guidance. Deepest gratitude is also due to the members of the supervisory committee, professor Suvichaporn Nichakul, Ph.D without their knowledge and assistance this study would not have been successful.

Special thanks also to all professors, especially the International Industrial Engineering Program professors for sharing the literature and invaluable assistance. Not forgetting to the professor's assistants who support everything.

The author would also like to truly thank to the Faculty for providing the financial means and laboratory facilities. The author wishes to express her love and gratitude to her beloved families; for their understanding & endless love, through the duration of her studies.

Jarumas Chantapanich
September 2011

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure **Page**

Appendix Figure

SOLVING THE TRAVELING SALESMAN PROBLEM WITH GAUSSIAN PROCESS REGRESSION

INTRODUCTION

The traveling salesman problem (TSP) is widely studied by mathematicians and operation researchers because it is commonly found in real-world problems, such as finding a minimum distance in logistics problems (Dorigo and Gambardell, 1996) and optimizing a production sequence for scheduling problems (Jeong, 1997). The problem can simply be stated as: a traveling salesman wishes to visit exactly once each of a list of n cities (where the cost of traveling from city i to city j is C_{ij}) and then return to the home city. The objective of TSP is to minimize the total cost of traveling, $\sum_{i,j=1}^{n} C_{ij}$ (Hoffman and Padberg, 1985).

TSP is one of combinatorial optimization problems. TSP is NP-complete. Thus, the running times for any heuristic algorithms to solve TSP increases exponentially with the number of cities (Hall, 1995). Although the problem is difficult, a large number of heuristics perform well; some instances with many thousands of cities can be solved. Applegate *et al.* (2006) solve a traveling salesman problem which models the production of printed circuit boards having 7,397 holes. Later, they solve another problem with over the 13,509 largest cities in the U.S.

We employ some random tours “independent variables” and total costs or total distances “dependent variables” to generate a total cost regression function. Therefore, TSP can be viewed as a regression problem. The relationship between dependent variables and independent variables are likely to be nonlinear; thus, a

multiple linear regression cannot be applied. Gaussian process regression (GPR) is capable of fitting arbitrary-shaped functions, so it is selected to fit a response function for TSP

GPR provides a powerful methodology for modeling data that exhibit complex characteristics such as nonlinear behaviors while retaining mathematical simplicity. Gaussian process is a collection of random variables, any finite number of which has (consistent) Gaussian distribution. An example of Gaussian process applications is in prediction control (Kocijan *et al.*, 2003).

OBJECTIVES

The objective of this study is to apply GPR to TSP to predict the minimum cost or distance and to apply a numerical method to estimate a corresponding TSP tour of this prediction.

Assumptions

This study will be considered under the following delimitation below:

We consider a geographical TSP. A geographical TSP has their coordinates as latitudes and longitudes of the Earth.

The distance or cost matrix are symmetric.

The GPR hyperparameters are calculated based on all of the generated data.

GPR prediction model based on a square exponential covariance function with automatic relevance determination (ARD) and independent noise.

Test problem is 96 Africa, Ulysses22, Ulysses16, and Burma14 from TSPLIB (Bixby and Reinelt 1995)

Significance of study

The benefit of this study is to speed up the calculation of the TSP tour and to provide a good (close to optimal) TSP tour within a small amount of time.

LITERATURE REVIEW

Development of TSP and related works on GPR are described as follows: Sections 1 and 3 describe the related TSP and GPR literature. We describe GPR and the Sparse Multiscale GPR theory in Sections 2 and 4. The squared Exponential Kernel and hyperparameter adaptation by using automatic relevance determination which is used in applying GPR to TSP are described in Sections 5 and 7. Spare Multiscale Gaussian Process Regression is our proposed method (Section 6).

1. Traveling Salesman Problems

The classical TSP is symmetric, i.e., distance from node i to node j is equal to distance from node j to node i. Lui *et al.* (2007) propose a heuristic for this type of TSP. Their method is to split a TSP tour into overlapped blocks and then improve each block separately. By doing a local search using the Generalized Crossing method, each block is explored intensively in order to improve the existing solution. When comparing with an adaptive neural network method (Cochrane, and Beasley, 2003), this algorithm obtains a better solution.

The constraints of TSP are not only to visit all the cities exactly once but sometimes TSP also has other conditions on distance or cost such as the Orienteering and Discounted-Reward TSP, where both are NP-hard (Blum *et al.*, 2007). The goal of the Orienteering TSP is to find the path with maximum reward collected, subject to a hard limit on total distance. While in the Discounted-Reward TSP, the length limit is given a discount factor in order to maximize total discount reward collected.

TSP is applied to many real-world situations. One of the common problems is when cities can be dynamically added or removed. Varga *et al.* (2009) propose a multi-agent approach, based on the sensitive stigmergic agent system model (Grasse 1959), refined with new types of messages between agents. The agent sends messages every time change occurs; for instance, when an agent observes that the city

has disappeared or appeared. After testing under various pheromone sensitivity levels and learning abilities for agents, the proposed model appear to have good performance.

Hasegawa (2006) shows that TSP can be applied to complex physical problems. The temperature cycling experiments is a local search process that has a resemblance to polymer glass dynamics at the point that a memory effect and a relaxation acceleration appear in the case of negative and positive cycling. The temperature cycling experiments is formulated as a random Euclidean TSP and is solved with the Metropolis algorithm (Metropolis *et al.,* 1953).

2. Gaussian Process

The material in this section is taken from Kalaitzis (2009), Shah (2009), and Rasmussen and Williams (2006).

Formally, a Gaussian process (GP) is a stochastic process over a feature space (an abstract space where each pattern sample is represented as a point in n-dimensional space. Its dimension is determined by the number of features used to describe the patterns). The probability distribution $p(f(x_1), f(x_2),\ldots, f(x_n))$ of a function $f(x)$ for any finite set of points $\{x_1, x_2,\ldots, x_n\}$ mapped to that space is Gaussian, and such that any of these Gaussian distributions is Kolmogorov consistent (Kalaitzis, 2009).

Kolmogorov consistency is satisfied when $K_{ij} = k(x_i, x_j)$ for some covariance function k such that all possible k are positive semi-definite (i.e., $y^T K_y \geq 0$). Exchangeability is satisfied when the random variable data are independent and identically distributed. It means that the order in which they become available has no

impact on the marginal distribution; hence there is no need to fix ordering of a data from the training set for validation purposes.

A GP can be specified by giving the second order characteristics: mean function and covariance function. Let us define the mean function as $\mu(x)$ and the covariance function as $k(x, x')$, as follows:

$$\mu(x) = E[f(x)]; \tag{1}$$

$$k(x, x') = E[(f(x) - \mu(x))(f(x') - \mu(x'))]. \tag{2}$$

A GP is thus a generalization of the Gaussian probability distribution. It is specified by a mean function $\mu(x)$ and covariance function $k(x, x')$ as follows:

$$f(x) \sim GP(\mu(x), k(x, x')). \tag{3}$$

A GP automatically implies the consistency property which simply means that if the GP specifies normal distribution of dependent variable by mean and covariance as $(f(x_1), f(x_2)) \sim N(\mu, \Sigma)$ then it has already specified $(f(x_1)) \sim N(\mu_1, \Sigma_{11})$ where Σ_{11} is a relevant sub-matrix of Σ. Examination of a larger set of variables does not change the distribution of the smaller set. Let us consider a simple Bayesian linear regression model $f(x) = \phi(x)^T \omega$ with prior $\omega \sim N(0, \Sigma_p)$. Thus, we have the mean and covariance to be

$$E[f(x)] = \phi(x)^T E[\omega] = 0 \tag{4}$$

$$E[f(x)f(x')] = \phi(x)^T E[\omega\omega^T]\phi(x') = \phi(x)^T \Sigma_p \phi(x'). \quad (5)$$

Thus, $f(x)$ and $f(x')$ are jointly Gaussian distributed with mean and covariance as given in the Equations (4) and (5). The choice of different covariance functions allows us to take into consideration different aspects of $f(x)$. In our case, the choice is the squared exponential covariance function, given below:

$$k(x,x') = \sigma_f^2 \exp\left(\frac{-(x-x')^2}{2l^2}\right) + \sigma_n^2 \delta(x,x'), \quad (6)$$

where σ_f^2 gives us the maximum allowable covariance of $f(x)$. When $x = x'$ then $k(x,x')$ approaches its maximum where the Gaussian process appears to be a smooth function as its neighbors are alike. The parameter l affects the length of the dependence. If x is far away from x', then $k(x,x') \approx 0$. Parameter σ_n^2 helps to decide the covariance of the noise, and $\delta(x,x')$ is the Kronecker delta function, where $\delta(x,x')=1$ if $x = x'$ and 0 otherwise.

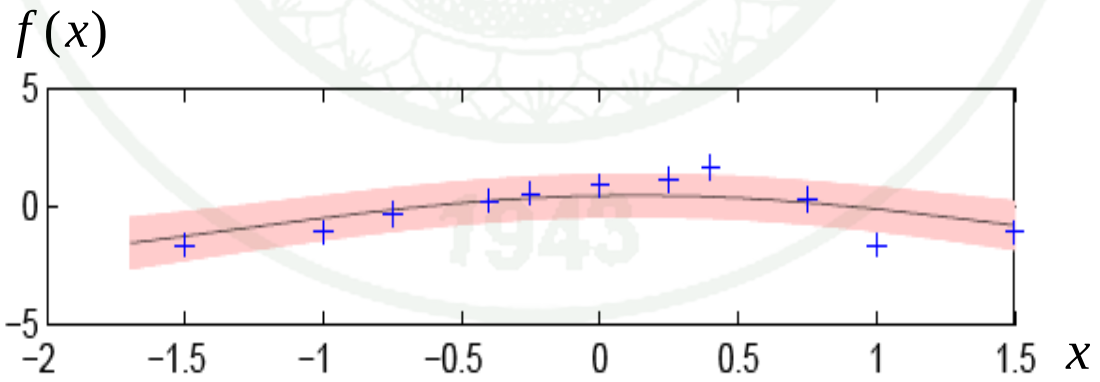


Figure 1 Effect of a hyperparameter l (l=2)**.**

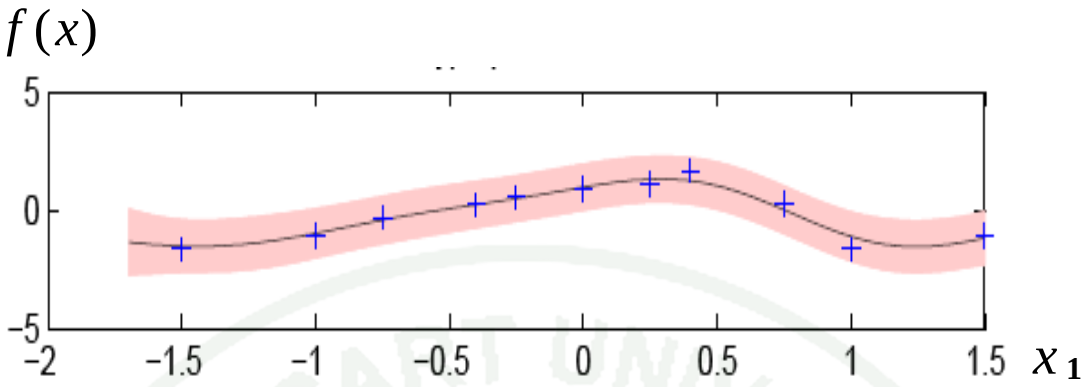


Figure 2 Effect of a hyperparameter l (l=0.5**).**

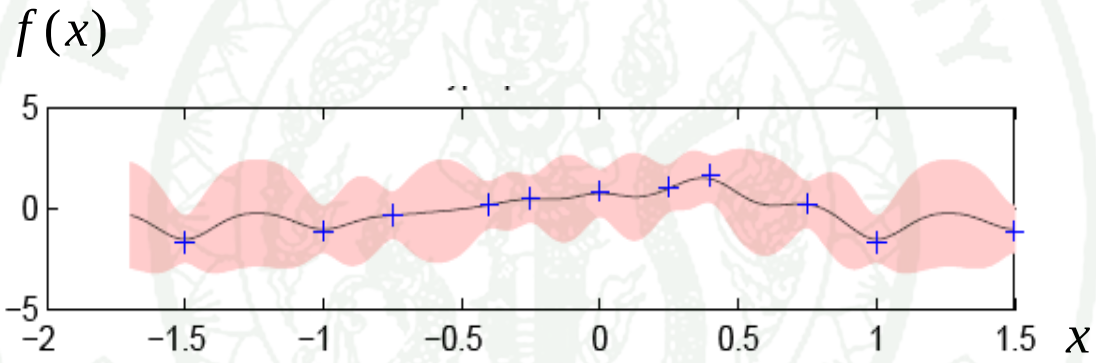


Figure 3 Effect of a hyperparameter l (l=0.1**).**

In Figures 1-3, sample data is generated from a GP with hyper-parameters (l, σ_f, σ_n) = (2, 1.27, 0.3). Using Gaussian process prediction, we obtain a 95% confidence region for the underlying function. Figures 2 and 3 show the Gaussian process predictions on the same data set using different hyper-parameters (0.5, 1.27, 0.3) and (0.1, 1.27, 0.3) respectively.

In Figure 3, we notice that the error variance is larger for the input values that are distant from the training data. When we have the length scale very large such as in Figure 1, the regressed mean does not pass near any training point. Thus, there is a

need of closely studying the hyper-parameters in order to get the right regression curve. It can be shown that the squared exponential covariance function corresponds to a Bayesian linear regression model with an infinite number of basis functions (Shah 2009). We can also obtain the covariance function from a linear combination of an infinite number of Gaussian- shaped basis functions.

Given n observations y which is scalar and $y = f(x)$ at a test points x, our objective is to predict y_* at a set of prediction points x_*. The GP can be represented as a sample from a multivariate Gaussian distribution as

$$\begin{bmatrix} y \\ y_* \end{bmatrix} \sim N\left(0, \begin{bmatrix} K & K_*^T \\ K_* & K_{**} \end{bmatrix}\right). \tag{7}$$

The three matrices in the covariance matrix are given by:

$$K = \begin{bmatrix} k(x_1,x_1) & k(x_1,x_2) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & k(x_2,x_2) & \dots & k(x_2,x_n) \\ . & . & \dots & . \\ . & . & \dots & . \\ . & . & \dots & . \\ k(x_n,x_1) & k(x_n,x_2) & \dots & k(x_n,x_n) \end{bmatrix}, \tag{8}$$

$$K_* = \begin{bmatrix} k(x_*,x_1) & k(x_*,x_2) & \dots & k(x_*,x_n) \end{bmatrix}, K_{**} = k(x_*,x_*), \tag{9}$$

where $k(x,x')$ is defined in Equation (2). For n training points and n_* test points, $k(x,x_*)$ is a $n \times n_*$ matrix. To get the posterior distribution over the function, we need to restrict the joint prior distribution to contain only those functions which

agree with the observed data points. Thus we need to condition the joint Gaussian prior distribution on the observation:

$$y_* \,|\, y \;\sim\; N(K_* K^{-1} y, K_{**} - K_* K^{-1} K_*^T). \tag{10}$$

Our best estimate y_* is the mean of this distribution:

$$\mu_* = K_* K^{-1} y\,, \tag{11}$$

and the variance in our estimate is

$$\sigma_*^2 = K_{**} - K_* K^{-1} K_*^T\,. \tag{12}$$

The above expressions can be written in a more simplified form. Consider the mean prediction as a linear combination of observations y. Thus, we look at the equation as a linear combination of n kernel functions (a weighing function), each of which is centered on a training point. Thus, we have that

$$\mu_* = \sum_{i=1}^{n} \alpha_i k(x_i, x_*), \tag{13}$$

where $\alpha = K^{-1} y$. The variance of a Gaussian process is the difference between two terms: the first term $k(x_*, x_*)$ is the prior covariance from which we subtract the information the observations or test points give us about the function. One particular implementation of Gaussian process is by using Cholesky decomposition, instead of directly inverting the matrix K. This is faster and numerically more stable. In MATLAB notations, the use of the following equations is recommended below:

$$L = cholesky(K) \tag{14}$$

$$\alpha = \frac{L^T}{\left(\frac{L}{y}\right)} \ . \tag{15}$$

The predictive mean Equation (11) becomes:

$$\mu_* = K_*^T \alpha \tag{16}$$

$$\upsilon = \frac{L}{K_*} . \tag{17}$$

The predictive variance is:

$$\sigma_*^2 = k(x_*, x_*) - \upsilon^T \upsilon \ . \tag{18}$$

The reliability of our Gaussian process is dependent on how well we select the covariance function $k(\bullet,\bullet)$. Thus, the choices of l, σ_f and σ_n in Equation (6) are vital. We determine their values by the maximum likelihood method. The marginal likelihood is the integral of the likelihood times its prior:

$$p\left(\theta \middle| X\right) = \int p\left(\theta \middle| y, X\right) p\left(y \middle| X\right) dy \, , \tag{19}$$

where $\theta = (l, \sigma_f, \sigma_n)$, as defined in Equation (6). The marginal likelihood is written on condition at the hyperparameters (the parameters of the covariance function) θ. We recall it as the log marginal likelihood since it is obtained through log-marginalize-marginal likelihood function over the latent function:

$$\log p(y|x,\theta) = -\frac{1}{2} y^T K_{yy}^{-1} y - \frac{1}{2}\log|K_{yy}| - \frac{n}{2}\log(2\pi) \quad (20)$$

where $K_{yy} = K_{xx} + \sigma_n^2 I$ is the covariance matrix for the noisy targets y and $[K_{xx}]_{ij} = k(x_i, x_j)$ is the covariance matrix for the noise-free latent. To maximize the posteriori estimate of θ, $p(\theta|x,y)$ has to be at its greatest. Thus, assuming we have little prior knowledge about what θ should be, we need to maximize the $\log(y|x,\theta)$ which is given by Equation (20). If the above method of Cholesky decomposition is used, then the log marginal likelihood can be calculated as

$$\log p(y|X) = -\frac{1}{2} y^T \alpha - \sum_i \log L_{ii} - \frac{n}{2}\log 2\pi \quad (21)$$

where L is specified on Equation (14).

3. Gaussian Process Regression

In statistics, regression analysis includes the techniques of analyzing the relationship between independent variables and dependent variables. The history of linear regression dates back to 1875 when Galton (1886) applies the technique to the inherited characteristics of sweet peas. Pearson (1986) presents a linear regression theory for a rigorous treatment of a regression model and their corresponding correlation. This model can only analyze the linear relationship between the independent variables and the dependent variables although the dependent variables can be transformed. The background theory in this section is taken from Shah (2009), and Rasmussen and Williams (2006).

The connection between the linear regression model and the Gaussian process regression (GPR) model comes from projecting the independent variables into a higher dimensional space where we may use the linear model. The concept of Gaussian process regression is named after Carl Friedrich Gauss because it is based on the Gaussian distribution.

A typical linear regression model is

$$y = f(x) + \varepsilon \text{ where } \varepsilon \sim N(0, \sigma^2). \tag{22}$$

A regression model in terms of GPR is

$$Y^T \sim N(\mu(x), k(x, x')). \tag{23}$$

The model of Y as a noise realization of μ is $p(y|\mu) = N(y|\mu, \sigma_n^2)$. Define $X = (x_1, ..., x_n)$, $[k_*]_i = k(x_*, x_i)$ and $[K_{xx}]_{ij} = k(x_i, x_j)$. For every data point, a vector k_* is "concatenated" as an extra line and column of the covariance matrix K_c to give rise to K_{c+1}, where $c = 1 \ldots N$ (N is dimension of feature space) is incremented every time a new k_* is added to K_c as follow

$$K_{c+1} = \begin{bmatrix} K_c & k_* \\ k_*^T & k(x_*, x_*) \end{bmatrix}. \tag{24}$$

Considering a zero mean function of the data ($\mu(x) = 0$) and $p(f(x_*)|y) = p(f(x_*), y) / p(y)$ thus

$$\begin{bmatrix} y \\ f(x_*) \end{bmatrix} \sim N\left(0, \begin{bmatrix} K_{xx} + \sigma_n^2 I & k_* \\ k_*^T & k(x_*, x_*) \end{bmatrix}\right), \tag{25}$$

when x^* is a test point and latent function $\mu_* = \mu(x_*)$; thus, we have that

$$\mu_* = Y^T (K_{xx} + \sigma_n^2 I)^{-1} k_{*,} \tag{26}$$

$$\sigma_*^2 = k(x_*, x_*) - k_*^T (K_{xx} + \sigma_n^2 I)^{-1} k_{*,} \tag{27}$$

$$K_{yy} = K_{xx} + \sigma_n^2 I. \tag{28}$$

A prediction distribution of GPR is

$$\log(p(y|X)) \propto -y^T K_{yy}^{-1} y - \log|K_{yy}| + c. \tag{29}$$

where c is a constant that is independent of the hyperparameters (Walder *et al.* 2008). In MATLAB notations, the use of the following equations is recommended below:

$$L = cholesky(K + \sigma_n^2 I). \tag{30}$$

Then L is substituted into Equations (15), (16) and (21) to determine the predictive mean, variance, and log marginal likelihood, respectively. Figure 4 shows a sampling observation with noise data, and Figure 5 shows the predictive distribution of GPR over a sampling observation with noise data in Figure 4. Even GPR distribution is not the best solution for fitting the curve function, but it gives predictions based on adjustable sets of parameters.

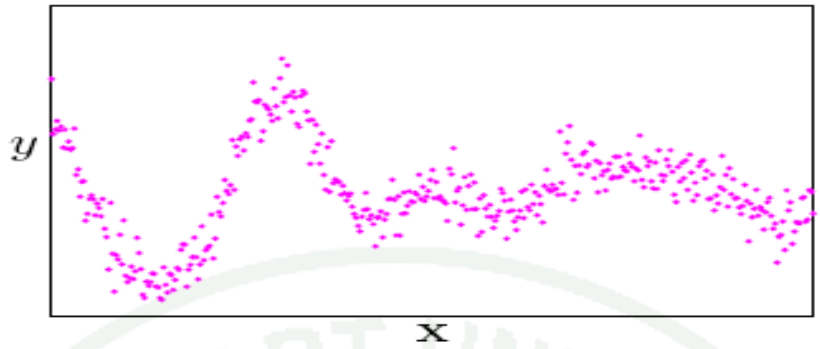


Figure 4 Sampling observation with noise data (Rasmussen, 2006)

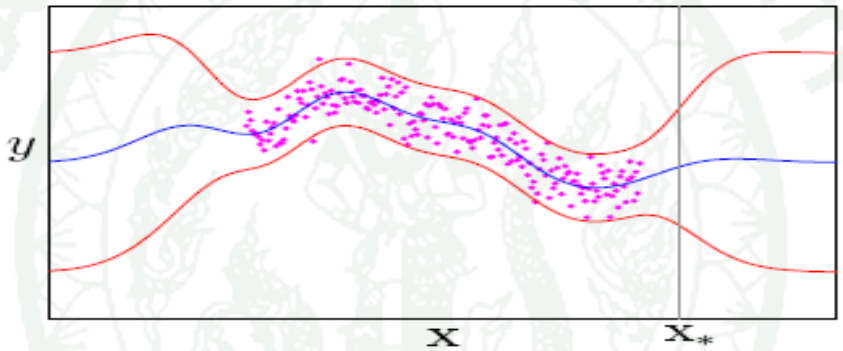


Figure 5 Predictive distribution of Gaussian process regression (Rasmussen, 2006)

4. Variation on Gaussian Process Regression

Ebden (2008) illustrates the GPR concept in a typical prediction problem. Given a set of random variables Y, he explains that the behavior of Y can be described by an underlying function $f(x)$ through the relation $Y = f(x) + N(0,\Sigma)$, where $N(0,\Sigma)$ is a normal random vector with mean of zero and covariance matrix Σ. Statistical methods can be used to approximate $E(Y|x^*)$ by estimating $f(x)$ from the given set Y.

Sollich and Williams (2005) use the equivalent kernel (EK) to understand GPR for large sample sizes based on a continuum limit. They use EK to estimate learning curves for GPR. EK provides a simple means to understand the learning curve of the behavior of GPR, even in the case where the learner's covariance function is not well matched to the structure of the target function.

Normally, Gaussian process terms have single outputs with a stationary covariance function and continuities because the covariance matrix must be positive definite. Meeds and Osindero (2006) develop a fully generative infinite mixture model for multi-model outputs of Gaussian processes with non-stationary covariance functions, discontinuities, multimodality and overlapping output signals. The infinite Gaussian mixture model is a generalization of finite Gaussian mixtures to an infinite number of components. This model is shown to be better than Rasmussen and Ghahramni (2002) model which is a conditional model by using stochastic indicator variables.

Boyle and Frean (2005) present an alternative to achieve Gaussian process model with multiple outputs by treating the Gaussian process with white noise convolved sources with smoothing kernels, and parameterizing the kernel instead. The applications of the model are limited because it is based on a covariance matrix.

Generally, GPR inputs must be statistically independent. Williams *et al.* (1998) present GPR with noise whose variances depend on input. They use a natural non-parametric prior for variable noise rates ($\delta(x, x')$) in Equation (6) and give an effective method of sampling the posterior distribution by using the Markov Chain

Monte Carlo. When applied to the data set with varying noise, the posterior noise rates obtained are well matched to the known structure.

5. The Squared Exponential Kernel

In the case of a GPR model with finite number of parameters, the covariance function K_{xx} can have at most as many non-zero eigenvalues as the number of parameters in the model (the material in this section is taken from Kalaitzis (2009), and Rasmussen and Williams (2006)). Hence, for any problem of any given size, the matrix K_{xx} is non-invertible. Ensuring that it is not ill-conditioned, the diagonal noise term is added to the covariance matrix. In an infinite-dimensional feature space ϕ, this issue does not occur as the features are integrated out and the covariance between data points is no longer expressed in terms of the features but by a covariance function. The covariance matrix K_{xx} are expressed in terms of the features ϕ

$$K_{ij} = \sigma_w^2 \sum_h \phi_h(x_i)\phi_h(x_j), \tag{31}$$

by considering a feature space and integrating with respect to their core. Specifically, we introduce the function $\phi(x)$ which maps a D-dimensional input vector x into an N dimensional feature space $f(x) = \phi(x)^T w$ then an Equation (31) becomes

$$K_y(x_i, x_j) = \lim_{N\to\infty} \frac{\sigma_w^2}{N} \sum_{h=1}^{N} \phi_h(x_i)\phi_h(x_j) = S\int_{-\infty}^{\infty} dh\phi_h(x_i)\phi_h(x_j)$$
$$= S\int_{-\infty}^{\infty} dh \ \exp\left(-\frac{(x_i-h)^2}{2r^2}\right) \ \exp\left(\left(-\frac{(x_j-h)^2}{2r^2}\right)\right)$$
$$= \sqrt{\pi r^2}S \ \exp\left(-\frac{(x_i-x_j)^2}{4r^2}\right),$$

(32)

where one ends up with an infinitely differentiable function on an infinite dimensional space of features. Taking the constant out-front as the signal variance σ_f^2 and dividing the denominator by two, gives rise to the standard form of the unvaried squared-exponential (SE) covariance function below,

$$k(x_i, x_j) = \sigma_f^2 \exp(-\frac{1}{2}(x_i - x_j)^T M(x_i - x_j)) + \sigma_n^2\delta_{ij}, \tag{33}$$

where M is a symmetric positive-definite (SPD) matrix containing the inverse hyperparameters of the kernel and δ_{ij} is a Kronecker delta function which is unity if $i = j$ and zero otherwise. The squared-exponential (SE) is a stationary kernel; it is a function of $d = (x_i - x_j)^T$ which makes it translation invariant in the input space. In the standard multivariate form Equation (34),

$$M = diag(\ell)^{-2}, \ \ell = (\ell_1, \ldots, \ell_D)^T, \tag{34}$$

where $\theta = (\{M\}, \sigma_f^2, \sigma_n^2)^T$ is a vector containing the hyperparameters of the SE kernel. Each ℓ_d, $d = 1, \ldots, D$ is a characteristic length scale associated to the d^{th} dimension of X and governs the amount that $f(x)$ varies along that dimension. This kernel is also known as the automatic relevance determination SE (ARDSE) kernel

because of its ability to highlight the relevance of attributes to the training target. A small lengthscale ℓ_d would mean that $f(x)$ varies very rapidly along the d^{th} dimension, and a large lengthscale would mean that $f(x)$ is almost a constant function of input x_d (see Figure 6)

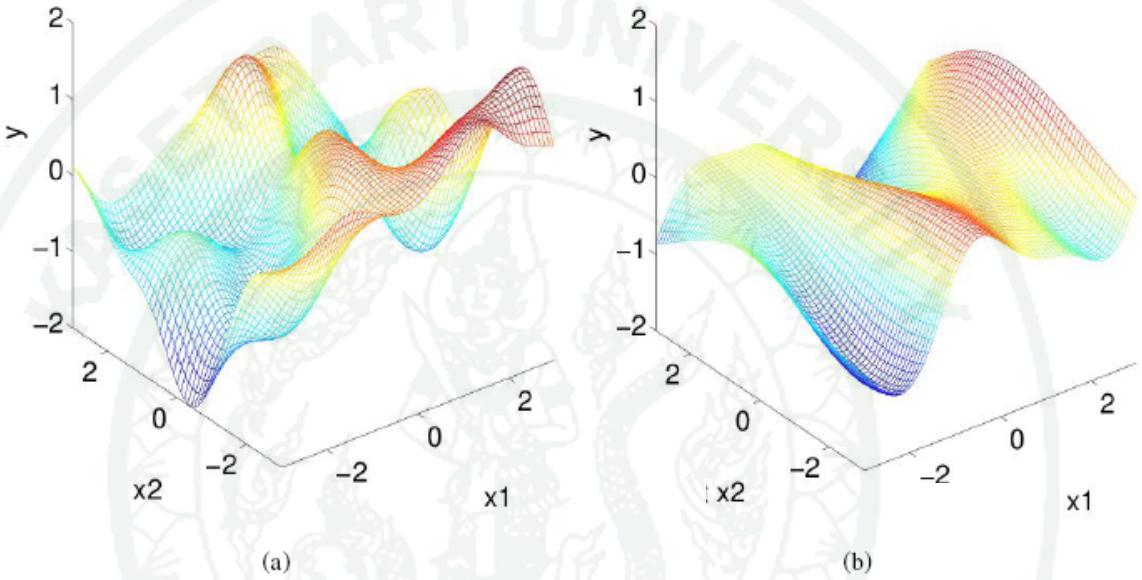


Figure 6 Two-dimensional functions drawn at random from noise-free exponential kernel (σ_n^2=0) Gaussian processes: (a) Function varies the same along both dimensions with hyperparameters $\ell = (1,1)^T$. (b) Function varies less rapidly along the dimension of x_2 with hyperparameters $\ell = (1,2)^T$. (Rasmussen and Williams, 2006)

This trait of the SE kernel becomes very powerful when combined with hyperparameter adaptation. Other hyperparameters include the signal variance σ_f^2 which is a vertical scale of function variation and the noise variance σ_n^2. It is not a hyperparameter of the SE itself, but unless we consider it as a constant in the noisy case, its adaptation can give different explanations about the latent function that generated the data. One can also combine covariance functions as long as they are

positive-definite. Examples of valid combined covariance functions include the sum and convolution of two covariance functions. In fact, Equation (33) is a combined the SE kernel with a covariance function of isotropic Gaussian noise.

6. Sparse Multiscale Gaussian Process Regression

Let $\varsigma(k)$ be a Gaussian process defined by zero mean Gaussian random variable with covariance K_{xx}. Let μ be drawn from $\varsigma(k)$, called μ_x, distributed according to (Walder *et al.* 2008)

$$p_{\mu x}(\mu) = N(\mu|0, K_{xx}). \tag{35}$$

To determine the likelihood of a function expressed as a summation of fixed basis functions, the probability density function (p.d.f.) of μ_x is set to be $\sum_{i=1}^{m} c_i \mu_i$, for some $c_i \in R$. At the end, an infinite limit of the above case is considered, so taking the limit n $\rightarrow \infty$ of uniformly distributed points x leads to the following p.d.f. for $\varsigma(k)$. Equation (35) can be drawn from the following p.d.f. for $\varsigma(k)$,

$$p_{\varsigma(k)}(\sum_{i=1}^{m} c_i \mu_i) = \left|2\pi K_{xx}^{-1}\right|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}\sum_{i,j=1}^{m} c_i c_j \psi_k(\mu_i, \mu_j)\right). \tag{36}$$

A norm in the Hilbert space with reproducing kernel K will be denoted by $\|\bullet\|_k$, and inner product is denoted by $\langle \bullet, \bullet \rangle_K$ (Melkumyan and Ramos 2009). Let $\mu = K_{xx}\alpha$, and K_{xx} is invertible, then $\alpha = K_{xx}^{-1}\mu$. Following this finite analogy, k^{-1} is intended for the function which, for

$$\mu = \int \alpha(x) k(x, \bullet) dx, \tag{37}$$

satisfies $\int \mu(x)k^{-1}(x,\bullet)dx = \alpha(\bullet).$ Define $M_k : \alpha \to M_k\alpha = \int \alpha(x)k(x,\bullet)dx,$ $k(x,y) = cg(x,y,\sigma)$, where $c > 0$, $\sigma > 0 \in R^d$ and g is a normalized Gaussian on $R^d \times R^d$ with diagonal covariance matrix. Then we have that

$$\alpha(\bullet) = \int (M_k\alpha)(x)k^{-1}(x,\bullet)dx, \tag{38}$$

$$\mu_i(x) = g(x,v_i,\sigma_{ij}) = \left(\frac{1}{c}M_{cg(\cdot,\cdot,\sigma)}g(\bullet,v_i,\sigma_i-\sigma)\right)(x). \tag{39}$$

As the covariance function and the basis functions are all Gaussian, we can obtain, in closed form,

$$\begin{aligned}\psi_k(\mu_i,\mu_j) &= \iint k^{-1}(x,y)g(x,v_{i,}\sigma_i)\left(\frac{1}{c}M_{cg(\cdot,\cdot,\sigma)}g(\bullet,v_j,\sigma_j-\sigma)\right)(y)dxdy \\ &= \frac{1}{c}g(x,v_{i,}\sigma_i+\sigma_j-\sigma).\end{aligned} \tag{40}$$

From Equations (38), (39), and (40), we can write a prior probability of arbitrary Gaussian mixtures as shown in Equation (40)

$$P_\zeta(cg(\bullet,\bullet,\sigma))\left(\sum_{i=1}^{m}c_ig(\bullet,v_{i,}\sigma_i)\right) \propto \exp\left(-\frac{1}{2}\sum_{i,j=1}^{m}\frac{1}{c}c_ic_jg(v_{i,}v_{j,}\sigma_i+\sigma_j-\sigma)\right). \tag{41}$$

The neglected factor is equal to the inverse of the integral of the right hand side of Equation (41) with respect to all functions $\sum_{i=1}^{m}c_ig(\bullet,v_i,\sigma_i)$ (Walder *et al.* 2008).

7. Hyperparameter adaptation by using automatic relevance determination

In general, the automatic relevance determination (ARD) functionality of the SE kernel allows one to give intuitive interpretations of the adapted hyperparameters by optimizing hyperparameters. ARD provides a method of maximum likelihood that allows the relative importance of different inputs to be inferred from the data as step below:

1. Give the hyperparameter vector which controls how far away from zero each weight is allowed to go.

2. Maximize the marginal likelihood of the model to train the hyperparameter vector and the outcome of this optimizing method is many elements of hyperparameter vector go to infinity.

An example below shows the Gaussian process context of automatic relevance determination, or ARD. Suppose we consider a Gaussian process with 2-D input space $\boldsymbol{x} = (x_1, x_2)$. The SE kernel function formulation can be written as:

$$k(x, x') = \theta_0 \exp\left\{-\frac{1}{2}\sum_{i=1}^{2}\eta_i (x_i - x'_i)^2\right\}. \tag{42}$$

In Figure 7, we see that, as a particular parameter η_i becomes small, the function becomes relatively insensitive to the corresponding input variable x_i. By adapting these parameters to a data set using maximum likelihood, it becomes possible to detect input variables that have little effect on the predictive distribution, because the corresponding values of η_i will be small. This can be useful in practice because it allows inputs to be discarded.

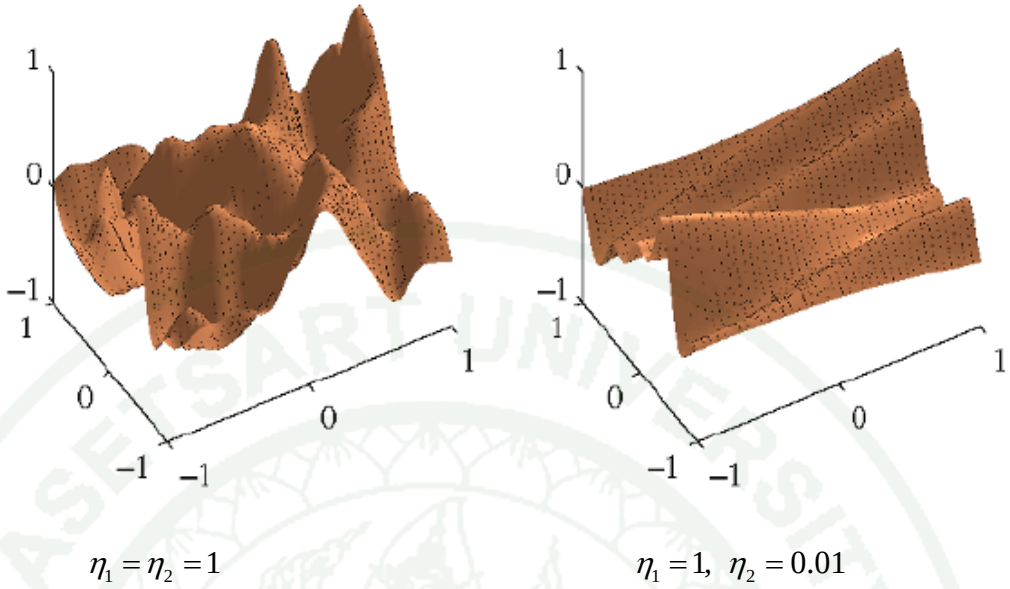


Figure 7 Three-dimensional functions when adapting η_i parameter where two horizontal axis is inputs(x_1 and x_2) and vertical axis is output of a function ($k(x, x')$).

An ARD is illustrated using a simple synthetic data set having three inputs x_1, x_2 and x_3 in Figure 8. The target variable t, is generated by sampling 100 values of x_1 from Gaussian distribution, evaluating the function $\sin(2\pi x_1)$ and then adding Gaussian noise. Values of x_2 are given by copying the corresponding values of x_1 and adding noise, and values of x_3 are sampled from an independent Gaussian distribution. Thus x_1 is a good predictor of t, x_2 is a more noisy predictor of t, and x_3 has only chance correlations with t.

The marginal likelihood for a Gaussian process with ARD parameters η_1, η_2 ,and η_3 is optimized using the scaled conjugate gradients algorithm. From Figure 8 that η_1 converges to a relatively large value, η_2 converges to a much smaller value, and η_3 becomes very small indicating that x_3 is irrelevant for predicting t

$k(x, x')$

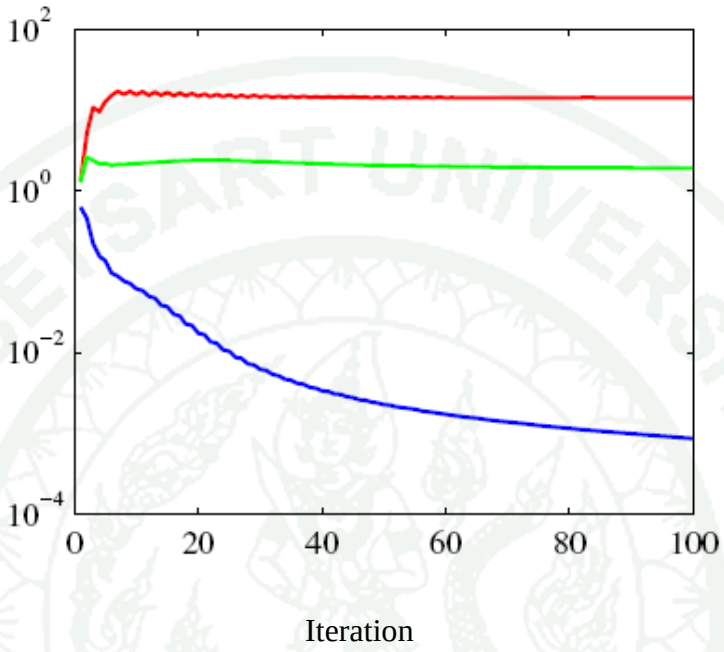


Figure 8 Illustration of automatic relevance determination in a Gaussian process for a synthetic problem having three inputs x_1, x_2 and x_3 for which the curves show the corresponding values of the hyperparameters η_1 (red), η_2 (green), and η_3 (blue) as a function of the number of iterations when optimizing the marginal likelihood.

The ARD framework is easily incorporated into the exponential-quadratic kernel to give the following form of kernel function, which has been found useful for applications of Gaussian processes to a range of regression problems.

In our research, we use Gaussian process for machine learning tool box (GPML tool box) with an SE covariance function and allow the separate length scale for each input with an ARD and independent noise.

MATERIALS AND METHOD

Materials

1. Hardware: Laptop computer, Lenovo Group Limited, 7735E21 model, OS name is Microsoft Window XP Professional Service Pack 3.
2. Special software: MATLAB R2009A for Gaussian process analysis. (GPML)
3. Literature: The literature will be copied from the website of the main library of Kasetsart University and the Gaussian process website.
4. Data Sources:

4.1 Test problem from http://softlib.rice.edu/pub/tsplib/tsp/., January 30, 2010, unpublished.

4.2 Geographic TSP calculated distance formula from http://www.neverreadpassively.com/2008/05/tsplib-library-of standard-tsp.html

4.3 Gaussian process for Machine Leaning (GPML) software from http://www.gaussianprocess.org/gpml/chapters/

4.4 MATLAB manual from http://www.mathworks.es/

Method

We use GPR technique for creating a prediction function and solving TSP. We define our notations as follows:

y_r = total distance of sampling tours r,

$r = 1, 2, 3, \ldots, R$

$x_{r,i}$ = path i of the sampling tour r, where

$$x_{r,i} = \begin{cases} 0: \text{ if path } i \text{ does not exist on tour } r \\ 1: \text{if path } i \text{ exists on tour } r \end{cases}$$

$$a_i = \text{distance of path } i.$$

Our objective function is

$$\text{Min } y_r = \sum_{i=1}^{m} a_i x_{r,i}. \tag{43}$$

We apply the GPML toolbox developed by Rasmussen and Williams (2006) to fit the GPR function and optimize the prediction function. From Equation (27), GPML toolbox defines Equation (44) in term of α below

$$\alpha = (K_{xx} + \sigma_n^2 I)^{-1} Y^T, \tag{44}$$

thus prediction term is

$$\mu_* = k_* \alpha. \tag{45}$$

The objective function for applied GPR on TSP is

$$\text{Min } \mu_* = k_* \alpha, \tag{46}$$

where x^* is a TSP tour.

The GPML toolbox use a GPR with a squared exponential covariance function and allow a separate length scale for each input with ARD for determining hyperparameters and independent noise:

$$k_* = \sigma_f^2 \exp(-\frac{1}{2}(x_* - x_i)^T M (x_* - x_i)) + \sigma_n^2 \delta_{*,i}, \quad (47)$$

where σ_f is a hyperparameter, δ_{pq} is a Kronecker delta function, $M = diag(\ell)^{-2}$, and ℓ is a vector of positive value hyperparameter. The steps in applying GPR are divided into two cases:

- No reduction of independent variables.

1. Generate sampling tours and calculate their corresponding total distance
2. Use a GPML tool box to obtain a GPR function to approximate an optimal TSP tour.
3. Transform an optimal solution to a TSP tour.

- Reduction of independent variables.

1. Construct a subtour by a greedy heuristic.
2. Generate sampling tours and calculate their corresponding total distance
3. Use a GPML tool box to obtain a GPR function to approximate an optimal TSP tour.
4. Transform an optimal solution to a TSP tour.

The second case is set up for reducing the size of the search space by reducing the number of nodes and reducing a computational time by constructing a subtour with n_f nodes. When the number of nodes in a TSP decreases, α as defined can be calculated faster. The flow diagram of our TSP calculation is shown in Figure 9.

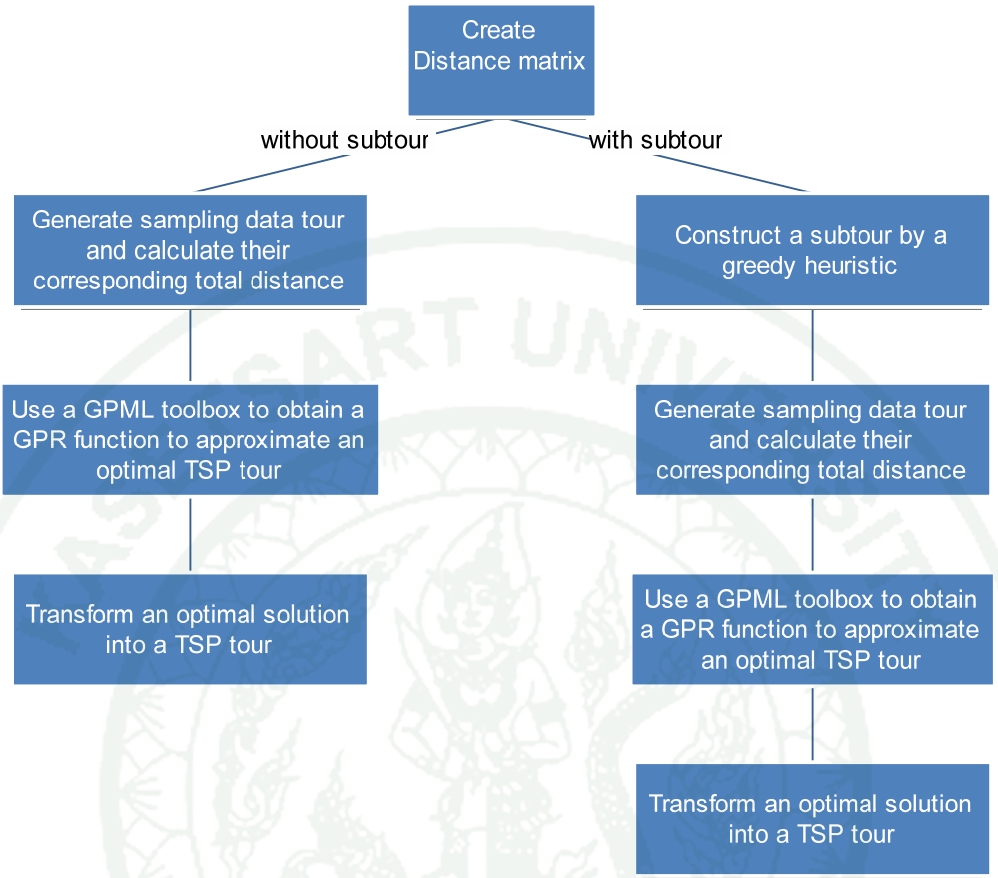


Figure 9 Flow Diagram of our Heuristic method

The steps to apply GPR for TSP are explained in details below:

2.1 Construct a subtour by a greedy heuristic.

First, we search a distance matrix to find a minimum distance d_{ij} where d_{ij} is the ij^{th} element of the distance matrix D. Then these two nodes are connected to form a subtour. These steps are repeated until we get n_f nodes in a subtour. "SearchNodeMin" function is used to construct a subtour (Figures 10-11).

```
[Fit_path,Fit_node] =
SearchNodeMin(Number_Node,Dis_matrix,N_SearchNodeFix)
```

Figure 10 "SearchNodeMin" function in MATLAB

This "SearchNodeMin" parameters are described below:

- Number_Node = Total number of nodes or cities in TSP (n).
- Distance Matrix = Distance matrix of the TSP matrix of size[$n \times n$]
- N_SearchNodeFix = Number of paths in subtour matrix of size [n_f -1]
- Fit_Path = Paths in subtour matrix of size [$n_f \times 2$]
- Fit_node = Nodes in subtour matrix of size [$n_f \times 1$]

This function return a subtour matrix. The programming flow diagram is explained in Figure 11 below:

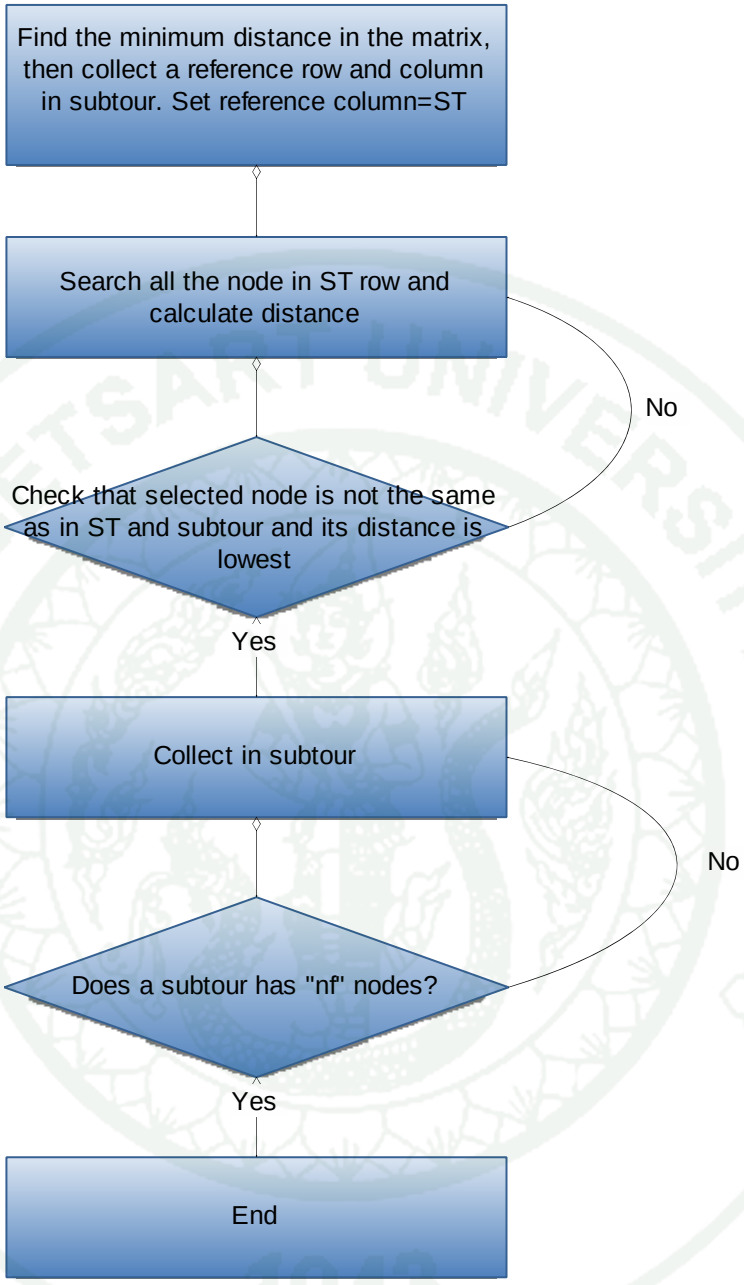


Figure 11 Flow chart of a greedy heuristic to construct a subtour

2.2 Generate sampling tours and calculate their corresponding total distance.

Tours and their corresponding total distance are generated by starting with a subtour of size n_f. We randomly permute the remaining $n-n_f$ nodes, by

using “Randperm” function in MATLAB, to form a complete tour. We repeat this step r times to get r random tours. Sample tours are aggregated into a data matrix X, whose dimension is $[r \times m]$, where $m = \sum_{i=1}^{n-1}(n-i)$ and its corresponding total distance into a response vector Y, whose dimension is $[r \times 1]$. To generate sampling tour and their corresponding total distance, we develop MATLAB functions below:

- “notmain”
- “Gen_routhfixstart” and “Gen_routhfixstartSearch”

The “notmain” function is used to return nodes that are not in a subtour

```
[possibleNodeRandom] = notmain(main,node)
```

Figure 12 “notmain” function

The “notmain” parameters are described below:

- main = Nodes in subtour $[1 \times n_f]$
- node = Number of node in the problem $[n]$

The “notmain” return the nodes which are not in subtour matrix. Their matrix size is $[(n-n_f) \times 1]$. The “Gen_routhfixstart” and “Gen_routhfixstartSearch” functions are used to generate a tour and their corresponding total distance, but the “Gen_routhfixstartSearch” functions is used when we want to decrease the size of the problem by creating subtours.

```
[total_dist,pop] =
Gen_routhfixstart(Number_Node,Dis_matrix,N_Random,N_iter)

[total_dist,pop,kk] = Gen_routhfixstartSearch
(Number_Node,number_fixnode,Dis_matrix,N_Random,N_iter,startnode,Endnod
e,possibleNodeRandom,fixRoutemetric)
```

Figure 13 "Gen_routhfixstart" and "Gen_routhfixstartSearch" functions

"Gen_routhfixstart" and "Gen_routhfixstartSearch" parameters are described below:

- Number_Node = Number of nodes in problems (n)
- number_fixnode = Number of nodes in subtour (n_f)
- Distance matrix = Distance matrix of the TSP.
- N_Random = Number of sampling tours (R)
- N_iter = Number of desired iterations for the algorithm to run
- startnode = The beginning node in subtours
- Endnode = The end of node in the subtours
- possibleNodeRandom = Nodes that are not in subtour matrix.
- $[1\times(n-n_f)]$
- fixRoutemetric = Nodes in subtour matrix $[1\times n_f]$

N_iter is number of iteration for creating a group of R sampling data but in our experiment we set to use only one iteration.

2.3 Use a GPML toolbox to obtain a GPR function to approximate an optimal TSP tour

A GPR function is created from sampling tours and their corresponding total distance for predicting a minimum total distance. The steps are described as follows:

- Determine a starting solution, which is a TSP tour.
- Estimate parameters $\alpha()$ and covariance function $k(x,\bullet)$ in Equations (16) and (18).
- Using a GPR function, μ, and $\alpha()$ from Equations (16) and (18) determine a route with minimum distance for the GPML implementation is called "minimize.".

The programming function for using approximate an optimal TSP tour is in Figure 14.

```
[n,D]=size(x);                                                    (1)
covfunc = {'covSum', {'covSEard','covNoise'}};                    (2)
logtheta0 = zeros(D+2,1);                                         (3)
logtheta0(D+2) = -1.15;                                           (4)
logtheta = minimize(logthetao, 'gpr', -100, covfunc, x, y);       (5)
[X, fX, i] = minimize2( X_0 ,-100,logtheta, covfunc, x, y);       (6)
```

Figure 14 GPR functions to approximate an optimal TSP tour

The detail of these programming in Figure 14 are shown below:

Line (1): Find the number of rows and columns in X .

Line (2): Specify a covariance function made up of the sum of a squared exponential (SE) covariance term with ARD, and independent noise.

Line (3): Create a zero matrix vector

Line (4): Set a starting point of hyperparameter (X_0).

Line (5): Train the hyperparameter by maximizing the approximate marginal likelihood of the SD method

Line (6): Predict the optimal TSP tour and total distance.

The “minimize2” modifies the original “minimize.” in GPML so that it can solve the TSP.

2.4 Transform an optimal solution into a TSP tour.

The route that we obtain in Part 2.3 (the predicted route) may not be a TSP tour, so we need to construct a tour from it. We first build disjoint subtours and connect them to form the rest of a TSP tour by a greedy heuristic, that is to select links with small distance first. After that, we link the rest of a TSP tour with a subtour that we create in part 2.1.

To transform an optimal solution into a TSP tour is using the step of programming function below:

- “GetrouteX” function
- “Fitnode” function
- “Fit” function

The “GetrouteX” function returns a decimal path matrix from the predicted binary tour.

```
[RoutX,XX,CharacterX] = GetrouteX(X,node, possibleNodeRandom)
```

Figure 15 "GetrouteX" function

"GetrouteX" parameters are described below:

- X = Predicted solution matrix in term of a binary tour [$1\times m$] (from "minimize2" function)
- node = Number of nodes that are not in subtour ($n-n_f$)
- possibleNodeRandom = Matrix of node that are not in subtour [$1\times(n-n_f)$] (from " notmain" function)
- RouteX = Predicted solution matrix in term of decimal path matrix [$2\times(n-n_f+1)$]
- XX = Predicted solution matrix in term of decimal path matrix [$2\times m$]
- CharacterX = Matrix that explains the meaning of predicted solution matrix in term of a binary tour

The decimal path matrix is transformed to a TSP tour by using "Fitnode" function for creating T matrix and using "Fit" function for creating HH matrix based on the programming flow in Figure 18.

```
[T,H] = Fitnode(RouteX,node,startnode,distancematrix)
```

Figure 16 "Fitnod" function

"Fitnod" parameters are described below:

- RouteX = Predicted solution matrix in term of decimal path matrix $[2\times(n-n_f+1)]$
- node = Number of nodes that are not in the subtour
- $(n-n_f)$
- startnode = Starting search node in T matrix
- distancematrix = Distance matrix of problem
- T = T matrix
- H = H matrix

```
[HH] = Fit (T,node,H, distancematrix, possibleNodeRandom)
```

Figure 17 "Fit" function

"Fit" parameters are described below:

- T = T matrix
- node = Number of nodes that are not in subtour
- $(n-n_f)$
- H = H matrix
- distancematrix = Distance matrix of problem
- possibleNodeRandom = Nodes that are not in subtour matrix. $[1\times(n-n_f)]$
- HH = HH matrix

The detail of transforming a predicted route into a TSP tour is shown in Figure 18. This lowest distance is selected in “T route”, and we find the path which can connect with the end of T route and that has a lowest distance. This step is repeated until we cannot find any more connecting path. The nodes which are not in T route is called “K matrix” and the routes which are not in T route is called “H matrix”. Then we delete the routes in H matrix which have the same nodes as T route and delete the node in K matrix which have node the same as H matrix. Then we select the lowest distance path in H, collect it in HH matrix and repeat this step until we select all of paths in H. We collect nodes of K which have lowest path distance in HH until all of nodes in K are selected. After that we combine HH matrix, T matrix and subtour. Finally, we calculate total distance.

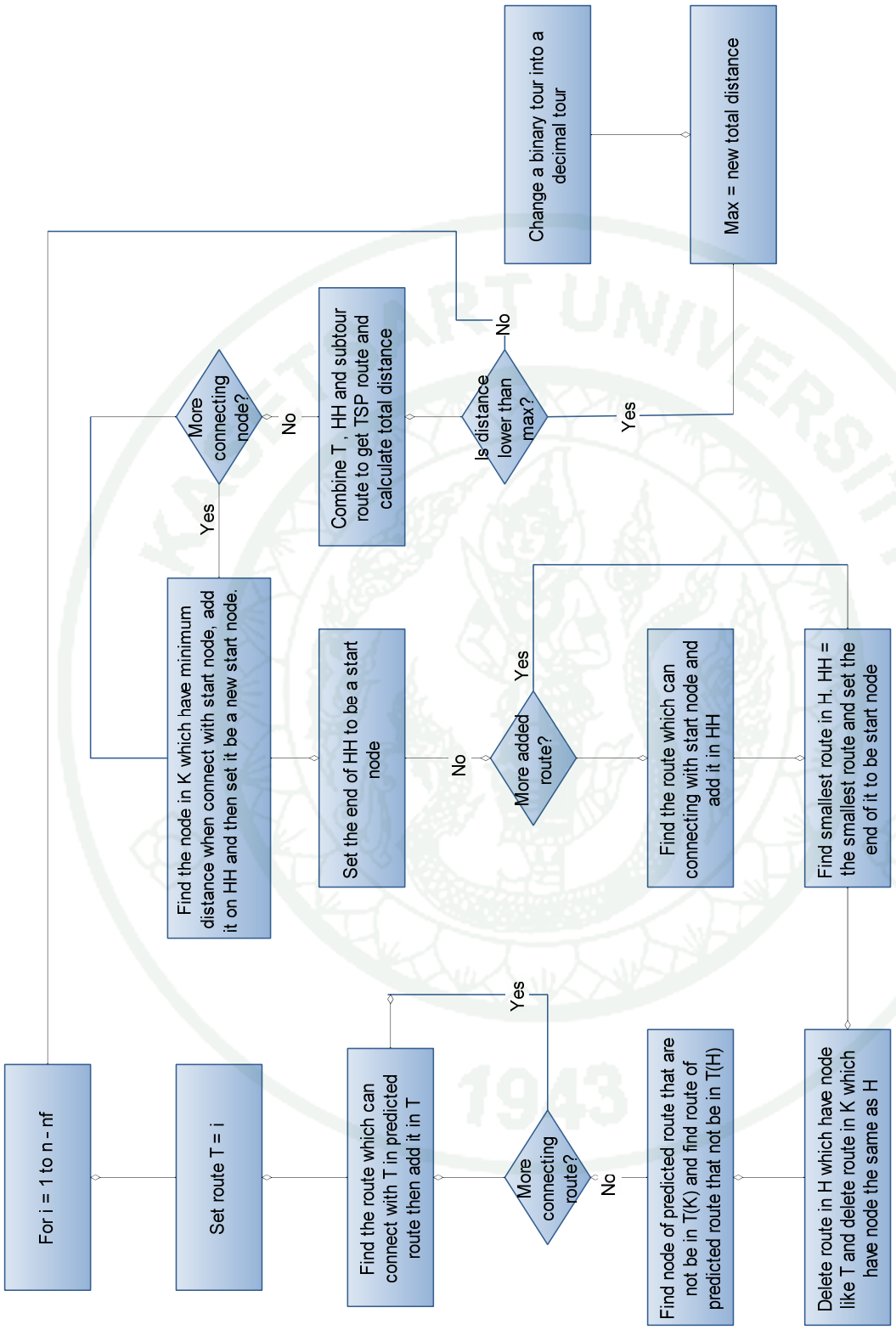


Figure 18 Transform a GPR optimal solution into a TSP tour flow chart.

RESULTS AND DISCUSSION

We fist describe details of test problems in Section 1 and the results of experiments are shown in Section 2.

1. Test problems

We use the geographical TSP from TSPLIB (Bixby and Reinelt 1995, http://softlib.rice.edu/pub/tsplib/tsp/).

1.1 96 Africa-Sub: 96 cities with the possible number of solutions is 9.92×10^{149} (14!) and the true optimal solution is 52,277.9 km. (Optimum tours are in appendix Table A4).

1.2 Ulysses22: 22 cities with the possible number of solutions is 1.12×10^{21} (22!) and the true optimal solution is 6,945.2. km. (Optimum tours are in appendix Table A1).

1.3 Ulysses16: 16 cities with the possible number of solutions is 2.09×10^{13} (16!) and the true optimal solution is 6,795.8 km. (Optimum tours are in appendix Table A2).

1.4 Burma14: 14 cities with the possible number of solutions is 8.71×10^{10} (14!) and the true optimal solution is 3,356.1 km. (Optimum tours are in appendix Table A3).

In the experiment, the sample sizes (R) for GPR are 500, 1500, 2500 and 3500.

2. Results of numerical experiments

We have two control parameters: n_f , is the number of nodes in a subtour and R is the number of random tours in a sample for creating a GPR function. The results are shown in Tables 1-2 and Figure 19.

Table 1 shows the predicted minimum distance from the GPR functions, but these tours may not form TSP tours. The underlined numbers are minimum distance for a given n_f . The best cases seem to be when the sample size is largest (R =3500). We can conclude that when R and n_f increase our method seem to predict a better solution. The best scenarios for each test problems in Table 1 are close to the true optimal solutions, so we can use these lowest predictions to be our target to find TSP tours as shown in Table 2.

As expected, we see that as the test problems become more difficult (higher number of cities), the distances of our TSP tours deviate from the optimal tours significantly. For the Burma14 problem, the best relative deviation is only 16% whereas for the Ulysses22 problem, the best relative deviation is 33%. The benefit of reducing the search space by initially creating a subtour (n_f > 0) is greater for a large test problem. In Figure 19, we see that the relative deviations of n_f = 10, 12 and 14 are smaller than the case with n_f = 0. For the Ulyssess22 problem, while setting n_f =2 and 4 do not seem to help improve the quality of the solutions.

Considering that our heuristic sees only a very small fraction of the search space, less than 4×10^{-6} %, its performance is impressive. For the test problem with 22 cities, our TSP tour gives the distance within 33% to 76% of the optimal distance,

ignoring the possible outlier at 117%. The numerical results seem to suggest that we get a better solution when we increase the sample size R.

However that result shown in Table 2 and Figure 19 are based on one experiment for each case. To see the trend of results, we repeat experiments again and the result shown in Figure 20.

From Figure 20, we see that if n_f increases, the 95% of confident intervals are shorter. As a result, our predictions are more precise when we increase n_f. Moreover, 95% intervals include the lowest optimal solution. Thus, if we repeat the experiment, we will have more opportunity to get better predicted TSP tours.

Table 1 Predicted minimum distance from a GPR function before being transformed to TSP tours.(the underlined entries are the minimum of that row.)

Ulysses22				
	R			
n_f	500	1,500	2,500	3,500
0	12,493.4	12,246.8	11,966.7	11,694.5
2	11,193.1	11,188.5	10,584.0	10,579.6
4	10,175.1	9,578.1	9,565.3	10,137.2
6	9,929.3	8,869.2	9,398.7	9,013.7
10	8,316.8	9,162.2	9,131.3	7,043.2
12	6,699.8	6,697.2	6,379.9	9,138.9
14	9,222.8	6,137.8	9,217.5	6,093.0
Ulysses16				
0	9,080.1	10,308.0	9,829.0	9,289.8
2	9,180.7	8,362.2	8,474.6	8,955.4
4	7,454.6	7,590.5	7,571.8	7,763.3
6	7,944.5	7,492.9	7,040.0	6,898.9
Burma14				
0	4,305.5	4,131.9	4,253.3	3,803.7
2	4,237.7	4,305.2	3,852.2	4,136.7
4	4,044.3	3,454.5	4,018.3	3,830.4

Table 2 Distance of TSP tours after being transformed to a TSP tour.

Ulysses22 (Optimum value = 6945)				
n_f	R			
	500	1,500	2,500	3,500
0	13,174.3	12,592.2	14,206.2	11,694.5
2	13,037.2	13,326.6	13,059.6	11,879.7
4	11,905.6	10,900.8	12,292.0	12,236.0
6	12,006.4	12,156.9	12,237.4	15,064.9
10	9,917.0	10,034.7	11,529.5	10,610.4
12	9,624.2	9,847.6	9,476.0	9,583.9
14	9,222.8	10,712.9	10,528.5	9,222.8
Ulysses16 (Optimum value = 6795)				
0	9,572.2	11,282.0	12,991.0	9,289.8
2	9,761.5	11,993.8	9,994.8	10,393.2
4	7,454.6	8,387.1	8,853.7	10,507.5
6	9,729.9	8,949.0	8,440.0	6,898.9
Burma14(Optimum value = 3356)				
0	4,305.5	4,237.9	4,705.7	3,886.1
2	6,285.2	5,130.6	6,005.5	5,250.1
4	5,602.5	4,289.9	5,483.9	4,742.6

Relative deviations from the optimal solutions of after transformed to a TSP tour are shown in Figure 19.

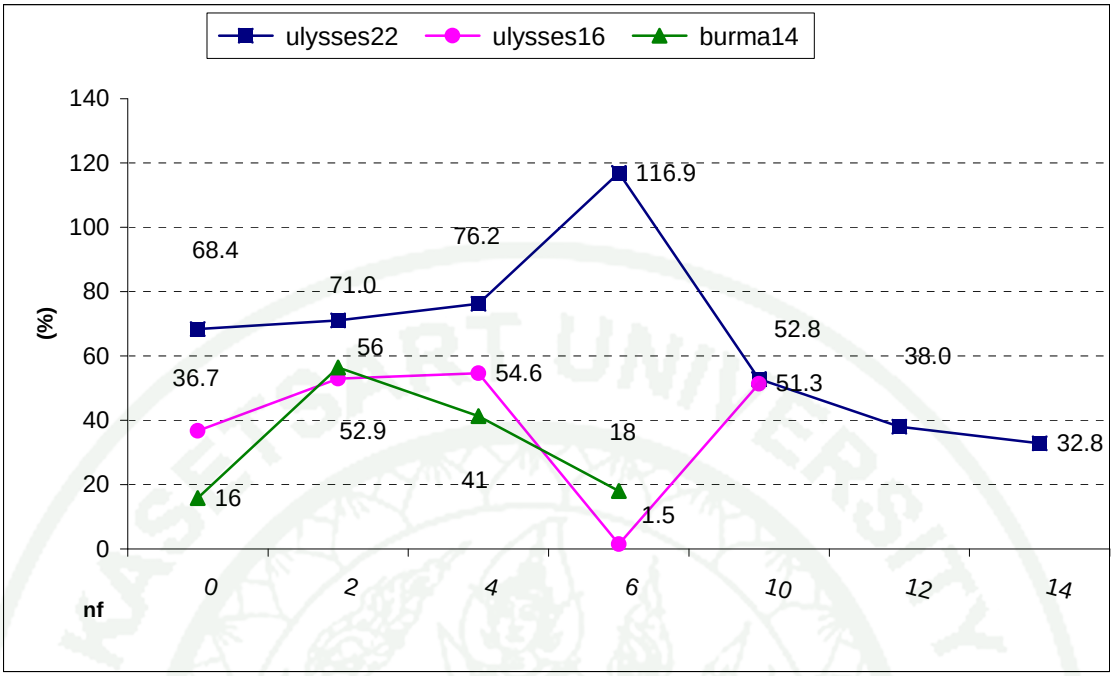


Figure 19 Percentage of prediction total distance at after transformed to a TSP tour and real optimal value with R =3500 and 1 replications.

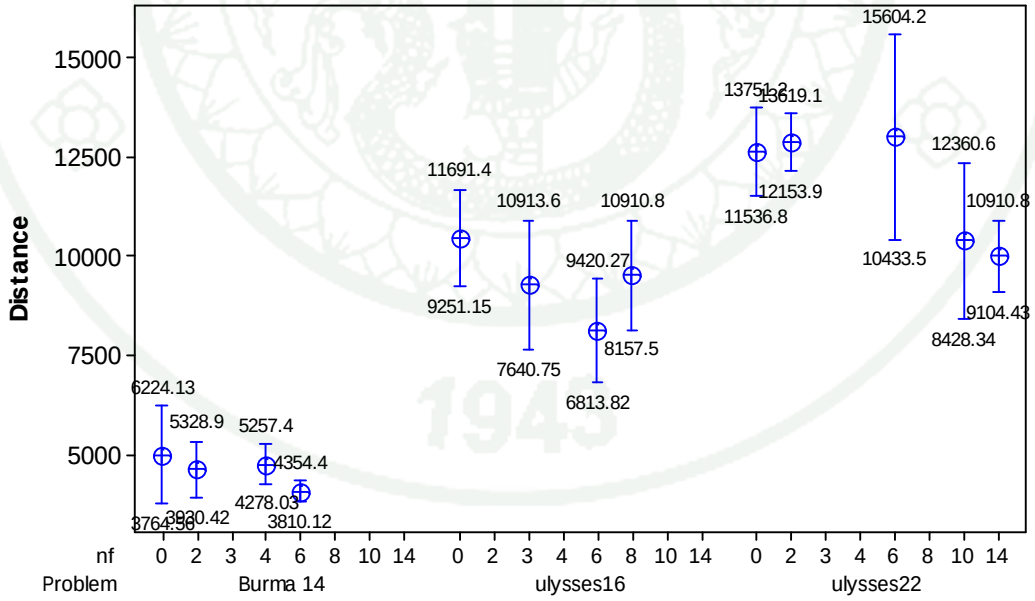


Figure 20 95% Confidence Intervals of repeating experiment at after transformed to a TSP tour with R =3500 and 4 replications. Horizontal axis is n_f and vertical axis is distance.

n_f does not matter when a number of cities is small but when a number of cities is large, large n_f helps to predict the better solution. From Figure 20, at Ulysses 22 problem, comparing with n_f = 0 and n_f =14, predicted value of n_f =14 is better than n_f = 0.

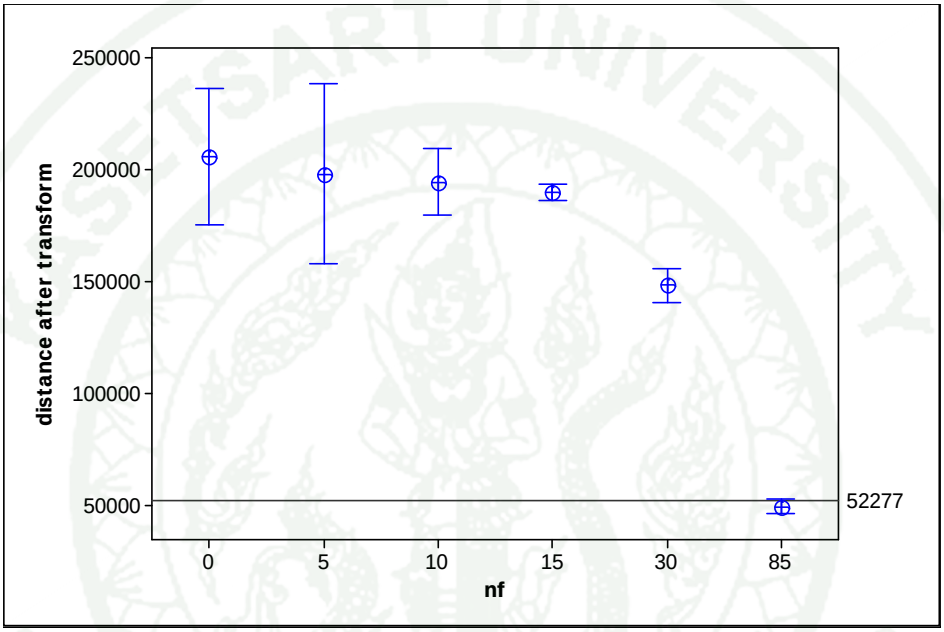


Figure 21 95% Confidence Intervals of repeating experiment at R =2500 and 3 replications at 96 cities of 666-cities Africa

In Figure 21 and Table 3, although a number of cities in this problem are 96 cities that are means this problem is more complexity than the last problems (Ulysses22, Burma 16 and Burma 14), the results from applied GPR in TSP at n_f =85 are better for predicted the TSP tour and their corresponding total distance compare with optimum distance from TSPLIB (52,277). Thus increasing a number of n_f in subtour and increasing R, the predicted values are closer to the actual optimum solution as shown in Figures 20-21. Finally, we can conclude that applied GPR in TSP is a powerful heuristic method for solving the TSP which is an NP-Hard problem.

Table 3 Distance of TSP tours after and before being transformed to a TSP tour of 96 cities of 666-cities Africa problem and 3 replications.

n_f	**Distance Before**	**Distance After**
0	178,171.35	217,850.81
0	177,852.82	193,327.82
0	179,013.57	207,356.83
5	175,478.64	216,858.57
5	173,920.71	188,153.22
5	174,190.50	189,585.07
10	163,095.47	201,510.02
10	167,241.25	190,020.23
10	165,204.68	192,247.27
15	157,373.69	190,100.37
15	160,042.21	188,292.77
15	160,070.82	191,396.45
30	126,786.33	147,495.90
30	127,097.54	151,531.07
30	124,544.93	145,304.33
45	56,140.39	71,549.36
45	81,935.82	89,776.47
45	62,433.32	80,114.00
85	42,008.44	48,036.16
85	43,159.27	48,880.55
85	42,612.13	50,648.49

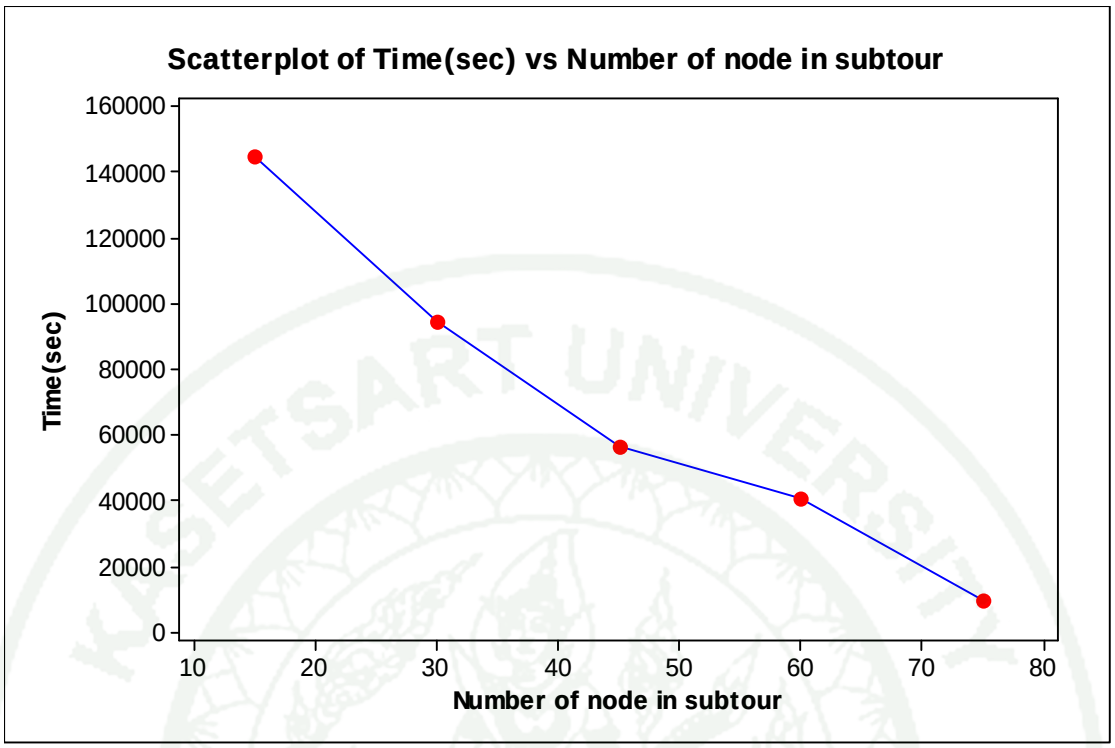


Figure 22 A relations of number of node in subtour and a computational time

Figure 22 show that when number of nodes in subtour increase, a computational time will decrease. So our method of grouping subtour, not only provide a good estimate solution but also reduce a computational time of programming.

CONCLUSION

We apply GPR to TSP with an initial numerical experiment. Our idea is to reduce the size of the problem by initially creating a subtour. Then, we use a small sample of TSP tours to create a GPR function and minimize it to get the solution with the minimum distance. Finally, it is transformed into a TSP tour.

At the first time, test problem in our experiment is only 10 to 25 nodes. After we evaluate test problems, we expand the limit of number of node to 100 nodes. As a result, we select 96 cities of Africa to be one of our test problems. In our numerical experiment, we show that our heuristic performs quite well. In particular, when the sample size R increases, the predicted results are better. Moreover, when the number of nodes in a subtour n_f increases, the distances of our TSP tours are closer to the true optimal ones.

We expect our heuristic to improve if we can find a better method for constructing a subtour and transforming to a TSP tour. One of the major problem in solving TSP by GPR is running time when generate GPR function. As a result, we recommend a Sparse Multiscale Gaussian Process Regression method for reducing the running time in a process of estimate GPR hyperparameter by using randomly sampling data. When apply GPR in TSP, we recommend to reduce 50% of a total number of nodes at the first time and then reduce more than 50% later for saving time to get a best solution. Moreover, we can not use this method if a number of a rest node (a total number of nodes minus a number of nodes in subtour) equal to a number of sampling data.

LITERATURE CITED

Applegate, D.L., R.M. Bixby, V. Chvátal and W.J. Cook. 2006. **The Traveling Salesman Problem. A Computational Study**. Princeton University Press, vol. 1, ISBN 978-0-691-12993-8.

Boyle, P. and M. Frean. 2005. Dependent Gaussian Processes, pp. 217-224. In L. K. Saul, Y. Weiss, and L. Bottou, eds. **Proceedings of the 17** *th* **Neural Information Processing Systems (NIPS)** . The MIT Press.

Blum, A., S. Chawla, D.R. Karger, T. Lane, A. Meyerson, and M. Minkoff. 2007. Approximation Algorithms for Orienteering and Discounted-Reward TSP. **Society for Industrial and Applied Mathematics** 37 : 653-670.

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

Dorigo M. & L.M. Gambardella (1997). Ant Colonies for the Traveling Salesman Problem**, BioSystems** 43: 73-81

Ebden, M. 2008. **Gaussian Process for Regression.** Available source: http://www.robots.ox.ac.uk/~mebden/reports/GPtutorial.pdf. January 30, 2010.

Galton, F. 1886. Regression towards mediocrity in hereditary stature. **Journal of Anthropological Institute** 15: 246-63.

Grasse, P.P. 1959. La reconstruction du nid et les coordinations interindividuelles chez bellicositermes natalensis et cubitermes sp. **La theorie de la stigmergie: essai d'interpretation du comportement des termites constructeurs**, Insectes Sociaux 6 : 41 – 81.

Gutin, G., A. Yeo and A. Zverovich. 2002. Traveling salesman should not be greedy: domination analysis of greedy-type heuristics for TSP. **Discrete Applied Mathematics** 117 : 81–86.

Hall, L.A. 1995. Approximation algorithms for scheduling., pp. 1-45, In Dorit S. Hochbaum. In **Proceedings of Approximation Algorithms for NP-Hard Problems**. PWS Publishing Company.

Hasegawa, M. 2006. Glassy Dynamics in Local Search by Metropolis Algorithm: Temperature-Cycling Experiments On Traveling Salesman Problem . **Intelligent Interaction Technologies** 832: 578-581. University of Tsukuba, Tsukuba.

Hoffman, K. and M. Padberg. 1985. **Traveling Salesman Problem ,Encyclopedia of Operations research**. Springer-Verlag.

Jeong, E.Y. 1997. Application of traveling salesman problem (TSP) for decision of optimal production sequence. **Korean Journal of Chemical Engineering** 14 (5) : 416-421.

Kalaitzis, A.A. 2009. **Image inpainting with Gaussian processes.** Master degree of Science, Thesis, School of Informatics, The University of Edinburgh. Supervised by Christopher K.I. Williams.

Kocijan, J., R. Murray-Smith, C.E. Rasmussen, and B. Likar. 2003. Predictive control with Gaussian process models, pp. 352-356. In **Proceedings of IEEE Region 8 Eurocon 2003: Computer as a Tool**.

Liu, S.B., K.M. Ng, and H.L. Ong. 2007. A new heuristic algorithm for the classical symmetric traveling salesman problem. **International Journal of Computational and Mathematical Sciences** 1(4) : 234-238.,2007

Meeds E. and S. Osindero. 2006. An alternative infinite mixture of Gaussian process experts, pp. 883-890. In Y. Weiss, B. Schölkopf, and J. Platt, editors. In **Proceedings of the 18 *th* Neural Information Processing Systems** (NIPS). The MIT Press, Cambridge, MA.

Melkumyan, A. and F. Ramos. 2009. **Multi-Kernel Gaussian Processes**. Neural Information Processing Systems (NIPS) Workshops, Understanding Multiple Kernel Learning Methods and Understanding Multiple Kernel Learning Methods 4.

Metropolis, N., A. Rosenbluth, A. Rosenbluth, A. Teher, and E. Teher. 1953. "Equations of State Calculations by Fast Computing Machines." **Journal of Chemistry and Physical**. 21: 1087–1092.

Pearson, K. 1896. Mathematical contributions to the theory of evolution. III. Regression, **Heredity and Panmixia. in the Philosophical Transactions of the Royal Society of London** 187: 253–318.

Rasmussen, C.E. and Z. Ghahramani. 2002. Infinite mixtures of Gaussian process experts, pp. 881-888. In T. G. Diettrich, S. Becker, and Z. Ghahramani, editors. In **Proceedings of the 14 *th* Neural Information Processing Systems** (NIPS). The MIT Press.

Rasmussen C. E. and C.K.I. Williams. 2006. **Gaussian Processes for Machine Learning**, MIT Press.

Bixby, B., and Reinelt, G. 1995. Rice Univeristy Software Distribution Center. **TSP problem.** Available source: http://softlib.rice.edu/pub/tsplib/tsp/., January 30,2010,unpublished.

Shah, S. 2009. **Robust Heart Rate Variability Analysis using Gaussian Process Regression**, Master Electrical and Computer, Electrical and Computer Science, The Ohio State University, Ohio,

Sollich, P. and C.K.I. Williams. 2005. Using the Equivalent Kernel to Understand Gaussian Process Regression, pp. 1313–1320. In Saul, L. K., Weiss, Y., and Bottou, L., editors. In **Proceedings of the 17** *th* **Neural Information Processing Systems (NIPS)**. MIT Press.

Varga, A., C. Chira, and D. Dumitrescu. 2009. A Multi-agent Approach To Solving Dynamic Traveling Salesman Problem, pp. 189-197. In **Proceedings of the 1** *st* **International Conference on Bio-Inspired Computational Methods Used for Difficult Problems Solving: Development of Intelligent and Complex Systems. The American Institute of Physics (AIP)** 1117.

Walder, C., K. Williams, and B. Scholkopf. 2008. Sparse Multiscale Gaussian Process Regression, pp. 1112-1119. In **Proceedings of the 25** *th* **International Conference on Machine Learning**. ACM Press, New York.

Williams C.K.I., P.W. Goldberg, and C.M. Bishop. 1998. Regression with Input-dependent Noise:A Gaussian Proces Treatment, pp. 493–499. In M. I. Jordan, M. J. Kearns, and S. A. Solla, editors, In **Proceedings of the 10** *th* **Neural Information Processing Systems**. The MIT Press, Cambridge, MA.

APPENDICES

Appendix A
TSP data of test problem

Appendix Table A1 Data of Ulysses 22, Ulysses16 and Burma14.

	Ulysses 22		**Ulysses 16**		**Burma 14**	
Node	X	Y	X	Y	X	Y
1	38.24	20.42	38.24	20.42	16.47	96.1
2	39.57	26.15	39.57	26.15	16.47	94.44
3	40.56	25.32	40.56	25.32	20.09	92.54
4	36.26	23.12	36.26	23.12	22.39	93.37
5	33.48	10.54	33.48	10.54	25.23	97.24
6	37.56	12.19	37.56	12.19	22	96.05
7	38.42	13.11	38.42	13.11	20.47	97.02
8	37.52	20.44	37.52	20.44	17.2	96.29
9	41.23	9.1	41.23	9.1	16.3	97.38
10	41.17	13.05	41.17	13.05	14.05	98.12
11	36.08	-5.21	36.08	-5.21	16.53	97.38
12	38.47	15.13	38.47	15.13	21.52	95.59
13	38.15	15.35	38.15	15.35	19.41	97.13
14	37.51	15.17	37.51	15.17	20.09	94.55
15	35.49	14.32	35.49	14.32		
16	39.36	19.56	39.36	19.56		
17	38.09	24.36				
18	36.09	23				
19	40.44	13.57				
20	40.33	14.15				
21	40.37	14.23				
22	37.57	22.56				

Appendix Table A2 Data of 96 Africa-Sub problem.

Africa-Subproblem											
Node	X	Y	Node	X	Y	Node	X	Y	Node	X	Y
1	4.55	23.31	25	13.38	25.21	49	6.2	3.24	73	15.25	28.17
2	8.06	15.24	26	15.2	38.53	50	6.2	7.27	74	20.09	28.36
3	2.38	16.54	27	9	38.5	51	0.2	6.44	75	17.5	31.03
4	31.38	8	28	11.36	43.09	52	3.4	8.47	76	15.47	35
5	33.39	7.35	29	8.06	15.57	53	3.52	11.31	77	19.49	34.52
6	34.02	6.51	30	4.4	17.26	54	4.22	18.35	78	25.58	32.35
7	34.05	4.57	31	3.28	16.39	55	0.2	9.27	79	15.57	5.42
8	35.48	5.45	32	1.51	15.35	56	4.16	15.17	80	7.15	12.3
9	35.43	0.43	33	16.46	3.01	57	4.18	15.18	81	22.59	14.31
10	36.4	3.03	34	12.39	8	58	0.04	18.16	82	22.34	17.06
11	22.5	5.3	35	10.23	9.18	59	5.54	22.25	83	26.38	15.1
12	36.2	6.37	36	9.31	13.43	60	0.3	25.12	84	24.45	25.55
13	36.48	10.11	37	8.3	13.15	61	3.23	29.22	85	25.45	28.1
14	34.44	10.46	38	6.18	10.47	62	1.57	30.04	86	26.15	28
15	32.54	13.11	39	5.19	4.02	63	0.19	32.25	87	29.12	26.07
16	32.07	20.04	40	6.41	1.35	64	1.17	36.49	88	29.55	30.56
17	31.12	29.54	41	5.33	0.13	65	2.01	45.2	89	33	27.55
18	31.16	32.18	42	6	1.13	66	4.03	39.4	90	33.58	25.4
19	29.58	32.33	43	6.2	2.37	67	6.1	39.11	91	33.55	18.22
20	30.03	31.15	44	12.22	1.31	68	6.48	39.17	92	23.21	43.4
21	24.05	32.53	45	13.3	2.07	69	8.48	13.14	93	18.55	47.31
22	19.37	37.14	46	12	8.3	70	12.44	15.47	94	12.16	49.17
23	15.36	32.32	47	11.51	13.1	71	11.4	27.28	95	20.1	57.3
24	13.11	30.13	48	12.07	15.03	72	12.49	28.13	96	4.38	55.27

Appendix Table A3 Optimal solution of Ulysses 22, Ulysses16 and Burma14.

Ulysses 22		Ulysses 16		Burma 14	
Rank	Node	Rank	Node	Rank	Node
1	1	1	1	1	1
2	14	2	14	2	2
3	13	3	13	3	14
4	12	4	12	4	3
5	7	5	7	5	4
6	6	6	6	6	5
7	15	7	15	7	6
8	5	8	5	8	12
9	11	9	11	9	7
10	9	10	9	10	13
11	10	11	10	11	8
12	19	12	16	12	11
13	20	13	3	13	9
14	21	14	2	14	10
15	16	15	4		
16	3	16	8		
17	2				
18	17				
19	22				
20	4				
21	18				
22	8				

Appendix Table A4 Optimal solution of 96 Africa-Sub problem.

No.	Node	No.	Node	No.	Node	No.	Node
1	29	25	26	49	74	73	50
2	2	26	28	50	84	74	52
3	3	27	27	51	86	75	53
4	4	28	65	52	85	76	55
5	5	29	96	53	78	77	51
6	6	30	94	54	88	78	49
7	7	31	95	55	87	79	43
8	8	32	93	56	89	80	42
9	9	33	92	57	90	81	41
10	10	34	77	58	91	82	40
11	12	35	76	59	83	83	39
12	13	36	68	60	82	84	44
13	14	37	67	61	81	85	45
14	15	38	66	62	80	86	11
15	16	39	64	63	79	87	33
16	17	40	63	64	70	88	34
17	20	41	62	65	69	89	35
18	18	42	61	66	57	90	38
19	19	43	60	67	56	91	37
20	21	44	59	68	58	92	36
21	25	45	71	69	54	93	32
22	24	46	72	70	48	94	31
23	23	47	73	71	47	95	30
24	22	48	75	72	46	96	1

Appendix Table A5 Distance metric of Ulysses 22

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	-	517.5	493.9	325.1	1,035.4	726.2	638.1	80.1	1,023.0	709.7	2,277.7	462.1	443.3	468.0	623.0	145.2	345.0	330.9	637.9	587.6	583.0	202.1
2	517.5	-	130.9	454.1	1,548.3	1,233.4	1,133.7	546.5	1,453.8	1,123.4	2,768.7	959.7	948.2	981.8	1,136.9	566.3	226.2	475.8	1,075.4	1,026.1	1,019.4	383.4
3	493.9	130.9	-	515.3	1,527.9	1,181.0	1,074.0	540.4	1,364.0	1,033.7	2,696.8	904.3	898.0	939.7	1,115.6	508.9	286.9	536.9	993.3	945.4	938.2	409.2
4	325.1	454.1	515.3	-	1,188.1	982.4	916.8	276.6	1,334.1	1,029.7	2,535.1	747.7	719.6	720.7	797.6	465.6	231.4	21.8	953.4	904.0	900.0	154.0
5	1,035.4	1,548.3	1,527.9	1,188.1	-	477.8	596.2	1,002.2	871.5	883.7	1,466.2	691.8	676.7	613.7	412.4	1,038.2	1,346.8	1,173.9	819.6	826.7	833.6	1,178.4
6	726.2	1,233.4	1,181.0	982.4	477.8	-	125.1	727.5	487.0	408.4	1,556.0	276.8	285.2	262.9	298.8	672.3	1,070.3	975.8	341.9	351.7	358.9	914.0
7	638.1	1,133.7	1,074.0	916.8	596.2	125.1	-	650.4	463.7	306.0	1,640.2	176.1	197.9	207.1	343.2	568.1	982.9	912.7	228.2	230.5	237.3	833.5
8	80.1	546.5	540.4	276.6	1,002.2	727.5	650.4	-	1,057.9	754.0	2,283.4	477.3	452.6	465.0	591.9	218.6	350.3	278.0	676.8	627.5	623.6	187.1
9	1,023.0	1,453.8	1,364.0	1,334.1	871.5	487.0	463.7	1,057.9	-	330.7	1,366.6	599.4	635.1	665.9	783.9	911.0	1,350.8	1,333.9	386.3	437.0	442.4	1,225.1
10	709.7	1,123.4	1,033.7	1,029.7	883.7	408.4	306.0	754.0	330.7	-	1,681.1	349.0	389.4	446.1	641.5	588.0	1,026.9	1,032.0	92.3	131.6	133.4	910.0
11	2,277.7	2,768.7	2,696.8	2,535.1	1,466.2	1,556.0	1,640.2	2,283.4	1,366.6	1,681.1	-	1,816.3	1,834.6	1,818.9	1,760.8	2,202.9	2,622.7	2,527.2	1,706.9	1,753.2	1,760.6	2,469.5
12	462.1	959.7	904.3	747.7	691.8	276.8	176.1	477.3	599.4	349.0	1,816.3	-	40.4	106.9	339.2	396.0	806.8	744.9	256.9	223.4	225.1	658.6
13	443.3	948.2	898.0	719.6	676.7	285.2	197.9	452.6	635.1	389.4	1,834.6	40.4	-	72.9	309.8	389.2	788.3	716.0	297.3	263.6	265.1	636.4
14	468.0	981.8	939.7	720.7	613.7	262.9	207.1	465.0	665.9	446.1	1,818.9	106.9	72.9	-	237.2	434.3	810.1	714.9	354.1	325.9	328.4	651.7
15	623.0	1,136.9	1,115.6	797.6	412.4	298.8	343.2	591.9	783.9	641.5	1,760.8	339.2	309.8	237.2	-	632.0	939.5	785.9	554.6	538.6	542.9	771.8
16	145.2	566.3	508.9	465.6	1,038.2	672.3	568.1	218.6	911.0	588.0	2,202.9	396.0	389.2	434.3	632.0	-	439.8	473.2	525.0	474.4	468.7	328.5
17	345.0	226.2	286.9	231.4	1,346.8	1,070.3	982.9	350.3	1,350.8	1,026.9	2,622.7	806.8	788.3	810.1	939.5	439.8	-	253.1	964.7	914.1	908.4	168.4
18	330.9	475.8	536.9	21.8	1,173.9	975.8	912.7	278.0	1,333.9	1,032.0	2,527.2	744.9	716.0	714.9	785.9	473.2	253.1	-	954.5	905.4	901.6	169.2
19	637.9	1,075.4	993.3	953.4	819.6	341.9	228.2	676.8	386.3	92.3	1,706.9	256.9	297.3	354.1	554.6	525.0	964.7	954.5	-	50.6	56.5	839.7
20	587.6	1,026.1	945.4	904.0	826.7	351.7	230.5	627.5	437.0	131.6	1,753.2	223.4	263.6	325.9	538.6	474.4	914.1	905.4	50.6	-	8.1	789.3
21	583.0	1,019.4	938.2	900.0	833.6	358.9	237.3	623.6	442.4	133.4	1,760.6	225.1	265.1	328.4	542.9	468.7	908.4	901.6	56.5	8.1	-	784.5
22	202.1	383.4	409.2	154.0	1,178.4	914.0	833.5	187.1	1,225.1	910.0	2,469.5	658.6	636.4	651.7	771.8	328.5	168.4	169.2	839.7	789.3	784.5	-

Appendix Table A6 Distance metric of Ulysses 16

node \ node	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	-	517.5	493.9	325.1	1,035.4	726.2	638.1	80.1	1,023.0	709.7	2,277.7	462.1	443.3	468.0	623.0	145.2
2	517.5	-	130.9	454.1	1,548.3	1,233.4	1,133.7	546.5	1,453.8	1,123.4	2,768.7	959.7	948.2	981.8	1,136.9	566.3
3	493.9	130.9	-	515.3	1,527.9	1,181.0	1,074.0	540.4	1,364.0	1,033.7	2,696.8	904.3	898.0	939.7	1,115.6	508.9
4	325.1	454.1	515.3	-	1,188.1	982.4	916.8	276.6	1,334.1	1,029.7	2,535.1	747.7	719.6	720.7	797.6	465.6
5	1,035.4	1,548.3	1,527.9	1,188.1	-	477.8	596.2	1,002.2	871.5	883.7	1,466.2	691.8	676.7	613.7	412.4	1,038.2
6	726.2	1,233.4	1,181.0	982.4	477.8	-	125.1	727.5	487.0	408.4	1,556.0	276.8	285.2	262.9	298.8	672.3
7	638.1	1,133.7	1,074.0	916.8	596.2	125.1	-	650.4	463.7	306.0	1,640.2	176.1	197.9	207.1	343.2	568.1
8	80.1	546.5	540.4	276.6	1,002.2	727.5	650.4	-	1,057.9	754.0	2,283.4	477.3	452.6	465.0	591.9	218.6
9	1,023.0	1,453.8	1,364.0	1,334.1	871.5	487.0	463.7	1,057.9	-	330.7	1,366.6	599.4	635.1	665.9	783.9	911.0
10	709.7	1,123.4	1,033.7	1,029.7	883.7	408.4	306.0	754.0	330.7	-	1,681.1	349.0	389.4	446.1	641.5	588.0
11	2,277.7	2,768.7	2,696.8	2,535.1	1,466.2	1,556.0	1,640.2	2,283.4	1,366.6	1,681.1	-	1,816.3	1,834.6	1,818.9	1,760.8	2,202.9
12	462.1	959.7	904.3	747.7	691.8	276.8	176.1	477.3	599.4	349.0	1,816.3	-	40.4	106.9	339.2	396.0
13	443.3	948.2	898.0	719.6	676.7	285.2	197.9	452.6	635.1	389.4	1,834.6	40.4	-	72.9	309.8	389.2
14	468.0	981.8	939.7	720.7	613.7	262.9	207.1	465.0	665.9	446.1	1,818.9	106.9	72.9	-	237.2	434.3
15	623.0	1,136.9	1,115.6	797.6	412.4	298.8	343.2	591.9	783.9	641.5	1,760.8	339.2	309.8	237.2	-	632.0
16	145.2	566.3	508.9	465.6	1,038.2	672.3	568.1	218.6	911.0	588.0	2,202.9	396.0	389.2	434.3	632.0	-

Appendix Table A7 Distance metric of Burma 14

node \ node	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-	177.1	550.9	718.1	981.7	615.2	455.4	83.7	137.9	345.6	136.7	564.4	344.8	434.7
2	177.1	-	449.9	668.1	1,017.0	638.0	521.6	213.1	314.4	478.0	313.7	574.7	433.6	402.9
3	550.9	449.9	-	270.0	748.0	421.8	469.4	509.5	662.8	896.2	646.5	354.8	486.5	210.0
4	718.1	668.1	270.0	-	504.9	279.4	434.1	653.2	797.5	1,054.6	775.7	248.7	512.4	283.6
5	981.7	1,017.0	748.0	504.9	-	379.2	530.0	898.7	993.5	1,247.1	968.0	445.8	647.6	634.9
6	615.2	638.0	421.8	279.4	379.2	-	197.7	534.6	649.3	911.1	624.3	71.5	309.3	263.4
7	455.4	521.6	469.4	434.1	530.0	197.7	-	371.8	465.5	723.7	440.0	189.0	118.5	261.2
8	83.7	213.1	509.5	653.2	898.7	534.6	371.8	-	153.3	401.5	137.9	486.2	261.4	370.1
9	137.9	314.4	662.8	797.5	993.5	649.3	465.5	153.3	-	262.6	25.6	610.5	347.0	516.9
10	345.6	478.0	896.2	1,054.6	1,247.1	911.1	723.7	401.5	262.6	-	287.1	873.1	605.5	771.7
11	136.7	313.7	646.5	775.7	968.0	624.3	440.0	137.9	25.6	287.1	-	586.2	321.5	496.1
12	564.4	574.7	354.8	248.7	445.8	71.5	189.0	486.2	610.5	873.1	586.2	-	284.4	192.4
13	344.8	433.6	486.5	512.4	647.6	309.3	118.5	261.4	347.0	605.5	321.5	284.4	-	280.5
14	434.7	402.9	210.0	283.6	634.9	263.4	261.2	370.1	516.9	771.7	496.1	192.4	280.5	-

Appendix Table A8 Prediction distance for applied GPR in TSP**.**

Problem	n_f	Distance
Ulysses 22	-	12,592
Ulysses 22	-	11,694
Ulysses 22	-	13,298
Ulysses 22	-	12,991
Ulysses 22	2	13,037
Ulysses 22	2	13,060
Ulysses 22	2	13,240
Ulysses 22	2	12,209
Ulysses 22	6	13,563
Ulysses 22	6	11,929
Ulysses 22	6	11,519
Ulysses 22	6	15,065
Ulysses 22	10	9,505
Ulysses 22	10	11,489
Ulysses 22	10	9,159
Ulysses 22	10	11,426
Ulysses 22	14	10,020
Ulysses 22	14	10,233
Ulysses 22	14	10,555
Ulysses 22	14	9, 223
Ulysses 16	-	10,872
Ulysses 16	-	11,172

Appendix Table A8 Prediction distance for applied GPR in TSP.

Problem	n_f	Distance
Ulysses 16	-	10,424
Ulysses 16	-	9,417
Ulysses 16	3	10,704
Ulysses 16	3	9,323
Ulysses 16	3	8,380
Ulysses 16	3	8,702
Ulysses 16	6	8,440
Ulysses 16	6	8,456
Ulysses 16	6	6,899
Ulysses 16	6	8,673
Ulysses 16	8	8,753
Ulysses 16	8	8,817
Ulysses 16	8	10,283
Ulysses 16	8	10,283
Burma 14	-	4,717
Burma 14	-	4,717
Burma 14	-	6,587
Burma 14	-	3,886
Burma 14	2	3,881
Burma 14	2	4,860
Burma 14	2	4,215
Burma 14	2	4,942

Appendix Table A8 (Continued)

Problem	n_f	Distance
Burma 14	4	4,578
Burma 14	4	4,872
Burma 14	4	4,290
Burma 14	4	4,743
Burma 14	6	3,961
Burma 14	6	4,279
Burma 14	6	3,919
Burma 14	6	4,170

Appendix B
Programming

Appendix B1 Generate Route and distance function

```
function [total_dist,pop] = Gen_routhfixstart
(Number_Node,Dis_matrix,N_Random,N_iter)

Nod = Number_Node;
[nr,nc] = size(Dis_matrix);
if Nod ~= nr || Nod ~= nc
   error('Invalid XY or DMAT inputs!')
end
n = Nod - 1;
pop_size = N_Random;
num_iter = max(1,round(real(N_iter)));
% Initialize the Population
pop = zeros(pop_size,n);
for k = 1:pop_size
   pop(k,:) = randperm(n)+1;
end
 global_min = Inf;
total_dist = zeros(1,pop_size);
dist_history = zeros(1,num_iter);
tmp_pop = zeros(4,n);
new_pop = zeros(pop_size,n);
dmat=Dis_matrix;
for iter = 1:num_iter
      for p = 1:pop_size
       d = dmat(1,pop(p,1));
       for k = 2:n
          d = d + dmat(pop(p,k-1),pop(p,k));
       end
       total_dist(p) = d;
   end
```

```
   total_dist=total_dist';
end

_ _ _ _ _ — _ _ _ _ _ — _ _ _ _ _ — _ _ _ _ _

function [total_dist,pop,kk] = Gen_routhfixstartSearch
(Number_Node,number_fixnode,Dis_matrix,N_Random,N_iter,startnode,Endnode,p
ossibleNodeRandom,fixRoutemetric)

Nod=Number_Node;
n = Number_Node-number_fixnode;
[nr,nc] = size(Dis_matrix);
if Nod ~= nr || Nod ~= nc
   error('Invalid XY or DMAT inputs!')
end
pop_size = 4*ceil(N_Random/4);
num_iter = max(1,round(real(N_iter)));
pop = zeros(pop_size,n);
kk = zeros(pop_size,n);
for k = 1:pop_size
   kk(k,:)=randperm(n);
end
for k = 1:pop_size
   for j=1:n
   ff=kk(k,j);
   pop(k,j)=possibleNodeRandom(1,ff);
   end
end
global_min = Inf;
total_dist = zeros(1,pop_size);
dist_history = zeros(1,num_iter);
tmp_pop = zeros(4,n);
```

```
new_pop = zeros(pop_size,n);
dmat=Dis_matrix;
    pp=number_fixnode-1;
    fixdis =zeros(1,1);
      for k = 1:pp
         fixdis = fixdis+ dmat(fixRoutemetric(1,k),fixRoutemetric(1,k+1));
      end
for iter = 1:num_iter
   for p = 1:pop_size
      d = dmat(startnode,pop(p,1))+dmat(pop(p,n),Endnode); % Add Start Distance
and end distance
      for k = 2:n
         d = d + dmat(pop(p,k-1),pop(p,k));
      end
      total_dist(p) = d+fixdis;
   end
   total_dist=total_dist';
end
end
```

Appendix B2 Search subroute function

```
function [X,Y] = SearchNodeMin (Number_Node,Dis_matrix,N_SearchNodeFix)

X=zeros(N_SearchNodeFix,2);
P=max(max(Dis_matrix));
Y=zeros(N_SearchNodeFix+1,1);
for i = 1:Number_Node
   for j = 1:Number_Node
 if isequal(i,j)==0;
        if Dis_matrix(i,j)<P
          P= Dis_matrix(i,j);
```

```
            Y(1)=i;
          X(1,1)=i;
          X(1,2)=j;
          end
  end
    end
end
for u=2:N_SearchNodeFix
  for i = 1:Number_Node
 if i==X(u-1,2)
    P=max(max(Dis_matrix));
    for j = 1:Number_Node
       if isequal(i,j)==0
          if isequal(j,X(1,1))==0
      s=0;
        for y=1:(u-1)
         if
or(and(isequal(i,X(y,1)),isequal(j,X(y,2))),and(isequal(j,X(y,1)),isequal(i,X(y,2))))==
1
            s=s+1;
          end
          if isequal(j,Y(y))==1
            s=s+1;
          end
         end
         if s==0
          if or(Dis_matrix(i,j)<P,Dis_matrix(i,j)==P)
            P= Dis_matrix(i,j);
            Y(u)=i;
          X(u,1)=i;
          X(u,2)=j;
          end
```

```
        end
          end
       end
   end
 end
  end
end
pp=X(N_SearchNodeFix,2);
Y(N_SearchNodeFix+1)=pp;
 Y=Y';
 end
```

Appendix B3 Transform to Binary code

```
[Rdsize,n]=size(RoutX);
N=n+1;
Num_var=0;
tt=0;
BeforeP=0;
AfterP=0;
Count=0;
for I=1:(N-1)
Num_var= N-I+Num_var;
end
CharacterX=zeros(Rdsize,Num_var);
   for J=1:Rdsize
      p=0;
      p=RoutX(J,1);
  CharacterX(J,p-1)=1;
pp=RoutX(J,n);
CharacterX(J,pp)=1;
 for I=2:n
```

```
   BeforeP= RoutX(J,I-1);
   AfterP= RoutX(J,I);
   if BeforeP>AfterP
      Add=BeforeP-AfterP;
      Count=0;
      for f=1:AfterP-1
         Count=N-f+Count;
      end
       Add=Count+Add;
       CharacterX(J,Add)=1;
   end
   if BeforeP<AfterP
      Add=AfterP-BeforeP;
      Count=0;
      for f=1:BeforeP-1
         Count=N-f+Count;
      end
       Add=Count+Add;

    CharacterX(J,Add)=1;
   end
 end
   end
end
```

– – – – – — – – – – — – – – – — – – – – –

```
function CharacterX = EditXSearch(RoutX)

 [Rdsize,n]=size(RoutX);
N=n;
Num_var=0;
%Generate X data
```

```
for I=1:(N-1)
Num_var= N-I+Num_var;
end
CharacterX=zeros(Rdsize,Num_var);
   for J=1:Rdsize
      p=RoutX(J,1);
 CharacterX(J,p)=1;
pp=RoutX(J,n);
CharacterX(J,pp)=1;
 for I=2:n
   BeforeP= RoutX(J,I-1);
   AfterP= RoutX(J,I);
   if BeforeP>AfterP
      Add=BeforeP-AfterP;
      Count=0;
      for f=1:AfterP-1
         Count=n-f+Count;
      end
       Add=Count+Add;
       CharacterX(J,Add)=1;
   end
   if BeforeP<AfterP
      Add=AfterP-BeforeP;
      Count=0;
      for f=1:BeforeP-1
         Count=n-f+Count;
      end
       Add=Count+Add;
    CharacterX(J,Add)=1;
   end
 end
   end
```

```
end
```

Appendix B4 Transform to TSP route

```
function [X,XX,CharacterX] = GetrouteX(RoutX,node,routenotfit)

[Rdsize,kk]=size(RoutX);
N=node;
Num_var=0;
X=zeros(2,N-1);
for I=1:(N-1)
Num_var= N-I+Num_var;
end
if kk ~= Num_var
  error('Error: Number of x do not agree with number of nodes')
end
CharacterX=zeros(2,Num_var);
XX=zeros(2,Num_var);
xx=0;
for I=1:(N-1)
    count=node-I;
   K=0;
 for J=I:(N-1)
     K=K+1;
    count=count-1;
    if I==1
CharacterX(1,K)=node-count;
CharacterX(2,K)=I;
    else
 CharacterX(1,xx+K)=node-count;
 CharacterX(2,xx+K)=I;
    end
```

```
end
xx=xx+N-I;
end
for I=1:Rdsize
 for J=1:Num_var
    p=RoutX(I,J);
    if p>0
   XX(1,J)= CharacterX(1,J);
   XX(2,J)= CharacterX(2,J);
    end
 end
   end
count=0;
 for J=1:Num_var
    p=XX(1,J);
    if p>0
       count=count+1;
  X(1,count)= XX(1,J);
   X(2,count)= XX(2,J);
    end
 end
binary
 if nargin == 3
     [nr,cr]=size(X);
     for i=1:nr
        for j=1:cr
          PL=routenotfit(1, X(i,j));
          X(i,j)=PL;
        end
     end
  end
```

```
end
_ _ _ _ _ — _ _ _ _ _ — _ _ _ _ _ — _ _ _ _ _

function [X,Y] = Fitnode(RouteX,node,startnode,distancemetric)

X=zeros(node,1);
before=startnode;
Ex=RouteX;
for i=1:node
nds=find(RouteX==before);
[row3 col3]=ind2sub(size(RouteX),nds);
[row,col] = Find_smalllestdistance(RouteX,row3,col3,distancemetric,X,before);
if row==1
    next=RouteX(row+1,col);
    X(i,1)=before;
    X(i+1,1)=next;
   before=next;
   Ex(1,col)=0;
   Ex(2,col)=0;
elseif row==2
    next=RouteX(1,col);
    X(i,1)=before;
   X(i+1,1)=next;
   before=next;
   Ex(1,col)=0;
   Ex(2,col)=0;
else
   X(i+1,1)=0;
   before=0;
end
end
```

```
 [nRR,cRR]=size(X);
  count=0;
  for i=1:nRR
     if(X(i,1)>0)
        count=count+1;
     end
  end
  if count>0
  c=0;
  ccc=zeros(count,1);
  [nRR,cRR]=size(X);
  for i=1:nRR
     if X(i,1)>0
       c=c+1;
       ccc(c,1)=X(i,1);
     end
  end
  X=ccc;
  end
  [n,c]=size(Ex);
Count=0;
for i=1:c
if Ex(1,i)>0
   Count=Count+1;
end
end
nn=Count;
Y=zeros(2,nn);
Count=0;
for i=1 :c
if Ex(1,i)>0
   Count=Count+1;
```

```
   Y(Count,1)=Ex(1,i);
   Y(Count,2)=Ex(2,i);
end
end
Y=Y';
end
_ _ _ _ _ — _ _ _ _ _ — _ _ _ _ — _ _ _ _ _

function [RR] = Fit(main,node,Other,d,j)
b
if nargin == 4
[Notmain] = notmain(main,node);
elseif nargin == 5
[Notmain] = notmain(main,node,j);
end
if Notmain==0
   RR=main;
else
 [Other] = CheckMainforOther(main,Notmain,Other);
 [n,c]=size(main);
 if  Other==0
   [RR,freenode] = connectfreenode(main(n,1),Notmain,d);
 else
   [RR,freenode,Otherroute] = connectfreenode(main(n,1),Notmain,d,Other);
 end
 [nRR,cRR]=size(RR);
 count=0;
 for i=1:nRR
    if(RR(i,1)>0)
       count=count+1;
    end
 end
```

```
 if count>0
 c=0;
 ccc=zeros(count,1);
 [nRR,cRR]=size(RR);
 for i=1:nRR
    if RR(i,1)>0
      c=c+1;
      ccc(c,1)=RR(i,1);
    end
 end
 RR=ccc;
 end
end
end
```

Appendix B5 Subfunction of Transform to TSP route

```
function [rowf,colf,nextx,dis1,z] =
Find_smalllestdistance(RouteX,row,col,disstancemetric,X,before)

if isempty(row)==0
 [n,c]=size(row);
 nextx=zeros(n,1);
 dis1=zeros(n,1);
for i=1:n
   p=row(i,1);
 k=col(i,1);
if p==1
   nextx(i,1)=RouteX(p+1,k);
elseif p==2
   nextx(i,1)=RouteX(p-1,k);
end
```

```
dis1(i,1)=disstancemetric(before,nextx(i,1));
 end
 for i=1:n
 nds=find(X==nextx(i,1));
 [rownex colnex]=ind2sub(size(X),nds);
if isempty(rownex)==0
   nextx(i,1)=0;
end
 end
   disMax=max(max(disstancemetric));
   [n,c]=size(row);
   rowf=0;
  colf=0;
for i=1:n
    if nextx(i,1)~=0
  if or(dis1(i,1)< disMax,dis1(i,1)== disMax)
     disMax=dis1(i,1);
          rowf=row(i,1);
          colf=col(i,1);
z=1;
  end
    end
end
elseif isempty(row)==1
   rowf=0;
  colf=0;
end
end
```

– – – – – — – – – – — – – – – — – – – – –

```
function [freenode] = freenodecheck(freenode,Otherroute)
```

```
[nO,cO]=size(Otherroute);
count=0;
[n,c]=size(freenode);
for i=1:n
   before=freenode(i,1);
   nds=find(Otherroute==before);
   [row3 col3]=ind2sub(size(Otherroute),nds);
   if nds~=0
     freenode(i,1)=0;
     count=count+1;
   end
end
c=count;
if c~=0
count=0;
 freenode2=zeros(c,1);
[n,c]=size(freenode);
for i=1:n
   if (freenode(i,1)~=0)
     count=count+1;
     freenode2(count,1)=freenode(i,1);
   end
end
freenode=freenode2;
end
end
```

– – – – – — – – – – — – – – – – — – – – – –

```
function [Notmain] = notmain(main,node,j)

if nargin == 2
   X=zeros(node,1);
```

```
for i=1:node
   X(i,1)=i;
end
elseif nargin == 3
X=j';
[oo,node]=size(j);
end
 count=0;
for i=1:node
   before=X(i,1);
   nds=find(main==before,1);
   if (isempty(nds)==0)
     X(i,1)=0;
     count=count+1;
   end
end
c=node-count;
if c~=0
count=0;
Notmain=zeros(c,1);
for i=1:node
   if X(i,1)~=0
     count=count+1;
     Notmain(count,1)=X(i,1);
   end
end
else
Notmain=0;
end
end
```

– – – – – — – – – – — – – – – – — – – – – –

```
function [Otherroute] = other(freenode,d,Otherroute)

      [nf,cf]=size(freenode);
   [nO,cO]=size(Otherroute);
   Otherroute2=Otherroute;
count=0;
  for i=1:nO
     for j=1:cO
   before=Otherroute2(i,j);
   nds=find(Otherroute==before);
   [row3 col3]=ind2sub(size(Otherroute),nds);
   [nR,cR]=size(row3);
   if nds~=0
      dis1=max(max(d));
    for z=1:nR
       if Otherroute2(row3(z),col3(z))~=0
       if and(row3(z),col3(z))~=0
       if row3(z)==1
     dis=d(Otherroute2(row3(z),col3(z)),Otherroute2(row3(z)+1,col3(z)));
       elseif row3(z)==2
     dis=d(Otherroute2(row3(z),col3(z)),Otherroute2(row3(z)-1,col3(z)));
       end
       if dis<=dis1
          rZ=row3(z);
          cZ=col3(z);
          dis1=dis;
       end
       end
       end
    end
        for z=1:nR
       if col3(z)~= cZ
```

```
 Otherroute(1,col3(z))=0;
 Otherroute(2,col3(z))=0;
      end
       end
        end
   end
     end
  count=0;
  [nO,cO]=size(Otherroute);
  for i=1:cO
     if Otherroute(1,i)>0
        count=count+1;
     end
  end
  if count>0
   Otherroute2=zeros(2,count);
   c=0;
   for i=1:cO
      if Otherroute(1,i)~=0
         c=c+1;
        Otherroute2(1,c)= Otherroute(1,i);
        Otherroute2(2,c)= Otherroute(2,i);
      end
   end
   Otherroute=Otherroute2;
  end
  end
```

– – – – – — – – – – — – – – – — – – – – –

```
function [Other] = CheckMainforOther(main,Notmain,Other)

 [n,c]=size(Notmain);
K=Other;
```

```
count=0;
[nOther,cOther]=size(Other);
newOther=zeros(nOther,cOther);
for i=1:n
   before=Notmain(i,1);
   nds=find(Other==before);
   [rowX colX]=ind2sub(size(Other),nds);
   [nm,cm]=size(rowX);
   for h=1:nm
   if (nds~=0)
    newOther(rowX(h),colX(h))=Other(rowX(h),colX(h));
   end
   end
end
for i=1:cOther
   for j=1:nOther
   if newOther(j,i)==0
     if j==1
      newOther(2,i)=0;
     elseif j==2
       newOther(1,i)=0 ;
     end
   end
   end
end
[n,c]=size(main);
count=0;
for i=1:cOther
   if newOther(1,i)~=0
      count=count+1;
   end
end
```

```
if count==0
   Other=0;
else
K=zeros(2,count);
count=0;
for i=1:cOther
   if newOther(1,i)~=0
      count=count+1;
      K(1,count)=newOther(1,i);
      K(2,count)=newOther(2,i);
   end
end
Other=K;
end
end
```

Appendix B6 Create distance matrix

```
function distances=dis2(x,y)
N_cities = size(x,1);
distances = zeros(N_cities,N_cities);
for i = 1:N_cities
  for j = (i+1):N_cities
     distances(j,i) = pos2dist(x(i),y(i),x(j),y(j),2);
    distances(i,j)=distances(j,i);
  end
end
end
```

– – – – – — – – – – — – – – – — – – – – –

```
function dist = pos2dist(lag1,lon1,lag2,lon2,method)
```

```
if nargin < 4
   dist = -99999;
   disp('Number of input arguments error! distance = -99999');
   return;
end
if abs(lag1)>90 || abs(lag2)>90 || abs(lon1)>360 || abs(lon2)>360
   dist = -99999;
   disp('Degree(s) illegal! distance = -99999');
   return;
end
if lon1 < 0
   lon1 = lon1 + 360;
end
if lon2 < 0
   lon2 = lon2 + 360;
end
if nargin == 4
   method = 1;
end
if method == 1
   km_per_deg_la = 111.3237;
   km_per_deg_lo = 111.1350;
   km_la = km_per_deg_la * (lag1-lag2);
   if abs(lon1-lon2) > 180
      dif_lo = abs(lon1-lon2)-180;
   else
      dif_lo = abs(lon1-lon2);
   end
   km_lo = km_per_deg_lo * dif_lo * cos((lag1+lag2)*pi/360);
   dist = sqrt(km_la^2 + km_lo^2);
else
   R_aver = 6374;
```

```
    deg2rad = pi/180;
    lag1 = lag1 * deg2rad;
    lon1 = lon1 * deg2rad;
    lag2 = lag2 * deg2rad;
    lon2 = lon2 * deg2rad;
    dist = R_aver * acos(cos(lag1)*cos(lag2)*cos(lon1-lon2) + sin(lag1)*sin(lag2));
end
```

Appendix B7 Use GPML to create hyperparameter and minimize function

```
[n,D]=size(x);
logtheta0 = zeros(D+2,1);
logtheta0(D+2) = -1.15;
covfunc = {'covSum', {'covSEard','covNoise'}};
logtheta = minimize(logtheta0, 'gpr', -100, covfunc, x, y);
[X0, fX1, i0] = minimize2(x0,-100,logtheta, covfunc, x, y);
```

Appendix C
GPML Toolbox manual

Appendix C1 Description of the GPR function gpr.m on GPML toolbox

The basic computations needed for standard Gaussian process regression (GPR) are straight forward to implement in MATLAB. Several implementations are possible, here we present an implementation closely resembling Algorithm in C1.

Input : X (input), y (target), k (covariance function), σ_n^2 (noise level), x_* (test input)

$L = cholesky(K + \sigma_n^2 I)$

$\alpha = \dfrac{L^T}{\left(\dfrac{L}{y}\right)}$

$\overline{y_*} = K_*^T \alpha$

Predictive variance is on Equations

$\upsilon = \dfrac{L}{K_*}$

$V\left[y_*\right] = k(x_*, x_*) - \upsilon^T \upsilon$

Log marginal likelihood is on equation

$$\log p(y \mid X) = -\frac{1}{2} y^T \alpha - \sum_i \log L_{ii} - \frac{n}{2} \log 2\pi$$

Return: $\overline{y_*}$ (mean), $V\left[y_*\right]$ (variance), $\log p(y \mid X)$ (log marginal likelihood)

Appendix Figure C1 GPR algorithm

with three exceptions: Firstly, the predictive variance returned is the variance for noisy test-cases, whereas C1 Algorithm gives the variance for the noise-free latent function; conversion between the two variances is done by simply adding (or

subtracting) the noise variance. Secondly, the *negative* log marginal likelihood is returned, and thirdly the partial derivatives of the negative log marginal likelihood.

A simple implementation of a Gaussian process for regression is provided by the gpr.m program (which can conveniently be used together with minimize.m for optimization of the hyperparameters). The program can do one of two things:

- compute the negative log marginal likelihood and its partial derivatives the hyperparameters, usage

  [nlml dnlml] = gpr(logtheta, covfunc, x, y)

  which is used when "training" the hyperparameters, or

- compute the (marginal) predictive distribution of test inputs, usage

  [mu S2] = gpr(logtheta, covfunc, x, y, xstar)

Selection between the two modes is indicated by the presence (or absence) of test cases, xstar. The arguments to the gpr.m function are:

Appendix Table C1 Input and output of GPR function

Inputs	
logtheta	a (column) vector containing the logarithm of the hyperparameters
covfunc	the covariance function
X	a n by D matrix of training inputs
Y	a (column) vector if training set targets (of length n)
Xstar	a nn by D matrix of test inputs
Outputs	
Nlml	the negative log marginal likelihood
Dnlml	Column vector with the partial derivatives of the negative log marginal likelihood (the logarithm of the hyperparameters).
Mu	Column of predictive means
S2	Column vector of predictive variances

The covfunc argument specifies the function to be called to evaluate covariances. The covfunc can be specified either as a string naming the function to be used or as a cell array. A number of covariance functions are provided, see covFunctions.m for a more complete description. A commonly used covariance function is the sum of a squared exponential (SE) contribution and independent noise. This can be specified as:

```
covfunc = {'covSum', {'covSEiso','covNoise'}};
```

where covSum merely does some bookkeeping and calls the squared exponential (SE) covariance function covSEiso.m and the independent noise covariance covNoise.m. The squared exponential (SE) covariance function (also called the radial basis function (RBF) or Gaussian covariance function) is given by in equation below :

$$k_y(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2l^2}(x_p - x_q)^2 + \sigma_n^2 \delta_{pq}$$

for the scalar input case, and equation

$$k_* = \sigma_f^2 \exp(-\frac{1}{2}(x_* - x_i)^T M (x_* - x_i)) + \sigma_n^2 \delta_{*,i},$$

for multivariate inputs.

Appendix D
Numerical Example

Appendix D Numerical Example

We have 5 nodes of Geographic TSP thus 5 nodes have symmetric distance properties. We need to predict the minimum distance of this TSP problem and find TSP route by using GPR method which has 2 nodes in subtour. Latitudes and longitudes of Geographic TSP are shown in Appendix FigureD1:

Node	Latitude	Longitude
1	38.24	20.42
2	39.57	26.15
3	40.56	25.32
4	36.26	23.12
5	33.48	10.54

Appendix Figure D1 Latitude and longitude from GEO TSP.

Appendix D1 Calculate distance matrix

	1	2	3	4	5
1	0	517	493	325	1035
2	517	0	130	454	1548
3	493	130	0	515	1527
4	325	454	515	0	1188
5	1035	1548	1527	1188	0

Appendix Figure D2 Distance matrix

Appendix D2 Construct a subtour by a greedy heuristic

The minimum distance is 130 Km. from path i = 2 and j = 3 which shown in Appendix FigureD3, subtour is shown in Appendix Figure D4 and the rest tour are shown in Appendix FigureD5.

node / node	1	2	3	4	5
1	0	517	493	325	1035
2	517	0	130	454	1548
3	493	130	0	515	1527
4	325	454	515	0	1188
5	1035	1548	1527	1188	0

Appendix Figure D3 Distance matrix which shown minimum distance.

2
3

Appendix Figure D4 Subtour

1
4
5

Appendix Figure D5 The rest tour

Appendix D3 Generate sampling tour and their corresponding total distance

We generate 3 sampling tours (X) in Appendix Figure D6 and total distance (Y) in Appendix Figure D7. After that we redefine the rest node in ordinary number which shows in Appendix Figure D8 and sampling tour are redefined base on the rest tour in Appendix Figure D8 (Appendix Figure D9).

Tour No.	Tour		
1	1	4	5
2	4	5	1
3	5	1	4

Appendix Figure D6 3 sampling tours(X)

Tour No.	Distance
1	3684
2	3385
3	3471

Appendix Figure D7 Their corresponding total distance (Y)

Current	New
1	1
4	2
5	3

Appendix Figure D8 The rest tour are redefined

Tour No.	Tour		
1	1	2	3
2	2	3	1
3	3	1	2

Appendix Figure D9 Redefined sampling tour

Each of redefined sampling tours is re-writing in binary matrix that shown in Appendix Figure D10. The value in binary matrix is 1 if the path from node i to node j appear and 0 if the path from node i to node j do not appear. Because of geographic TSP are symmetric so the lower triangular of a binary matrix can be deleted from Appendix Figure D10(Appendix Figure D11). The tours are transformed to 1 row matrix (X') which shown in Appendix Figure D12.

Tour No.1				Tour No. 2				Tour No. 3			
node / node	1	2	3	node / node	1	2	3	node / node	1	2	3
1	0	1	0	1	0	0	1	1	0	1	1
2	1	0	1	2	0	0	1	2	1	0	0
3	0	1	0	3	1	1	0	3	1	0	0

Appendix Figure D10 Binary term of sampling tour

Tour No.1				Tour No. 2				Tour No. 3			
node / node	1	2	3	node / node	1	2	3	node / node	1	2	3
1	-	1	0	1	-	0	1	1	-	1	1
2	-	-	1	2	-	-	1	2	-	-	0
3	-	-	-	3	-	-	-	3	-	-	-

Appendix Figure D11 Binary matrix after reducing variable

Tour No.	Tour		
1	1	0	1
2	0	1	1
3	1	1	0

Appendix Figure D12 One row matrix of each tour (X')

Appendix D4 Use GPR function to approximate an optimal TSP tour.

We determine a starting solution, which is a TSP tour. In our experiment starting solution is TSP tour which has lowest total distance.

Tour No.	Tour			Distance
1	1	0	1	3684
2	0	1	1	3385
3	1	1	0	3471

Appendix Figure D13 Selected starting solution

Parameters are estimated to creating GPR function which is used for predicted the optimal value of TSP problem. The prediction value are shown in Appendix Figure D14

Tour			Distance
1	1	0	3570

Appendix Figure D14 Prediction value

Appendix D5 Transform an optimal solution to a TSP tour.

A one row route is transformed to binary matrix (Appendix Figure D15). It is transformed again to a path matrix $[2\times n_{path}]$ (Appendix Figure D16) and select a lowest distance path in a path matrix.

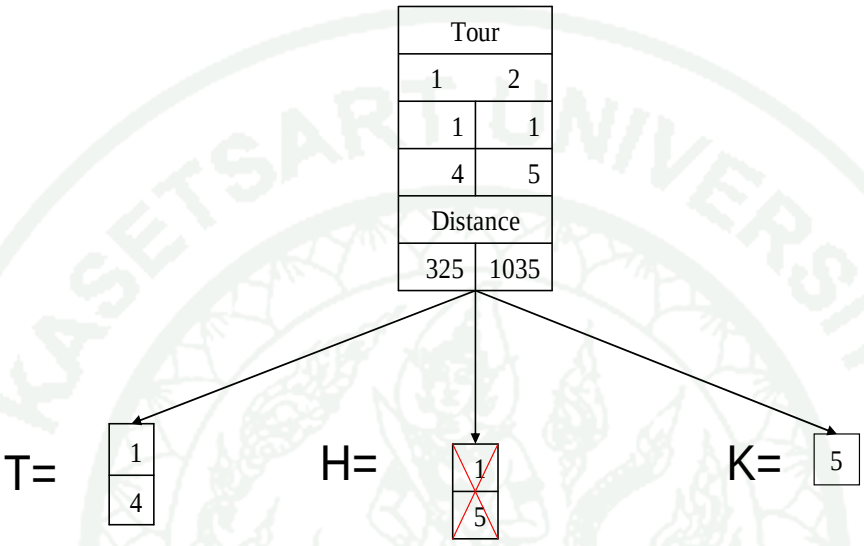
node / node	1	4	5
1	-	1	1
4	-	-	0
5	-	-	-

Appendix Figure D15 Binary Matrix of optimal solution

Tour	
1	**2**
1	1
4	5
Distance	
325	1035

Appendix Figure D16 Path matrix

We select the lowest distance tour which is tour 2, but tour 2 can not connect which tour 1 because the end of tour 2 is 3. We can separate T, H, and K matrixes in Appendix Figure D17 which H matrix is deleted



Appendix Figure D17 Delete paths in H which have node the same as T.

From matrix T, K an H, we will get TSP tour below in D18.

Tour					Distance
2	3	1	4	5	3684

Appendix Figure D18 TSP Tour

The optimal solution from GPR prediction is lower than a solution when transform to a TSP tour because of a tour construction in the last step.

CURRICULUM VITAE

NAME : Ms. Jarumas Chanatapanich

BIRTH DATE : October 17, 1985

BIRTH PLACE : Bangkok, Thailand

EDUCATION

YEAR	**INSTITUTE**	**DEGREE/DEPLOMA**
2008	Kasetsart Univ.	B.E.(Engineering)
2011	Kasetsart Univ.	M.E.(Engineering)

POSITION/TITLE :Buyer

WORK PLACE :Volvo Truck and Bus (Thailand) group Co.,Ltd