# A TRAVELING SALESMAN PROBLEM APPLICATION WITH CONSTRUCTION HEURISTICS

**VILASINEE LEOWARIN**

**A THEMATIC PAPER SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF BUSINESS ADMINISTRATION (BUSINESS MODELING AND ANALYSIS) FACULTY OF GRADUATE STUDIES MAHIDOL UNIVERSITY 2010**

Thematic Paper
entitled

# A TRAVELING SALESMAN PROBLEM APPLICATION WITH CONSTRUCTION HEURISTICS

……..……..……
Ms. Vilasinee Leowarin
Candidate

……..……..……
Lect. Ornlatcha Sivarak, Ph.D.
Major advisor

……..……..……
Lect. Prapoj Srinuwattiwong, Ph.D.
Co-advisor

……..……..……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……..……..……
Asst. Prof. Yingyot Chiaravutthi, Ph.D.
Program Director
Master of Business Administration
Program in Business Modeling and Analysis
International College
Mahidol University

Thematic Paper
entitled

# A TRAVELING SALESMAN PROBLEM APPLICATION WITH CONSTRUCTION HEURISTICS

was submitted to the Faculty of Graduate Studies, Mahidol University
for the degree of Master of Business Administration (Business Modeling and Analysis)
on
November 27, 2010

……
Ms. Vilasinee Leowarin
Candidate

……
Lect. Athapol Ruangkanjanases, Ph.D.
Chair

……
Lect. Prapoj Srinuwattiwong, Ph.D.
Member

……
Lect. Ornlatcha Sivarak, Ph.D.
Member

……
Prof. Banchong Mahaisavariya,
M.D., Dip Thai Board of Orthopedics
Dean
Faculty of Graduate Studies
Mahidol University

……
Assoc. Prof. Rassmidara Hoonsawat, Ph.D.
Dean
International College
Mahidol University

# ACKNOWLEDGEMENTS

The completion of this thematic paper can be succeeded by the support from many people. I express deepest gratitude to my advisor and co-advisor, Dr.Ornlatcha Sivaruk and Dr.Prapoj Srinuwattiwong, respectively who offered valuable support and guidance, as well as Dr.Athapol Ruangkanjanases, Chair of the committee. My sincere gratitude is also due to Dr.Kritsakorn Luangvilai for the overall programming advice and support. Special thanks to Dr.Wuthichai Wongthatsanekorn for the Visual Basic programming consultation.

I would like to thank the lecturers of the program who give me opportunities to learn as well as my friends and classmates for our sharing, discussion, and fruitful friendship throughout the duration of my studies. Not forgetting to my best friend, Nim, for her support and encouragement. I wish to express my honest gratitude to my beloved mother for her understanding and endless love.

Any shortcoming in any aspect of this thematic paper is solely my own responsibility.

Vilasinee Leowarin

A TRAVELING SALESMAN PROBLEM APPLICATION
WITH CONSTRUCTION HEURISTICS

VILASINEE LEOWARIN 5138402 ICMA/M

M.B.A. (BUSINESS MODELING AND ANALYSIS)

THEMATIC ADVISORY COMMITTEE : ATHAPOL RUANGKANJANASES, Ph.D., ORNLATCHA SIVARUK, Ph.D., PRAPOJ SRINUWATTIWONG, Ph.D.

ABSTRACT

The Traveling Salesman Problem (TSP) is an optimization problem to find the shortest tour starting from one location, visiting all locations exactly once, and returning back. Three construction heuristics – namely Nearest Neighbor (NN), Nearest Insertion (NI), and Farthest Insertion (FI) – were implemented and compared in terms of solution quality and computational time. The empirical test was conducted using two data sets, TSP Library and instances of real locations of selected department stores/ hypermarkets in Thailand. The results show that FI algorithms gave the best solution on average, with less than 10% deviation from the optimal solution for TSPs with up to 50 locations. The best solution among NI and FI algorithms was on par with the optimal solution for up to 10-location TSPs. For solving the same size of TSP, Excel Solver required more time and became inefficient for more than 10 locations. Thus, NI and FI heuristics can be applied, instead of exact methods, for small TSPs with an acceptable tolerance. This study also describes how to use Google Maps to locate locations and the Great Circle formula for estimating the distance from latitude-longitude coordinates. The methods from this study can be applied by individuals, salespersons, and businesses to solve a small TSP where optimization software is not available.

KEY WORDS: TRAVELING SALESMAN PROBLEM / SHORTEST ROUTE / CONSTRUCTION HEURISTICS / GOOGLE MAPS

60 pages

การประยุกต์ปัญหาเส้นทางเดินของพนักงานขายด้วยวิธีคอนสตรัคชั่นฮิวริสติกส์

A TRAVELING SALESMAN PROBLEM APPLICATION WITH CONSTRUCTION HEURISTICS

วิลาสินี เลี้ยววาริณ 5138402 ICMA/M

บธ.ม. (การวิเคราะห์และการสร้างตัวแบบธุรกิจ)

คณะกรรมการที่ปรึกษาสารนิพนธ์: อัตถพล เรืองกาญจนเศรษฐ์, Ph.D., อรลัชชา ศิวรักษ์, Ph.D., ประพจน์ ศรีนุวัตติวงศ์, Ph.D.

บทคัดย่อ

ปัญหาการเดินทางของพนักงานขาย (Traveling Salesman Problem; TSP) คือ ปัญหาการหารอบระยะทางสั้นที่สุดในการเดินทางจากสถานที่เริ่มต้นไปยังสถานที่ต่างๆ แห่งละครั้งและกลับมายังจุดเริ่มต้น การศึกษานี้ได้ประยุกต์ใช้คอนสตรัคชั่นฮิวริสติกส์สามประเภท ได้แก่ วิธีเลือกไปยังจุดใกล้สุด (Nearest Neighbor; NN) วิธีแทรกจุดใกล้สุด (Nearest Insertion; NI) และวิธีแทรกจุดไกลสุด (Farthest Insertion; FI) โดยเปรียบเทียบคำตอบและระยะเวลาที่ใช้ในการคำนวณ จากการทดสอบโดยใช้ข้อมูลจริง (Empirical test) ซึ่งประกอบด้วย ข้อมูลจาก TSP Library และสถานที่ตั้งจริงของห้างสรรพสินค้าและไฮเปอร์มาร์เก็ตบางแห่งในประเทศไทย ผลการศึกษาแสดงให้เห็นว่าโดยเฉลี่ยแล้ว ขั้นตอนวิธี FI ให้คำตอบดีที่สุด โดยมีค่าเบี่ยงเบนจากคำตอบเหมาะที่สุดน้อยกว่า 10% สำหรับ TSP ที่มีสถานที่ไม่เกิน 50 แห่ง หากเลือกใช้คำตอบที่ดีที่สุดระหว่างวิธี NI และ FI จะทำให้ได้คำตอบเท่ากับคำตอบเหมาะที่สุดในการแก้ปัญหา TSP สำหรับการเดินทางไปยังสถานที่ต่างกันไม่เกิน 10 แห่ง สำหรับการแก้ปัญหา TSP ที่มีขนาดเท่ากัน วิธีที่ให้คำตอบเหมาะที่สุดใน Excel Solver ใช้เวลามากกว่า และขาดประสิทธิภาพในการแก้ปัญหาที่มีสถานที่มากกว่า 10 แห่งขึ้นไป การศึกษานี้ยังได้อธิบายการใช้ Google Maps เพื่อระบุตำแหน่งและการใช้สูตร Great Circle ประมาณระยะทางจากพิกัดละติจูด/ลองจิจูด อาจกล่าวโดยสรุปได้ว่า ฮิวริสติกส์ NI และ FI สามารถนำไปใช้แก้ปัญหา TSP ขนาดเล็กได้ วิธีการจากการศึกษานี้สามารถนำไปประยุกต์ใช้ได้โดยบุคคลทั่วไป พนักงานขาย และบริษัทขนาดเล็ก หากไม่มีซอฟต์แวร์แก้ปัญหาการหาค่าเหมาะที่สุด

60 หน้า

# CONTENTS

# CONTENTS (cont.)

# LIST OF TABLES

# LIST OF FIGURES

# CHAPTER I
# INTRODUCTION

## 1.1 Background of the Study

The Traveling Salesman, or Salesperson, Problem (TSP) is an optimization problem with the classic goal of finding the shortest tour visiting each city exactly once before returning to the starting city. Although the problem statement is simple, solving TSP is difficult. No effective method is known for the general case especially when the number of locations grows larger. The number of feasible solutions and the computation time are exponentially increased with the size of the problem.

TSP importance arises from the variety of its applications either direct or appear as a sub-problem in many areas. It is the fundamental problem to the Vehicle Routing Problem (VRP) seeking to supply demand with a fleet of vehicles. Other applications include school bus routing, mail routing, circus tour routing, DNA sequencing, machine sequencing and scheduling, computer wiring, microchips manufacturing (Hahsler & Hornik, 2010).

There are exact algorithms and heuristics to solve the TSP. The exact methods of Integer Programming include LP Relaxations, Branch and Bound, Cutting Planes, Branch and Cut. As the number of nodes is significantly increased, the exact methods become inefficient. A number of heuristics and metaheuristics are extensively developed to obtain the near-optimal solutions within a reasonable computation time. The heuristics in this study to solve the TSP include tour construction and tour improvement heuristics. Examples of construction heuristics include Nearest Neighbor and insertion algorithms such as Nearest Insertion, Farthest Insertion, Cheapest Insertion algorithms. Popular metaheuristics for combinatorial problems include Genetic Algorithms, Tabu Search, Neural Network, Simulated Annealing and Ant Colony optimization.

## 1.2 Statement of the Problem

The TSP involves in our daily lives and also relates to various types of businesses. The size of problems ranges from a few visiting locations to thousands of points of wiring. However, the intent of this study is to implement simple heuristics, which often have an intuitive justification and give good results, to solve TSP for small-sized problems. From time to time, people face with a problem of this kind where they depart from somewhere to visit some other places and return back to their bases. For example, a family leaves home to go shopping at three department stores, have lunch at a restaurant, visits relatives at their places, then dine at another restaurant and return home. The TSP can be commonly applied for work or business perspectives. A salesman with direct marketing campaign starts from his office to visit his customers at their homes or working places and returns back to the office. A home-based territory/area manager departs from home to visit gas stations within his/her area of responsibility. A delivery vehicle departs from an office to deliver goods or products to their customers in different villages or areas before returning back.

This study focuses on attacking the TSPs with the size that are commonly occurred for people in general, saying locations less than fifty. This size of problem could also be applied to small and medium enterprises where the software to solve for optimal is not in place. This study implements the three simple heuristics and the assessment by empirical analysis is conducted with samples of department stores in Thailand. How to use the tool to solve TSP with an assistance of Google Maps will also be provided.

## 1.3 Objectives of the Study

1.3.1 To implement the three construction heuristics, namely, Nearest Neighbor, Nearest Insertion, and Farthest Insertion heuristics to solve the Traveling Salesman Problem with the selected instances.

1.3.2 To compare and assess the results of the selected three heuristics according to two criteria i.e. solution quality and time efficiency.

## 1.4 Significance of the Study

The deliverables of the study include the method to solve the TSP with three simple heuristics and the empirical analysis for these heuristics. The tool is constructed to demonstrate the methods to assist the decision making on planning the tour. This study intends to increase awareness on route planning to shorter distance needed and then save energy. Salespersons or some other small businesses e.g. the laundry service with a pickup van can benefit the short tour provided by the tool. People or tourists can plan their routes or trips with this tool with an assistance of Google Maps. The comparison of the results among the three heuristics for the testing instances can be used as a reference. The knowledge drawn from this study can be used to solve other instances of the TSP and extended to apply with other heuristics or more advanced metaheuristics.

## 1.5 Scope of the Study

1.5.1 This study relates to solve the basic TSP. No demand or time window restriction is associated with planning the route.

1.5.2 For a comparative purpose, Euclidean distance between a pair of locations is used.

# CHAPTER II
# LITERATURE REVIEW

## 2.1 Overview of the Traveling Salesman Problem (TSP)

Optimization problems are divided naturally into two categories: those with continuous variables, and those with discrete variables, which is called combinatorial (Papadimitriou C. & Steiglitz K., 1998). The Traveling Salesman, or Salesperson, Problem (TSP) is an intensively studied problem in combinatorial optimization aiming to find the shortest tour visiting each member of a collection of locations exactly once and returninf to the starting location. Solving TSP is not as simple as its statement since it is NP-complete (Hahsler M. & Hornik K., 2010).

Non-deterministically Polynomial, or NP problem is a problem where its solution comes from a finite set of possibilities and its candidate solutions are verifiable in polynomial time. NP-hard problem is basically a problem that is at least as hard as the hardest problems in NP and no efficient exact algorithm has been discovered to solve the problem optimally. In addition, NP-complete problem is a problem which is both NP and NP-hard. Even though a solution for NP problem can be verified in polynomial time, no known efficient and fast way exists to find a solution. With any currently known algorithms, the time required to solve the problem increases significantly quickly as the size of the problem grows. Solving moderately large TSPs by any computing power available currently can easily take time as billions of years. Thus, NP-complete problems are often solved by approximation algorithms.

TSP is an NP-Complete problem. To determine the optimal tour for a TSP is generally much more difficult than finding a shortest path on a network. The TSP is a decidable problem, since it can be solved by finding the best among a finite set of tours. Thus a computer could solve any instance of the TSP by systematically and exhaustively examining and evaluating all tours and then choosing the shortest one. The solution by this algorithm of a modestly sized instance of the TSP thus require many billion years, even under the most optimistic assumptions about the speed of

computers in the future. Finding the best tour of a 30-city TSP would involve the possibilities of 30! or about $2.65\times10^{32}$ different tours. Assuming a trillion additions per second, solving this 30-city TSP takes around 252,333,390,232,297 years. Adding one more city requires more time by a factor of 31.

Given the starting location, the number of feasible solutions is $(n-1)!$ for an asymmetric TSP and $(n-1)!/2$ for a symmetric TSP. A symmetric TSP is the one with same distance in each opposite direction between two locations. Examples of asymmetric TSPs include one-way road, traffic collision, road obstruction, and transportation fares for cities with different departure and arrival fees.

A classical definition of TSP is a problem that starts from a location, traverses to other cities exactly once and returns to the starting location with minimum total distance. The cities can be represented by other entities, for example, customers and soldering points and the distance can be replaced by travelling time or cost. Furthermore, solving TSP can be extended to various applications in different areas. For instance, machine scheduling for drilling holes in a circuit board where cities are replaced with holes to be drilled, and distance is replaced with the travel time moving the drill head from one hole to another. If this applied TSP can be solved optimally, the manufacturing process can be improved by reducing travel time of the drilling device hence reducing costs. Solving TSP in real-life applications would be considerably harder with additional constraints such as time windows or limited resources.

Other TSP in transportation and logistics applications include the routing of school to pick up students, the routing of trucks for parcel post pickup, the transportation from one location to another, the scheduling of service calls, the delivery of goods, and so on. Other applications include microchips manufacturing, frequency assignment in communication networks, ordering and clustering objects in statistical data analysis which is currently becoming popular in biostatistics (Hahsler M. & Hornik K., 2010).

In 1990, Gerhard Reinelt collected a library of test instances called TSPLIB. These test instances are used by researchers to measure the performance of proposed solution algorithms. The very first notable TSPLIB is “dantzig42” which is

the 42-city TSP, excluding 7 cities that the route is passed by, solved by Dantzig G., Fulkerson R., and Johnson S., 1945.

There are a number of studies in the algorithms for solving TSP and real applications to the business sectors and a wide range of industries. Chungcharoen P. (2001) developed an application program for selection of tourist's traveling routes using branch and bound technique demonstrated with the five popular provinces in tourism in southern Thailand including Phuket, Phang-Nga, Krabi, Surat-Thani, and Nakhon Si Thammarat. Jiranan (2007) solved the General TSP problem to obtain the optimal route for a bank vehicle to replenish the banknote of a group of ATMs.

## 2.2 Exact Methods for the TSP

Exact or explicit methods are the ones to solve for the optimal solutions.

### 2.2.1 TSP Formulation

Let $N$ be a set of $n$ locations or nodes and binary decision variables $x_{ij}$ for nodes $i$ and $j$, where:

$$x_{ij} = \begin{cases} 1 & \text{if the solution to TSP goes from node } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

Let $c_{ij}$ be the cost or distance associated with each variable $x_{ij}$. A TSP Integer Programming formulation is shown as follows:

$$\min z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1}$$

s.t.

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad \forall i = 1,2,...,n \tag{2}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad \forall j = 1,2,...,n \tag{3}$$

$$0 \le x_{ij} \le 1 \qquad \forall (i,j) \tag{4}$$

$$u_i - u_j + nx_{ij} \le n-1 \qquad \text{for } i \ne j; i = 2,3,...,n; j = 2,3,...,n \tag{5}$$

$$x_{ij} \text{ binary}, u_j \ge 0 \tag{6}$$

Number of constraints grows exponentially as number of locations increases. Constraints (2) and (3) ensure that the feasible TSP tour has an

"assignment" property i.e. each node *i* is assigned to exactly node *j*. This assignment property is insufficient to define a feasible solution since it allows subtours. A subtour in *N* is a cycle or loop that does not include all nodes. Thus, constraint (5) is added into TSP formulation to ensure that a feasible solution is the tour, not only subtour. However, constraint (5) disrupts the unimodularity of the assignment constraints i.e. the feasible solutions can be fractional values. Consequently, constraint (6) is added explicitly in the TSP formulation to ensure integer solutions.

### 2.2.2 Enumeration algorithm

Enumerating every possible tour and selecting the one with minimum cost is one simple way to find an optimal TSP solution. A possible tour for a network with *n* nodes is $(n-1)!$. From a start node, there are $(n-1)$ nodes to be possibly chosen next, then $(n-2)$ nodes, and so on. For the symmetric TSP which corresponds to the undirected network, half of these tours are the reverse of other tours and the number of possibilities becomes $(n-1)!/2$. In conclusion, the enumerative method is an order of $n!$ or $O(n!)$ algorithm which is inefficient for large problem and will not be improved by the computation speed. Table 2.1 shows the proportion of factorial function comparing to other polynomial factors.

**Table 2.1 Order of *n***

| $n$ | $n^2$ | $n^3$ | $2^n$ | $n!$ |
|---|---|---|---|---|
| 10 | 100 | 1,000 | 1,024 | 3,628,800 |
| 50 | 2500 | 125,000 | $1.13\times10^{15}$ | $3.04\times10^{64}$ |
| 100 | 10,000 | 1,000,000 | $1.27\times10^{30}$ | $9.33\times10^{157}$ |

In a year, there are approximately 31.5 million seconds. Currently, one of the world's fastest computers can perform a peak of 41,000 gigaflops (billion operations per second). This performance is not sustainable but suppose it was, solving TSP by enumeration on a problem with 100 nodes with such computer would require 981.5 million years to complete 2100 operations, and $7.226\times10^{157}$ years to complete 100! operations.

### 2.2.3 Branch and Bound (B&B)

The first B&B algorithm was developed in 1960 by Land A. and Doig G. for the general mixed and pure Integer Linear Programming problem. This section will present one approach of B&B to solve TSPs in which the subproblems reduce to assignment problems (Winston, 2004).

Subproblems are assignment problems with the Integer Programming formulation with constraints (1) – (4) in section 2.2.1. If the optimal solution to a subproblem contains no subtours, then it is a feasible solution to the TSP. Then, create new subproblems by branching to exclude a subtour and eliminate a subproblem if its optimal *z*-value is inferior to the best previously found feasible solution. A problem of the symmetric TSP with the pair-wise distance matrix shown in Table 2.2 is solved by B&B algorithm as a demonstration.

**Table 2.2 A pair-wise distance matrix of 5 places**

| Place | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0 | 132 | 217 | 164 | 58 |
| 2 | 132 | 0 | 290 | 201 | 79 |
| 3 | 217 | 290 | 0 | 113 | 303 |
| 4 | 164 | 201 | 113 | 0 | 196 |
| 5 | 58 | 79 | 303 | 196 | 0 |

A solution for the assignment problem can be a tour or two or more subtours. A tour is an itinerary beginning and ending at the same place and traverses through every place exactly once. A subtour, on the other hands, is a round trip not passing through all places. If the assignment problem is solved and yields a tour as the solution, then it is optimal solution to the TSP.

To begin, let cost $c_{ij}$ be the distance between places $i$ and $j$ and $c_{ii} = M$ where $M$ is a very large number. The general assignment problem contains no constraints to prevent subtours. In other words, TSP is a relaxation or less constrained problem of the original TSP. If the optimal solution to the assignment problem is feasible for the TSP i.e. the assignment solution contains no subtours, it is also optimal for the TSP. The results of the B&B procedure are given in Figure 2.1.

Let $z$ = objective function value which is the total distance. Firstly, solving the assignment problem in Subproblem 1 obtains $z = 495$ and the optimal solution as $x_{15} = x_{21} = x_{34} = x_{43} = x_{52} = 1$. This solution contains two subtours, 1-5-2-1 and 3-4-3, which cannot be the optimal solution. Then, branch on Subproblem 1 to prevent one of Subproblem 1's subtours from recurring to subsequent subproblems. By choosing to exclude the subtour 3-4-3, the optimal solution to the TSP is either $x_{34} = 0$ or $x_{43} = 0$.

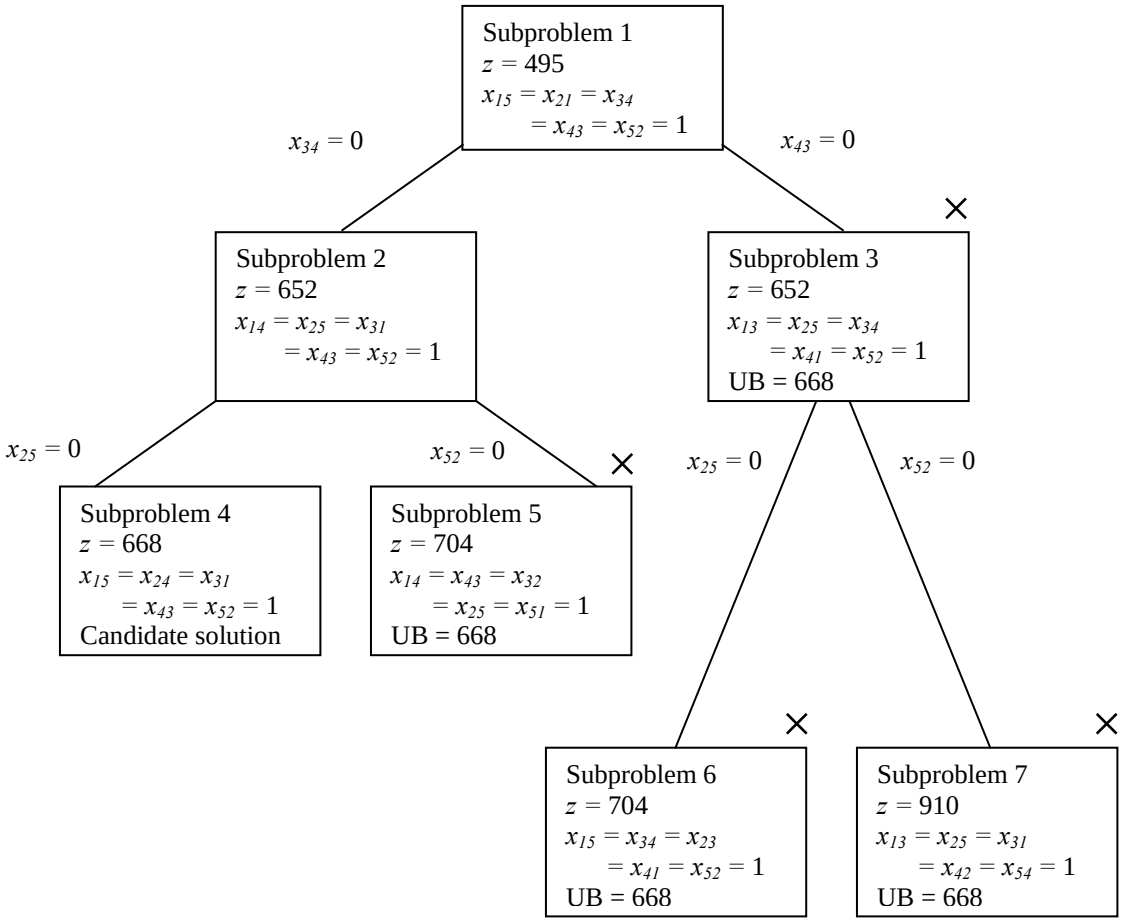


**Figure 2.1 Branch and Bound tree for a TSP**

Subproblem 1 can be branched by adding the following two subproblems.

Subproblem 2: Subproblem 1 + ( $x_{34} = 0$ or $c_{34} = M$ )

Subproblem 3: Subproblem 1 + ( $x_{43} = 0$ or $c_{43} = M$ )

By arbitrarily choosing Subproblem 2 to solve next manually with Hungarian method (Winston, 2004), the optimal solution to Subproblem 2 is $x_{14} = x_{25} = x_{31} = x_{43} = x_{52} = 1, z = 652$. This solution includes a subtour 1-4-3-1 and 2-5-2, so this cannot be the optimal solution to the TSP. Branching on Subproblem 2 in an effort to exclude the subtour 2-5-2 and ensure that either $x_{25}$ or $x_{52}$ equals zero. Thus, the following two subproblems are addeded.

Subproblem 4: Subproblem 2 + ( $x_{25} = 0$ or $c_{25} = M$ )

Subproblem 5: Subproblem 2 + ( $x_{52} = 0$ or $c_{52} = M$ )

Following the Last In First Out (LIFO) approach is solving Subproblem 4 or Subproblem 5 and Subproblem 4 is arbitrarily chosen. Applying the Hungarian method, the optimal solution $x_{15} = x_{24} = x_{31} = x_{43} = x_{52} = 1, z = 668$ is obtained. This solution contains no subtours and obtains the 1-5-2-4-3-1 tour. Thus, Subproblem 4 yields a candidate solution with $z$ = 668. Any node that cannot yield a z-value less than 668 may be eliminated from consideration.

Following LIFO rule, Subproblem 5 is solved next. The optimal solution to Subproblem 5 is $x_{14} = x_{43} = x_{32} = x_{25} = x_{51} = 1, z = 704$. This solution is a tour, but $z$ = 704 is not as good as the Subproblem 4 candidate's $z$ = 668. Thus, Subproblem 5 is removed from consideration and only Subproblem 3 remains. The optimal solution of $x_{13} = x_{25} = x_{34} = x_{41} = x_{52} = 1, z = 652$ is found. This solution contains the subtours 1-3-4-1 and 2-5-2. Since 652 < 668, it is possible for Subproblem 3 to yield a solution with no subtours that better than $z$ = 668. Thus, branching on Subproblem 3 is an effort to exclude the subtours. Any feasible solution to the TSP that emanates from Subproblem 3 must have either $x_{25} = 0$ or $x_{52} = 0$, so Subproblem 6 and 7 are created.

Subproblem 6: Subproblem 3 + ( $x_{25} = 0$ or $c_{25} = M$ )

Subproblem 7: Subproblem 3 + ( $x_{52} = 0$ or $c_{52} = M$ )

Next, Subproblem 6 is chosen and the optimal solution is $x_{15} = x_{34} = x_{23} = x_{41} = x_{52} = 1, z = 704$. This solution contains no subtours, but its $z$-value of 704 is inferior to the candidate solution from Subproblem 4, so Subproblem 6 will not give the optimal solution to the problem. The only remaining Subproblem is Subproblem 7 and the optimal solution is $x_{13} = x_{25} = x_{31} = x_{42} = x_{54} = 1, z = 910$.

Again, $z$ = 910 is inferior to $z$ = 668, so Subproblem 7 cannot yield the optimal solution. Subproblem 4 thus yields the optimal solution with a total distance of 668.

In conclusion, Branch and Bound method works by successively solving a number of relaxed problems to the original Integer Programming problems by Simplex algorithm efficiently. If a non-integer solution results from the solution of a relaxed problem, branching is required. If an integer solution is obtained from a relaxation of Linear Programming, this solution is a feasible candidate and the best integer candidate solution is maintained to the original problem. Branch and Bound method either finds a better solution, or determines that no better solution can be obtained.

### 2.2.4 Other algorithms

Other exact methods include Dynamic Programming, Cutting Plane, and Branch and Cut algorithms. Dynamic programming is a technique to compute recurrences by storing partial results and re-using them when needed. Cutting Plane method iteratively refines a feasible set or objective function by linear inequalities called cuts. Branch and Cut is a combination of Branch and Bound and Cutting Plane methods. Branch and Cut also uses branching to create a partially-enumerated search tree and the subtour elimination constraints will be imposed when violations occurred. For Branch and Cut method, new constraints (or cuts) may be introduced to the formulation prior to branching. Each cut eliminates some subset of solutions to the current relaxed problem without eliminating any feasible solutions to the original problem. Identifying violated cuts requires computational effort. One important tradeoff in designing Branch and Cut methods is to balance the amount of branching with the effort expended finding and introducing violated constraints.

## 2.3 Heuristics for the TSP

Using Linear Programming for small instances, the optimal solutions can be obtained in reasonable time. However, solving larger instances with guaranteed optimality will be very time-consuming (Nilsson C., n.d.). When using exact or explicit methods such as Branch and Bound to solve TSP with many nodes, large amount of time is required. Heuristics are used instead since they often lead to a good

feasible solution quickly. Good heuristics give solutions that are close to the optimal solution, and usually are efficient in terms of theoretical or practical running time (Erera A., 2009).

TSP heuristics can be categorized into two groups i.e. tour construction heuristics which form a tour from scratch and tour improvement heuristics which use simple local search heuristics to improve existing tours (Hahsler & Hornik, 2010). Tour construction heuristics include Nearest Neighbor and the collection of insertion algorithms. Tour improvement heuristics include *k*-opt heuristics and Lin-Kernighan heuristic.

### 2.3.1 Nearest Neighbor algorithm

The Nearest Neighbor algorithm (Rosenkrantz D. J., Stearns R. E., and Lewis II P. M., 1977) is a fast algorithm and perhaps the most straightforward TSP heuristics. It follows a greedy local search procedure maintains at each step a current path. The algorithm starts in a random or chosen node and in turn selects the closest unvisited node from either one or both current path endpoints. It always adds the closest neighbor to the tour and finally connects the remaining two endpoints together. The algorithm stops when all nodes are on the tour. The length of tour depends on the starting point i.e. different start points result in different tours.

An extension to this algorithm, called repetitive Nearest Neighbor, is to repeat it with each node as the starting point and return the best tour found (Hahsler & Hornik, 2010). With this extension, the tours for every node will be selected as the starting point and returns the shortest one as the solution. Solutions given by this algorithm often contains crossing edges especially the edge connecting the last and the first nodes can be quite long.

Given an undirected network $G = (N,A)$ with distance $d_{ij}$ for each $\{i, j\} \in A$. Let $T$ be the tour consisting of all nodes. The Nearest Neighbor algorithm can be given as follows:

Initialization:

Alternative 1: Let $P = \{i, j\}$ where arc $\{i, j\}$ has minimum distance among all arcs

Alternative 2: Choose an arbitrary start node *i*, and find $j \in N \setminus \{i\}$ such that $d_{ij}$ is minimized. Let $P = \{i, j\}$.

Steps: while $P < N$

1. Let $s = begin(P)$ and $t = end(P)$, the nodes at the beginning and end of the current path.

2. Choose the node $k \in N \setminus P$ to enter the path:

Alternative 1: Let $k$ be such that $d_{tk}$ is minimized. $P = P + \{k\}$.

Alternative 2: Let $k$ be such that $d_{tk}$ or $d_{ks}$ is minimized. The new path $P$ is either $P + \{k\}$ or $\{k\} + P$ respectively.

End

Let $T = P + \{begin(p)\}$.

### 2.3.2 Insertion algorithms

Insertion algorithms (Rosenkrantz et al., 1977) start with a tour consisting of an arbitrary node. At each step, a node *k* not yet visited is selected. This node is inserted into the existing tour between two consecutive nodes *i* and *j*, such that the added tour length is minimized. The algorithms stop when all nodes are in the tour. The added tour length or the increased cost is calculated as shown below, assuming *k* the selected node for insertion.

$$d(i,k) + d(k,j) - d(i,j)$$

There are the selection step and the insertion step. An insertion procedure takes a subtour on a number of nodes at one iteration and determines which of the remaining nodes shall be inserted to the subtour next (the selection step) and between which two nodes it should be inserted (the insertion step). By the nature of the method, the tour should never contain any edge crossings. The insertion algorithms are different in the selection step that how the node to be inserted is chosen next, but the insertion step is the same.

#### 2.3.2.1 Nearest Insertion algorithm

With the Nearest Insertion heuristic, start the tour by first choosing the arc between the pair of the closest nodes with the minimum distance. Next, create a subtour joining those two nodes and replace an arc with the insertion algorithm described earlier. Continue with this procedure until the best tour is obtained.

Given an undirected network $G = (N,A)$ with distance $d_{ij}$ for each $\{i, j\} \in A$. Let $T$ be the tour consisting of all nodes. The Nearest Insertion algorithm can be given as follows:

Initialization: Find the arc $(i, j)$ with minimum distance in $A$. Let $T = \{i, j, i\}$.

Steps: while $|T| < n+1$

1. Find the node $j \notin T$ that is closest to a node currently in $T$.

2. Find the insertion location for node $j$ into the existing tour. To do so, find the arc $(l,k)$ in the current tour with the minimum added length $d(l, j) + d(j,k) - d(l,k)$. Insert $j$ between $l$ and $k$ in $T$.

End

2.3.2.2 Farthest Insertion algorithm

Farthest Insertion heuristic starts the tour by first choosing the arc between the pair of the farthest nodes with the maximum distance. Next, replace an arc in the subtour with the minimum added tour length of the selected node. There are three possible variations for the Farthest Insertion algorithm in the selection step.

1. Farthest Insertion 1: insert the node whose minimal distance to a tour node is maximal.

2. Farthest Insertion 2: insert the node that has maximum distance to a tour node.

3. Farthest Insertion 3: insert the node whose maximal distance to a tour node is minimal.

Given an undirected network $G = (N,A)$ with distance $d_{ij}$ for each $\{i, j\} \in A$. Let $T$ be the tour consisting of all nodes. The procedure below shows the Farthest Insertion 1 algorithm.

Initialization: Find the arc $(i, j)$ with maximum distance in $A$. Let $T = \{i, j, i\}$.

Steps: while $|T| < n+1$

1. Let $D_j(T)$ be $\min_{i \in T} d_{ij}$ for all nodes $j \notin T$. Select the node $j$ with maximum $D_j(T)$.

2. Insert node *j* into the existing tour by finding the arc $(l,k)$ in the current tour with minimum added tour length $d(l,j)+d(j,k)-d(l,k)$. Insert *j* between *l* and *k* in *T*.

End

Step 1 finds the node *j* not on the tour for which the minimum distance to a node on the tour is maximal, and adds that node to the current tour. Farthest Insertion algorithms try to link nodes far outside into the tour first to establish the overall layout of the tour early in the insertion step.

2.3.2.3 Cheapest Insertion algorithm

Cheapest Insertion algorithm follows the pure greedy method. At each step, the node *k* is chosen such that the minimum added tour length incurs.

Given an undirected network $G = (N,A)$ with distance $d_{ij}$ for each $\{i,j\} \in A$. Let *T* be the tour consisting of all nodes.

Initialization: Find the arc $(i,j)$ with maximum distance in *A*. Let $T = \{i, j, i\}$.

Steps: while $|T| < n+1$

1. Let $D_{jl}(T)$ be $d_{lj} + d_{j\,next(l)} - d_{l\,next(l)}$ for all nodes $j \notin T$.

2. Let *j** and *l** be the indices of the minimum distance $D_{jl}$. Insert node *j* after node *l* in *T*.

End

2.3.2.4 Other Insertion algorithms

Other Insertion algorithms include Arbitrary Insertion, Largest Sum Insertion and Smallest Sum Insertion algorithms. For Arbitrary Insertion, the node *k* is chosen randomly from all unvisited nodes. Largest Sum Insertion algorithm inserts the node with maximum of the sum of distances to tour nodes while Smallest Sum Insertion algorithm inserts the node whose sum of distances to tour nodes is minimal.

**2.3.3 *k*-opt heuristics**

A *k*-opt move defines another tour which is a neighbor of an original tour by deleting *k* edges and replacing them by a set of different feasible edges. With this algorithm, the tour is improved iteratively by moving from one tour to its best neighbor. In practice, 2-opt and 3-opt heuristics are generally used. To illustrate the *k*-

opt heuristic, refer to the 2-opt algorithm in Figure 2.2. The 2-opt algorithm removes two edges from the tour, and reconnects the tour by another different two edges. This procedure is done if the new shorter tour will be obtained. The 2-opt method improves tour by reconnecting and reversing order of subtours. Every pair of edges is checked if an improvement is possible. The procedure is repeated until no further improvement can be done.

**Figure 2.2 A 2-opt move**

A tour in 2-dimensional Euclidean space containing crossing edges can always be improved by a 2-exchange, a swap of the two crossing edges with two non-crossing edges. Starting with the nearest neighbor tour, use 2-exchanges to eliminate all crossing edges. Make one 2-exchange at a time, and record the decrease in total distance for each one. When there are no more edge crossings, the total distance obtained is the solution.

### 2.3.4 Lin-Kernighan heuristic

Instead of using a fixed *k* value, Lin-Kernighan heuristic (Lin S. and Kernighan B, 1973) finds the best *k* for each move which can be represented as a series of 2-opt moves. The algorithm constructs a sequence of 2-opt moves and checks after each additional move whether a stopping rule is met (Hahsler & Hornik, 2010). These moves are applied until a local optimum is reached.

## 2.4 Evaluation of heuristics

Heuristics can be evaluated by three methods (Winston, 2004).

### 2.4.1 Performance guarantee

A performance guarantee gives a worst-case bound for the solution obtained from a heuristic comparing to optimality. For the worst solution of a TSP, if the optimal distance is *D*, the heuristic finds a solution no greater than $\alpha D$. For Nearest

Insertion and Cheapest Insertion algorithms, the worst case scenario for a symmetric TSP satisfying the triangle inequality is *2D*. In other words, Nearest Insertion and Cheapest Insertion algorithms cannot exceed twice the length of the optimal tour. The worst case scenario for Farthest Insertion is $2\ln(n)+0.16$. The worst case scenario for Nearest Neighbor is $0.5\times\lfloor\log_2(n)\rfloor+0.5$.

### 2.4.2 Probabilistic analysis

A heuristic can be evaluated with the assumption that the locations follow a known probabilistic distribution. For example, the locations are assumed independently uniformly distributed on a cube of unit length, width, and height. Then the ratio of the expected length by the heuristic and the expected length of an optimal tour will be computed. The closer the ratio is to one, the better the heuristic.

### 2.4.3 Empirical analysis

Heuristics can be compared to the known optimal solution for a number of problems. Golden, Bodin, Doyle, and Stewart (1980) found that, for five 100-city TSPs, Nearest Neighbor heuristic gives an average tour length around 15% longer than the optimal tour. For the same set of problems, Cheapest Insertion heuristic also produces tours that averaged 15% longer than the optimal tour (Winston, W., 2004).

Arman Boyaci (2007) stated that Farthest Insertion gives the best solutions among the three classical construction heuristics i.e. Nearest Neighbor, Arbitrary Insertion, and Farthest Insertion. For small instances, Farthest Insertion gives 8% gap from the optimal values. However, this gap increases rapidly respect to instance size. Also, Farthest Insertion requires less computational time.

## 2.5 Distance Calculation Methods

Let $d$ be the distance between the two points. There are several ways to determine the distance.

### 2.5.1 Euclidean distance

In Cartesian coordinates, if $p=(p_1,p_2,...,p_n)$ and $q=(q_1,q_2,...,q_n)$ are two points in Euclidean $n$-space, the distance from $p$ to $q$ is given by

$$d = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}$$

In three-dimensional Euclidean space, if the Cartesian coordinates of two points $p = (x_1, y_1, z_1)$ and $q = (x_2, y_2, z_2)$, the distance from $p$ to $q$ is given by

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

In two-dimensional space or a plane, the distance of the two points $p = (x_1, y_1)$ and $q = (x_2, y_2)$ is given by

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

### 2.5.2 Manhattan distance

Manhattan distance is the distance measured between two points along two axes of the right angle, also known as rectilinear distance, taxi cab metric, or city block distance. Given two points $p = (x_1, y_1)$ and $q = (x_2, y_2)$ in a plane, Manhattan distance is calculated as follows:

$$d = |x_1 - x_2| + |y_1 - y_2|$$

### 2.5.3 Geographical distance

Geographical distance is the distance measured between two points along the earth's surface. The points are defined by geographical coordinates or pairs of latitude and longitude which are usually expressed in degrees. Latitude is the angle in degrees above or below the equator. Longitude, on the other hand, is the angle east or west of the Greenwich meridian. The distance between any two real locations can be estimated by a perimeter between two points on a sphere.

Calculating the distance between geographical coordinates does not provide an exact distance. It is based on some level of abstractions including flat surface, spherical surface, and ellipsoidal surface. The ellipsoidal formulas are somewhat complicated and will be omitted.

Let the geographical coordinates of the two points be $(LAT_1, LONG_1)$ and $(LAT_2, LONG_2)$ respectively.

$d$ = distance between the two points, specifically along the surface of the earth

$LAT_1$, $LAT_{1R}$ = latitude of the first point in degrees, in radians

$LONG_1$, $LONG_{1R}$ = longitude of the first point in degrees, in radians

$LAT_2$, $LAT_{2R}$ = latitude of the second point in degrees, in radians

$LONG_2, LONG_{2R}$ = longitude of the second point in degrees, in radians

$R$ = the radius of the earth (6371.009 km = 3958.761 miles = 3440.069 nautical miles)

2.5.3.1 Flat surface formulas

A flat surface approximation for the earth's surface may be useful over small distances. The accuracy of this formula becomes increasingly inaccurate as the separation between the points becomes greater (Wikipedia, 2010).

1. Pythagorean formula with parallel meridians

A meridian is a longitude line which is an imaginary arc on the earth's surface from the North pole to the South pole, and perpendicular to all circles of latitude. This formula is not accurate enough for general use but with acceptable accuracy for points near the equator.

$$d = R \times \sqrt{(LAT_{2R} - LAT_{1R})^2 + (LONG_{2R} - LONG_{1R})^2}$$

2. Spherical earth projected to a plane

This formula takes into account the distance variation between meridians with latitude.

$$d = R \times \sqrt{(LAT_{2R} - LAT_{1R})^2 + (\cos((LAT_{1R} + LAT_{2R})/2)(LONG_{2R} - LONG_{1R}))^2}$$

2.5.3.2 Spherical-surface formulas

If a possible error of 0.5% is acceptable, the formulas of spherical trigonometry that best approximates the surface of the earth can be used.

1. Tunnel distance

For a spherical earth, a tunnel between points on Earth is defined by a chord of the Great Circle. For points near each other, the tunnel distance is slightly less than the great-circle distance. For the corresponding unit sphere, the Great Circle chord length can be calculated as follows.

$$\Delta X = \cos(LAT_{2R})\cos(LONG_{2R}) - \cos(LAT_{1R})\cos(LONG_{1R})$$

$$\Delta Y = \cos(LAT_{2R})\sin(LONG_{2R}) - \cos(LAT_{1R})\sin(LONG_{1R})$$

$$\Delta Z = \sin(LAT_{2R}) - \sin(LAT_{1R})$$

$$d = R \times \sqrt{(\Delta X)^2 + (\Delta Y)^2 + (\Delta Z)^2}$$

2. Great Circle distance

Great Circle distance is the shortest distance measured along a path on the sphere's surface between any two points on the surface instead of going through the sphere's interior.

(1) Great Circle distance approximate formula 1

This formula applies the trigonometry relationship and estimates that 1 latitude degree is 69.1 miles. The distance of 1 longitude degree varies from one location to another, thus the average of 53 miles per longitude degree is used. The distance $d$ for the Great Circle distance in unit of miles is as follows:

$$d = \sqrt{(69.1 \times (LAT_2 - LAT_1))^2 + (53 \times (LONG_2 - LONG_1))^2}$$

(2) Great Circle distance approximate formula 2

To be more precise than the approximate formula 1, the distance $d$ between longitude degrees refers to the position on the sphere.

$$d = \sqrt{(69.1 \times (LAT_2 - LAT_1))^2 + (69.1 \times (LONG_2 - LONG_1) \times \cos(LAT_1 / 57.3))^2}$$

(3) Great Circle distance formula

By applying spherical trigonometry formula, this formula calculates the exact distance between points.

$$d = 69.1 \times 180 / \pi \times \arccos\left[\sin(LAT_{1R}) \times \sin(LAT_{2R}) + \cos(LAT_{1R}) \times \cos(LAT_{2R}) \times \cos(LONG_{2R} - LONG_{1R})\right]$$

# CHAPTER III
# METHODS AND DATA

## 3.1 General Procedures

This study aims to compare among the three classical construction heuristics, namely Nearest Neighbor, Nearest Insertion and Farthest Insertion heuristics. It is also the intent to implement these heuristics for solving TSP with real locations with constructed tool. The general procedures are as follows:

3.1.1 Define the exact algorithm steps of each heuristic

3.1.2 Determine the methods to calculate distance

The key input for solving TSP is the distance between each pair of locations. The distance can be obtained or estimated by history or reliable maps. In addition, some forms of the approximate or exact formulas can be used to calculate the pair-wise distance from the geographical locations' coordinates.

3.1.3 Collect data

Two groups of data set are used for this study. The first group of data is obtained from the TSP Library, or so called TSPLIB available online from the reliable source. Each TSPLIB has either coordinates of the nodes or the pair-wise distance matrix as input. Since this study focuses more on the application of small and medium enterprises, most of the TSPs selected for comparison purpose are within a range of a hundred nodes. The second group of data set consists of real locations which are branches of the selected department stores and hypermarkets. The distance between each of these places is either the real distance from the suggested route in Google Maps or the distance estimated from the latitude-longitude coordinates which also obtained from Google Maps.

3.1.4 Construct the tool for heuristics implementation

This step includes coding and validating the algorithms, and developing user interface for input and output.

3.1.5 Compare and analyze the results

This study compares the result from empirical analysis with the two data sets i.e. TSPLIB and the real selected locations. For both data sets, the three heuristics are basically compared based on two main criteria which are solution quality and time efficiency. The solutions from the heuristics are compared against the optimal solutions for the TSP instances from TSPLIB and from groups of real locations with up to 13 places for each group. The optimal solutions of these groups of real locations are obtained by the exact method i.e. Branch and Bound method in Excel Solver. Meanwhile, for the data set of greater than 13 real locations with known optimal solutions are not available in this study, the solutions obtained from the heuristics are compared and analyzed against one another.

3.1.6 Conclude the findings

The study concludes the performance of the three heuristics implemented based on the empirical analysis conducted. It also aims to comment if the constructed tool is effective for real applications.

## 3.2 Heuristic Algorithms for Implementation

In this study, Nearest Neighbor, Nearest Insertion, and the three types of Farthest Insertion heuristics are implemented for analysis. Even though each of the three heuristics is simple and intuitive in nature, the algorithm itself can be different in detail. This part determines the exact steps for each heuristic implemented in this study. For all algorithms implemented, the same convention is applied, that is an undirected network $G = (N,A)$ where $N$ is the set of all nodes, $A$ is the set of all arcs connecting the nodes, $T$ is the tour or subtour created, and $d_{ij}$ is the distance from node $i$ to node $j$.

### 3.2.1 Nearest Neighbor algorithm

This algorithm gives different results depending on the starting location. This study implements the repetitive Nearest Neighbor algorithm, that is, taking the best of all solutions found when the algorithm is applied beginning at each location. Once a

node is selected at each step, it can be inserted from one side of the path or both sides of the path, this study selects to insert from one side of the path.

Initialization: For every start node *i*, find $j \in N \setminus \{i\}$ such that $d_{ij}$ is minimized. Let $P = \{i, j\}$.

Steps: while $P < N$

1. Let $s = begin(P)$ and $t = end(P)$, the nodes at the beginning and end of the current path.

2. Choose the node $k \in N \setminus P$ to enter the path such that $d_{tk}$ is minimized.

3. $P = P + \{k\}$

End

Let $T = P + \{begin(p)\}$.

### 3.2.2 Nearest Insertion algorithm

The Nearest Insertion algorithm begins with the pair of locations with minimum distance. Then, the next node selected is the one that has minimum distance from the nodes already in the path and insert that node where it provides minimum added length.

Initialization: Find the arc $(i, j)$ with minimum distance in *A*. Let $T = \{i, j, i\}$.

Steps: while $|T| < n + 1$

1. Find the node $j \notin T, i \in T$ such that $d_{ij}$ is minimized.

2. Find the insertion location for node *j* into the existing tour by finding the arc $(l, k)$ in the current tour with the minimum added length $d(l, j) + d(j, k) - d(l, k)$. Insert *j* between *l* and *k* in *T*.

End

### 3.2.3 Farthest Insertion algorithms

This study implements the three variations of farthest insertions, each of which starts with the pair of locations with maximum distance. The insertion algorithm is the same as Nearest Insertion. The difference of the three steps is the selection step.

Initialization: Find the arc $(i, j)$ with maximum distance in *A*. Let $T = \{i, j, i\}$.

Steps: while $|T| < n + 1$

1. Farthest Insertion 1: Let $D_j(T)$ be $\min_{i \in T} d_{ij}$ for all nodes $j \notin T$. The node *j* to be inserted into the tour is one with maximum $D_j(T)$.

Or, Farthest Insertion 2: Let $D_j(T)$ be $\max_{i \in T} d_{ij}$ for all nodes $j \notin T$. The node *j* to be inserted into the tour is one with maximum $D_j(T)$.

Or, Farthest Insertion 3: Let $C_j(T)$ be $\max_{i \in T} d_{ij}$ for all nodes $j \notin T$. The node *j* to be inserted into the tour is one with minimum $D_j(T)$.

2. Insert node *j* into the existing tour by finding the arc $(l,k)$ in the current tour with minimum added tour length $d(l,j)+d(j,k)-d(l,k)$. Insert *j* between *l* and *k* in *T*.

End

## 3.3 Distance Calculation Methods for Implementation

Whenever TSP is solved, the distance between each pair of nodes or places is required. The geographical distance database can be obtained by the history if one has experienced traversing on the route connecting locations. Many times, the distances between two places are not known or readily available. The new customer who we have never made delivery is a simple yet frequently occurred example. In case the new locations are introduced, the estimation of geographical distances should be in place. The formulas used to calculate distance in this study are Euclidean and Great Circle assuming latitude-longitude coordinates are known.

### 3.3.1 Euclidean Distance Formula

Euclidean distance is the distance between two points measured with a ruler, and is obtained from Pythagorean formula. The TSPLIB instances in this study are based on Euclidean distance.

Let location 1 and 2 have coordinates $(x_1, y_1)$ and $(x_2, y_2)$, respectively.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

### 3.3.2 Great Circle Distance Formula

The distance between two points on the earth's surface can be calculated by using its latitude and longitude coordinates (Hexa Software Development Center, 2003). The latitude-longitude coordinates of the locations are referred from Google Map. Note that one latitude degree or one longitude degree equals $\pi/180$ radians.

The Great Circle formula applies spherical trigonometry function to calculate the exact distance between points. This study applies this formula for the data set with real locations.

$$d = 69.1 \times 180/\pi \times$$
$$\arccos\left[\sin(LAT_{1R}) \times \sin(LAT_{2R}) + \cos(LAT_{1R}) \times \cos(LAT_{2R}) \times \cos(LONG_{2R} - LONG_{1R})\right]$$

## 3.4 Data Collection

Two data sets are collected i.e. TSPLIB instances and the instances of the department stores and hypermarkets. Specifically the Central Group, Robinson, The Mall Group, Tesco Lotus and Carrefour branches are collected. These real locations are selected since they are conveniently located by Google Maps.

### 3.4.1 TSPLIB

TSPLIB is a Library of Standard TSP instances. It is a classic resource for anyone interested in the Travelling Salesman Problem. TSPLIB is comprised of more than 100 instances of the TSP collected by Professor Gerhard Reinelt in the mid 1990's. Reinelt is currently associated with the University of Heidelberg, and as such, the TSPLIB is housed on the website of the Discrete Optimization Research Group at the University. TSPLIB used in this study is shown in Table 3.1.

A number of distance measures are used to assess tours in TSP instances, for example, EUC_2D (Euclidean distance rounded to an integer), GEO (Geographical for assessing longitude and latitude coordinates), ATT (Euclidean distance with scaling and rounding), MATRIX (pre-defined distances in a matrix), and CEIL_2D (Euclidean distance rounded up). The majority of the TSP instances are of the EUC_2D type, many of which have known solutions. The number following the TSPLIB name is normally the number of cities or places.

**Table 3.1 The selected TSPLIB for comparison**

| No. | TSPLIB | *n* | Description | Distance input type | Optimal Solution |
|---|---|---|---|---|---|
| 1 | gr17 | 17 | 17 cities (Groetschel) | distance matrix | 2085 |
| 2 | gr24 | 24 | 24 cities (Groetschel) | distance matrix | 1272 |
| 3 | fri26 | 26 | 26 Staedte (Fricker) | distance matrix | 937 |
| 4 | bays29 | 29 | 29 cities in Bavaria, street distances (Groetschel,Juenger,Reinelt) | distance matrix | 2020 |
| 5 | dantzig42 | 42 | 42 cities (Dantzig) | distance matrix | 699 |
| 6 | swiss42 | 42 | 42 Staedte Schweiz (Fricker) | distance matrix | 1273 |
| 7 | gr48 | 48 | 48 cities (Groetschel) | distance matrix | 5046 |
| 8 | eli51 | 51 | 51 cities (Christofides/Eilon) | coordinates | 430 |
| 9 | berlin52 | 52 | 52 locations in Berlin (Groetschel) | coordinates | 7544 |
| 10 | st70 | 70 | 70 cities (Smith/Thompson) | coordinates | 679 |
| 11 | eli76 | 76 | 76 cities (Christofides/Eilon) | coordinates | 545 |
| 12 | pr76 | 76 | 76 cities (Padberg/Rinaldi) | coordinates | 108159 |
| 13 | rd100 | 100 | 100 cities, random TSP (Reinelt) | coordinates | 7910 |
| 14 | kroA100 | 100 | 100 cities, A (Krolak/Felts/Nelson) | coordinates | 21285 |
| 15 | kroC100 | 100 | 100 cities, B (Krolak/Felts/Nelson) | coordinates | 20751 |
| 16 | kroD100 | 100 | 100 cities, C (Krolak/Felts/Nelson) | coordinates | 21294 |
| 17 | eli101 | 101 | 101 cities (Christofides/Eilon) | coordinates | 642 |
| 18 | lin105 | 105 | 105 cities ((Lin/Kernighan)) | coordinates | 14383 |
| 19 | tsp225 | 225 | 225 cities (Juenger,Raecke,Tschoecke) | coordinates | 3859 |
| 20 | pcb442 | 442 | Drilling problem (Groetschel/Juenger/Reinelt) | coordinates | 50784 |
| 21 | pr2392 | 2392 | 2392 cities (Padberg/Rinaldi) | coordinates | 378063 |
| 22 | ftv33 | 34 | Asymmetric TSP (Fischetti) | distance matrix | 1286 |
| 23 | ftv35 | 36 | Asymmetric TSP (Fischetti) | distance matrix | 1473 |

### 3.4.2 Real locations with latitude-longitude coordinates

An example of the Central department stores' geographical latitude-longitude coordinates collected from Google Maps is shown in Table 3.2. The Central Group has 11 branches in Bangkok and 4 branches up-country. The list of latitude-longitude coordinates of other locations used in this study is shown in Appendix A.

**Table 3.2 Latitude-longitude coordinates of Central department stores**

| No. | Branch Name | Latitude | Longitude |
|---|---|---|---|
| 1 | Central World | 13.747128 | 100.538987 |
| 2 | Central Chidlom | 13.744492 | 100.544372 |
| 3 | Central Silom Complex | 13.728801 | 100.535309 |
| 4 | Central Chaengwattana | 13.904180 | 100.528100 |
| 5 | Central Lardprao | 13.816645 | 100.561423 |
| 6 | Central Ramindra | 13.871856 | 100.601740 |
| 7 | Central Bangna | 13.669087 | 100.633830 |
| 8 | Central Praram II | 13.663632 | 100.439750 |
| 9 | Central Praram III | 13.697853 | 100.538115 |
| 10 | Central Pinklao | 13.777943 | 100.476132 |
| 11 | Central Future Park Rangsit | 13.987610 | 100.618442 |
| 12 | Central Pattaya Beach, Chonburi | 12.933797 | 100.882425 |
| 13 | Central Kad Suan Kaew, Chiangmai | 18.796168 | 98.975793 |
| 14 | Central Had-Yai, Songkla | 7.006025 | 100.470993 |
| 15 | Central Phuket, Phuket | 7.890902 | 98.367631 |

Source: Google Maps (August, 2010)

There is more than one way to obtain latitude-longitude coordinates of a place based on the current version of Google Maps. A general direct way is illustrated here with “Central World” as an example. It is case-insensitive in Google Maps, so typing “Central World” or “central world” makes no difference. A list of locations appears on the left panel and a map is on the right. One can zoom in the map as preferred and spot the place as desired. Figure 3.1 depicts the right click at “Central World” and “What’s here?” is selected.

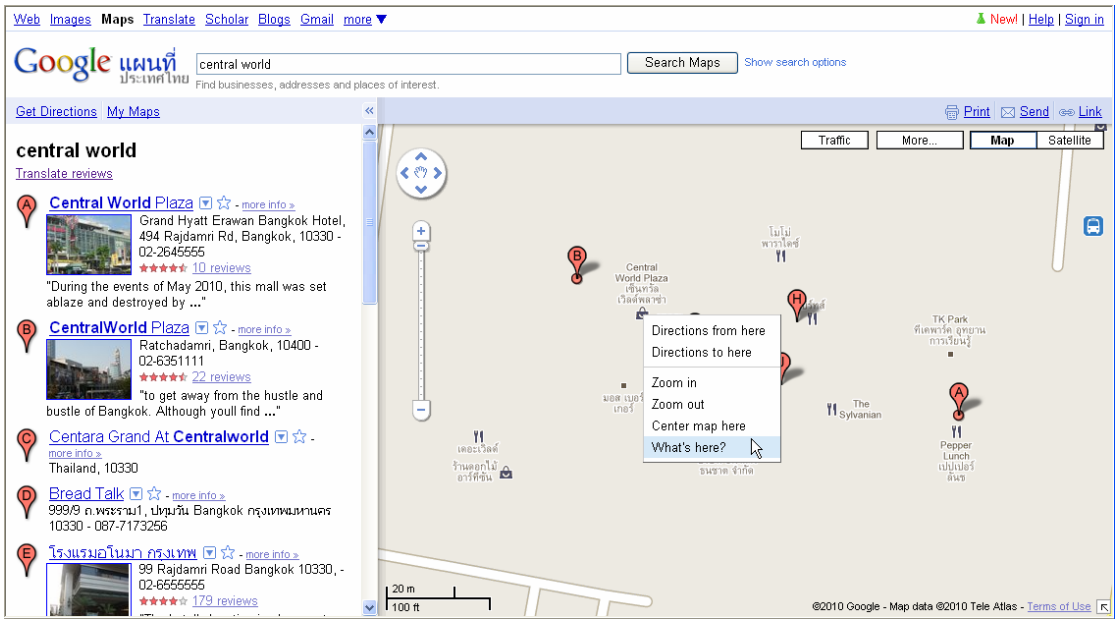


**Figure 3.1 A way to obtain the coordinates of "Central World" in Map view**

The coordinate of the place will be shown in the search box as in Figure 3.2. The latitude-longitude coordinate of "Central World" is obtained as (13.747128, 100.538987). Alternatively, holding the mouse hovered around the place, one can also see the coordinates in the hover tips box. If a location selected is large, the latitude-longitude coordinate can be slightly different depending on which point one clicks on the map. It does not make a significant difference. If it is not obvious spotting the place on the map, changing to Satellite view may help as in Figure 3.3.

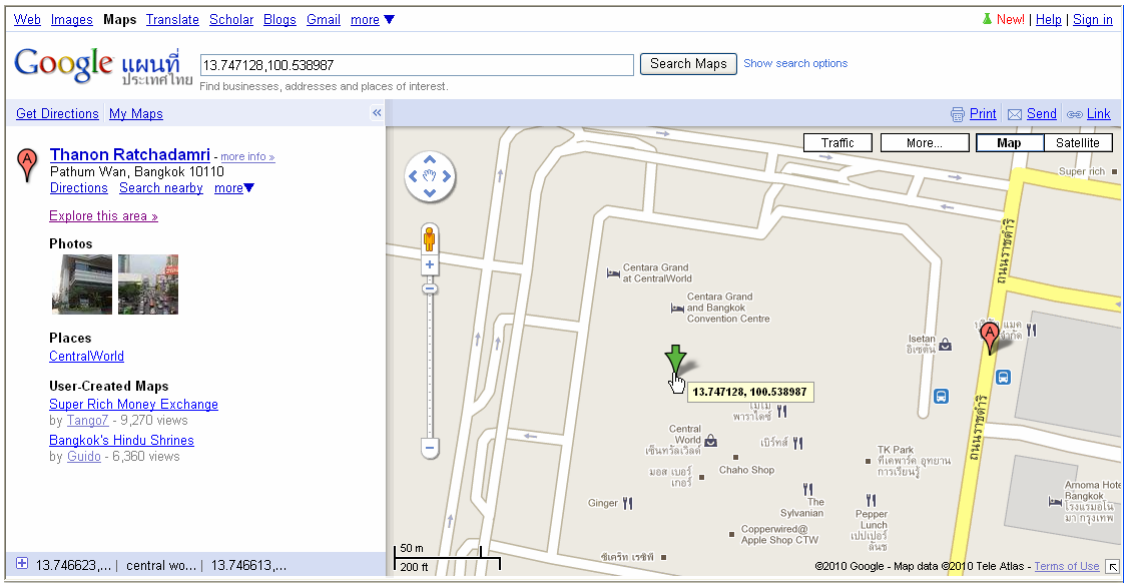


**Figure 3.2 Latitude-longitude coordinates of "Central World" in Map view**

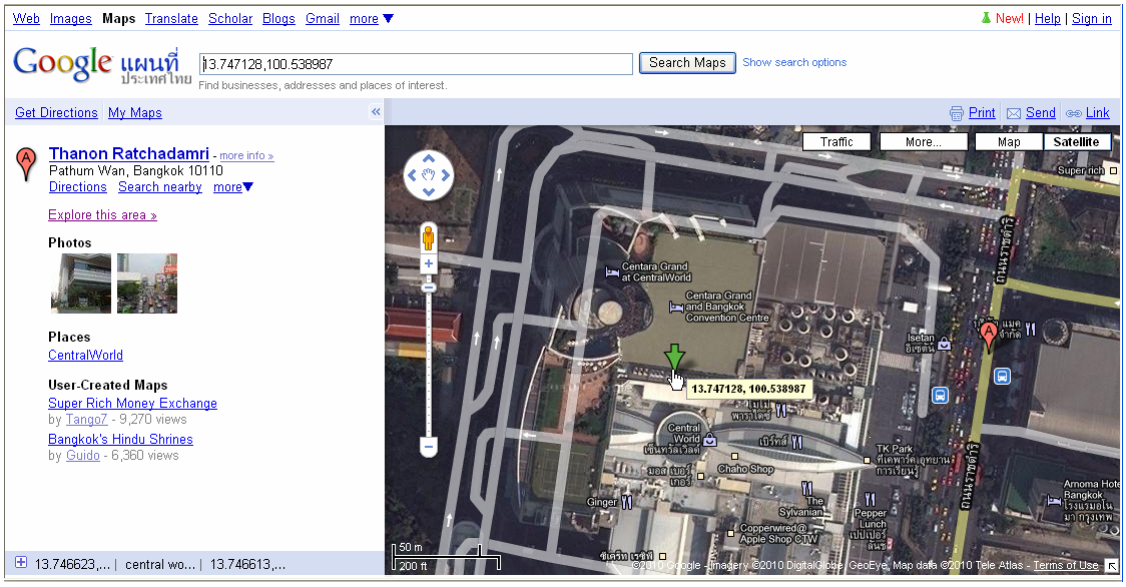


**Figure 3.3 Latitude-longitude coordinates of "Central World" in Satellite view**

For real application, it is often the case a user needs to input the places which are not readily available in Google Maps. To locate a place, the words input in the search box should be names of Soi, Road, District, Province, and Postcode. Searching an address in Thailand with Thai characters happens to take less time. Figure 3.4 shows that the address in Thai characters is typed in and its latitude-longitude coordinate is obtained in Figure 3.5.

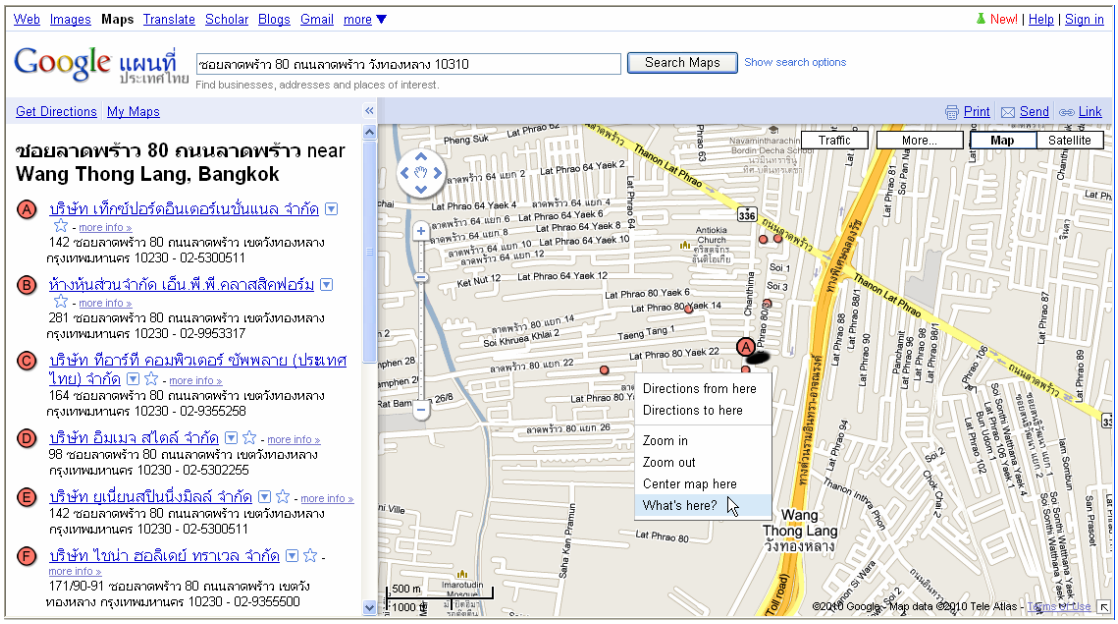


**Figure 3.4 Searching with Soi, Road and District names**

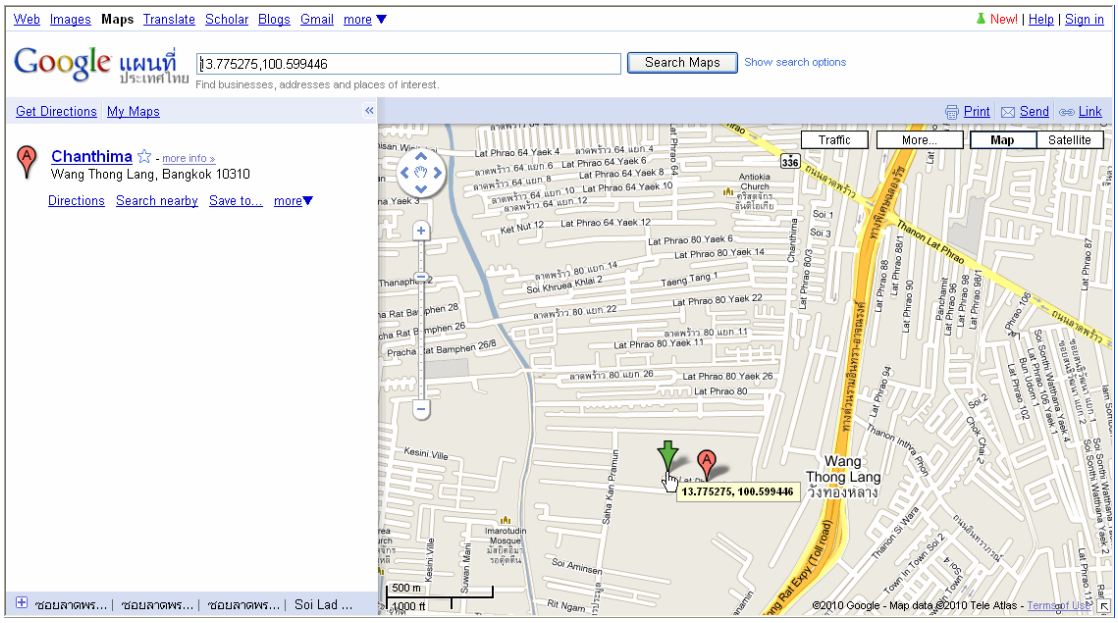


**Figure 3.5 Latitude-longitude coordinates of a search result in Map view**

### 3.4.3 Real locations with pair-wise distance matrix

Instead of using latitude-longitude coordinates and then calculate the distance based on Euclidean or Great Circle formulas, the distance between two locations or places can be found directly from Google Maps. The distance obtained from this approach represents the real commuting distance. An illustration is shown

here with an instance of “Central World” and “Central Chidlom”. To traverse from “Central Chidlom” with geographical coordinates (13.744492, 100.544372) to “Central World” with geographical coordinates (13.747128, 100.538987) by car, Google Maps suggests two directions depicted in Figure 3.6. The first suggested route mainly takes Rama I Road. Starting from “Central Chidlom”, head south on Soi Chidlom for 60 m, take the right turn onto Phloen Chit Road, and go straight on the road 550 m. Then, turn right to “Central World” and follow the way inside for another 312 m. The total distance of this suggested route is estimated around 950 m.

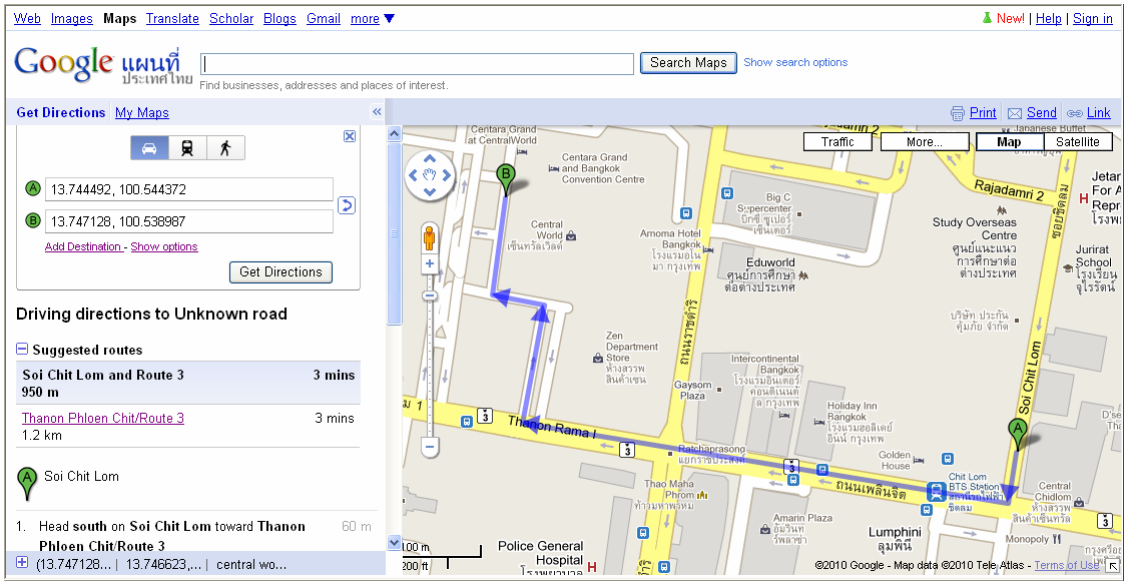


**Figure 3.6 The first suggested route from “Central Chidlom” to “Central World”**

The other suggested route is to take both Rama I Road and Ratchadamri Road as depicted in Figure 3.7. Starting also from “Central Chidlom”, this route suggests to take the right turn onto Phloen Chit Road and to take the right turn again onto Ratchadamri Road. After going straight on Ratchadamri Road for 400 m, the route suggests to turn left and then right to enter “Central World” at another entrance. This total distance of this route is 1.2 km. In most cases, there is more than one suggested route. The distance of the first suggested route is collected for this study.

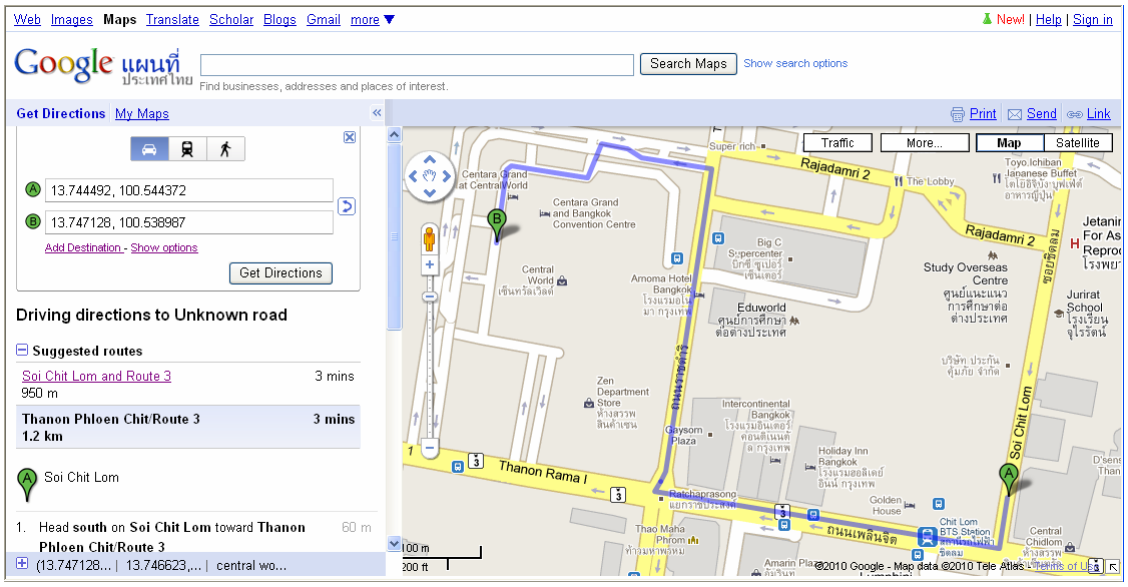


**Figure 3.7 The second suggested route from "Central Chidlom" to "Central World"**

If one chooses to go from "Central World" to "Central Chidlom" instead, Google Maps suggests three routes with the first suggested route depicted in Figure 3.8. The total distance data collected for this study is 1.1 km. The total distance collected from every pair of branches of 11 Central department stores in Bangkok is shown in Table 3.3.

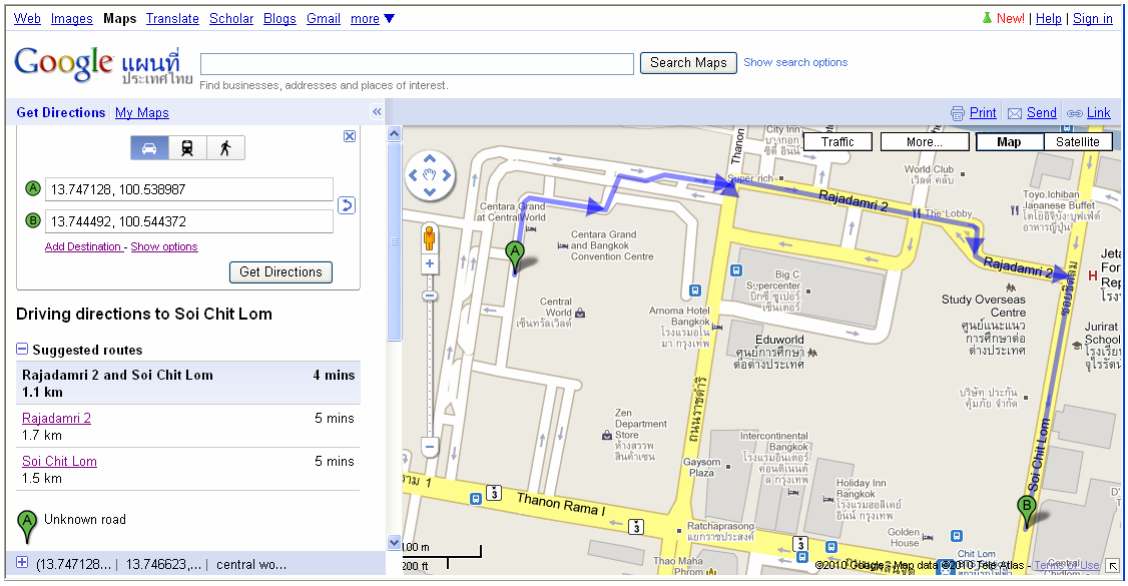


**Figure 3.8 Another direction from "Central World" to "Central Chidlom"**

**Table 3.3 Distance matrix of 11 Central department stores in Bangkok**

| Branch Name | World | Chidlom | Silom Complex | Chaengwattana | Lardprao | Ramindra | Bangna | Praram II | Praram III | Pinklao | Future Park |
|---|---|---|---|---|---|---|---|---|---|---|---|
| World | 0.0 | 1.1 | 2.7 | 21.9 | 10.3 | 21.9 | 17.4 | 20.9 | 9.8 | 11.2 | 31.1 |
| Chidlom | 1.0 | 0.0 | 2.4 | 22.3 | 10.7 | 18.6 | 16.5 | 21.9 | 9.4 | 11.6 | 31.6 |
| Silom | 5.1 | 5.4 | 0.0 | 25.0 | 14.3 | 14.5 | 15.0 | 18.2 | 7.1 | 15.1 | 36.1 |
| Chaengwattana | 24.1 | 24.4 | 24.0 | 0.0 | 14.1 | 9.3 | 36.8 | 39.4 | 28.3 | 20.6 | 21.6 |
| Lardprao | 9.9 | 11.2 | 13.5 | 15.3 | 0.0 | 9.2 | 23.6 | 29.0 | 16.5 | 14.4 | 22.9 |
| Ramindra | 21.5 | 21.8 | 24.1 | 9.3 | 11.3 | 0.0 | 30.6 | 39.6 | 27.1 | 20.6 | 15.9 |
| Bangna | 22.9 | 21.9 | 19.9 | 39.4 | 29.7 | 38.6 | 0.0 | 33.5 | 21.1 | 29.2 | 60.0 |
| Praram II | 21.7 | 20.7 | 18.7 | 38.2 | 28.5 | 40.9 | 27.4 | 0.0 | 12.9 | 17.7 | 50.0 |
| Praram III | 8.3 | 7.8 | 4.7 | 29.1 | 15.6 | 27.9 | 14.4 | 14.6 | 0.0 | 16.1 | 37.1 |
| Pinklao | 14.1 | 14.3 | 13.1 | 22.3 | 16.8 | 22.8 | 29.8 | 27.3 | 19.4 | 0.0 | 35.0 |
| Future Park | 32.6 | 33.0 | 35.3 | 21.7 | 22.5 | 15.4 | 45.4 | 50.8 | 38.3 | 33.3 | 0.0 |

This study includes the comparison of heuristics implementation between the true distance from Google Maps and the estimated distance from the geographical formula i.e. Great Circle method. Note that the true distance matrix can be asymmetrical while the estimated distance is symmetrical.

This data set includes the subset of branches of the three selected department stores, namely Central group, The Mall Group, and Robinson. The branches included for each instance are shown in Table 3.4.

**Table 3.4 Examples of TSP instances from real locations**

| No. | TSP Instance | | Locations/Branches |
|---|---|---|---|
| 1 | central7_1 | lat-long coordinates | Central: Central World, Chidlom, Silom Complex, Chaengwattana, Lardprao, Ramindra, Bangna |
| 2 | acentral7_1 | distance matrix | |
| 3 | central7_2 | lat-long coordinates | Central: Chaengwattana, Lardprao, Ramindra, Bangna, Praram II, Praram III, Pinklao |
| 4 | acentral7_2 | distance matrix | |
| 5 | central7_3 | lat-long coordinates | Central : Central World, Ramindra, Bangna, Praram II, Praram III, Pinklao, Future Park Rangsit |
| 6 | acentral7_3 | distance matrix | |
| 7 | central7_4 | lat-long coordinates | Central: Central World, Chidlom, Silom Complex, Lardprao, Praram II, Praram III, Pinklao |
| 8 | acentral7_4 | distance matrix | |
| 5 | themall6 | lat-long coordinates | The Mall Group: Ramkhamhaeng, Tha Pra, Ngamwongwarn, Bang Kae, Bangkapi, Nakorn Ratchasima |
| 6 | themall7 | lat-long coordinates | The Mall Group: Ramkhamhaeng, Tha Pra, Ngamwongwarn, Bang Kae, Bangkapi, The Emporium, Siam Paragon |
| 7 | themall8 | lat-long coordinates | The Mall Group: Ramkhamhaeng, Tha Pra, Ngamwongwarn, Bang Kae, Bangkapi, The Emporium, Siam Paragon |
| 8 | robinson9 | lat-long coordinates | Robinson: Ratchada, Sukhumvit, Bangrak, Bangkae, Srinakarin, Rangsit, Ramindra, Lad Ya, Ratanatibet |
| 9 | central11 | lat-long coordinates | Central: Central World, Chidlom, Silom Complex, Chaengwattana, Lardprao, Ramindra, Bangna, Praram II, Praram III, Pinklao, Future Park Rangsit |
| 10 | acentral11 | distance matrix | |

## 3.5 Tool and Platform Selection

A number of approaches either exact methods or heuristics have been developed to solve TSP. In reality; however, TSP can be practically solved by a computer program or application. Hence, the developers need to apply those methods or heuristics to the program which depends upon their discretion and expertise. TSP solver tools have been developed in a wide range of platforms and computer languages. For this study, MATLAB and Visual Basic language from the Visual Studio 2008 package are selected.

MATLAB is an interactive numerical computing environment. It provides easy, interactive environment as well as fast numerical algorithms. Microsoft Visual Studio is an Integrated Development Environment (IDE) from Microsoft to develop Graphical User Interface (GUI) along with web-based applications. Visual Basic (VB) is the third-generation event-driven programming language and IDE from Microsoft. VB is for Window-based users but with its capability to convert into executable files,

it can be used in other platforms. The programs developed are run on Mac OS Bootcamp with Microsoft Windows XP as follows.

System: Microsoft Windows XP Professional Version 2002 (Service Pack 3)

Computer: Intel Core2 Duo CPU, 2.25GHz, 1.72GB of RAM

For the three major operating systems used by computer users around the world, Windows has the highest user base, followed by Linux and Mac. The information from netmarketshare.com as of August 2010 indicated that Windows OS is far more common used than Mac and others such as Java ME, Linux, iPhone, iPad, Symbian, iPod, Android, Blackberry. The Operating System Market Share in Thailand as of April 2010 includes Windows 98.48% Mac 1.33% comparing to the Operating System Market Share Worldwide as of April 2010 includes 91.46% Mac 5.32% (Tanthawichian P., 2010). This information supports the idea of using Visual Basic as an alternative as a selected tool with user interface.

In this study, five different algorithms from the three classic construction heuristics are implemented in two applications i.e. MATLAB R2010b and Visual Studio 2008. These five algorithms include Nearest Neighbor (NN), Nearest Insertion (NI), Farthest Insertion 1 (FI1), Farthest Insertion 2 (FI2), and Farthest Insertion 3 (FI3). The three Farthest Insertion algorithms are different on the selection step. The algorithms can be developed in any programs upon the preferences and backgrounds of the users and developers. For this study, all empirical tests are conducted in MATLAB R2010b environment since the elapsed time is quite consistent. However, the program developed in Visual Basic 2008 is designed for general users since the application is more user-friendly with graphical user interface such as buttons, checkboxes, radio buttons.

Both applications require distance for every pair of locations as input which can be either one of the two formats i.e. a pre-determined pair-wise distance matrix or a set of coordinates. If a set of coordinates is input, two options of distance calculation are available i.e. Great Circle and Euclidean methods. Euclidean method is used for planar coordinates specifying positions in a plane. On the other hand, Great Circle method shall be applied to convert latitude-longitude coordinates into distance in the unit of kilometers.

### 3.5.1 MATLAB R2010b

The code of each algorithm is in an m-file, or script file, which is a simple text file where MATLAB commands are placed. The constructed m-files are also tested in MATLAB 2008a and MATLAB 7 environment and should also work with earlier versions. MATLAB R2010b user interface is displayed in Figure 3.9.

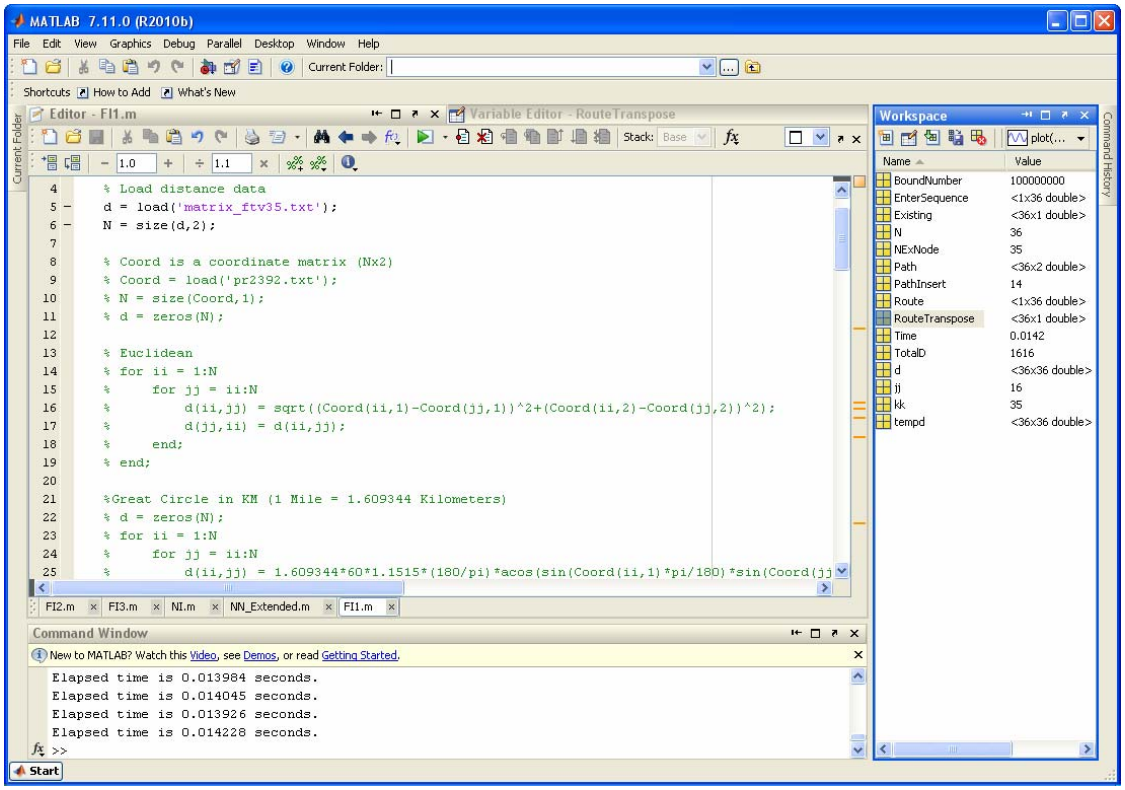


**Figure 3.9 MATLAB R2010b user interface**

One simple way is using the ‘load’ command to read the contents of the file but it requires that the data in the file are organized in an array and contain no text headings or column labels. Since the distance input file is simply either an array of distances or an array of coordinates, an input file in text (.txt) format is used and a ‘load’ command is thus applied. To get around this restriction, one can use file input/output commands. To further simplify the step, an input file is saved under the same path as m-files. Since the data file has no column and row labels, a tab or space can be used as a delimiter. Figure 3.10 illustrates the input text files of distance matrix and a set of coordinates.

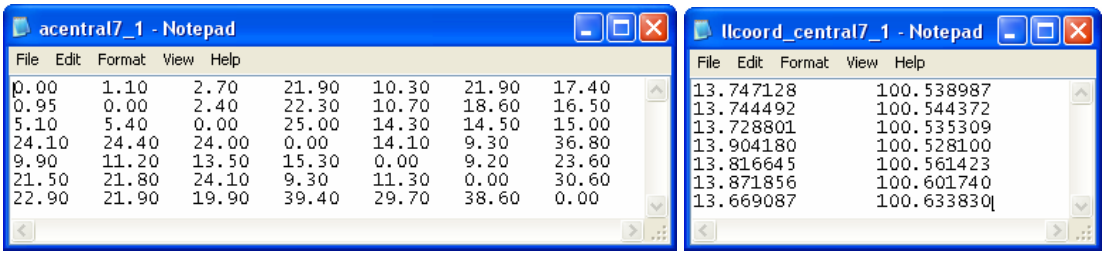


**Figure 3.10 Examples of the input text file format for MATLAB**

If the input text file is a set of coordinates, users need to select either Euclidean or Great Circle method to calculate distance by This is simply done by commenting the portion of codes that is not applicable. The first data in an input file is viewed as the node number “1”, the next one is number “2”, and so on. When the file is ready to run, users can click “Run” button; or, select “Debug” in Menu bar and then “Run” as illustrated in Figure 3.11. The tour constructed will be kept in a row array in “Route” variable or a column array in “RouteTranspose” variable. The associated total distance of the constructed tour is kept in “TotalD” variable.

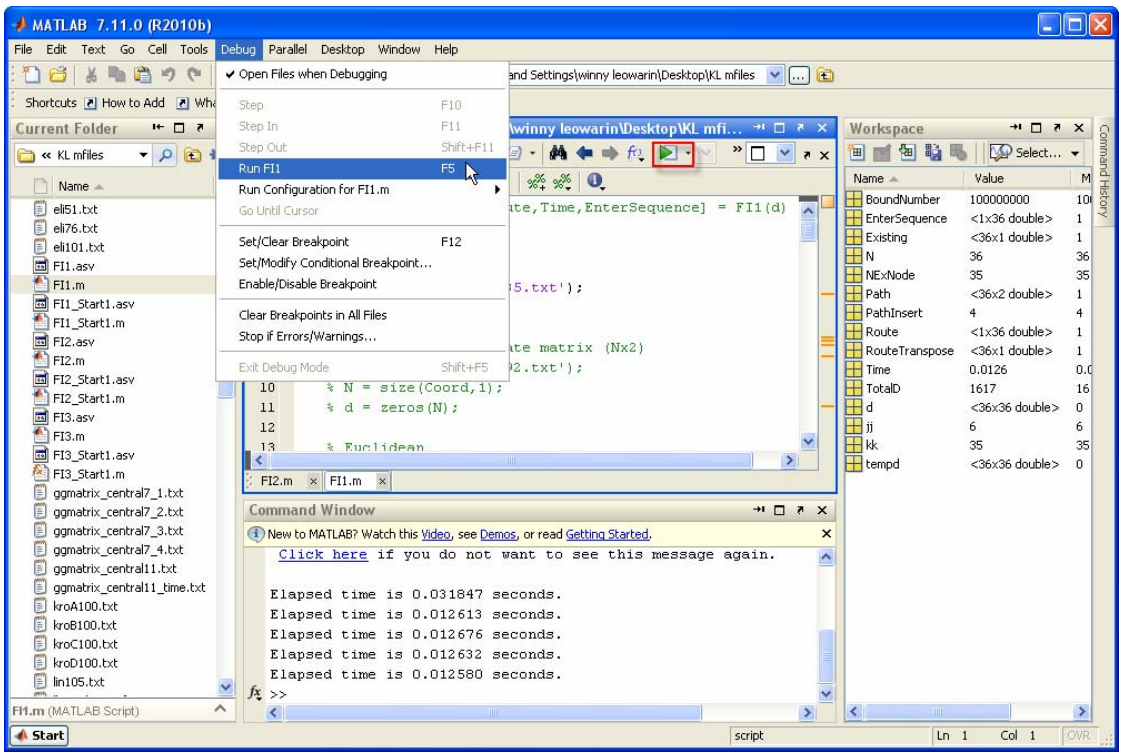


**Figure 3.11 Running an m-file to obtain the “Route”**

### 3.5.2 Visual Studio 2008

It is required to input the distance or coordinates files in Excel 2003 (.xls) format under the folder “bin” of the Visual Studio software package. After that, users only click each button to run the corresponding algorithm, or even click one button of “Run All” to run all algorithms and see the results shown in the textbox. Figure 3.12 and 3.13 show the user interface in Visual Studio 2008 when “Run FI3” and “Run All” buttons are clicked, respectively.

A user has two options for data input i.e. a pair-wise distance table or a set of latitude-longitude coordinates. The input file of latitude-longitude coordinates is designed in Excel 2003 format with the worksheet named “Location”. The coordinates will be translated to the distances between each pair of the locations with either one of the two methods. Also, the input file of distance matrix can be input instead with the worksheet named “Distance”. The files of latitude-longitude coordinates or distance matrix can be browsed from other folders but with the default sheet name “Sheet1”. Excel is used instead of Access or other relational database management tool. This is just to simplify the process and should be easily applicable to small and medium businesses.

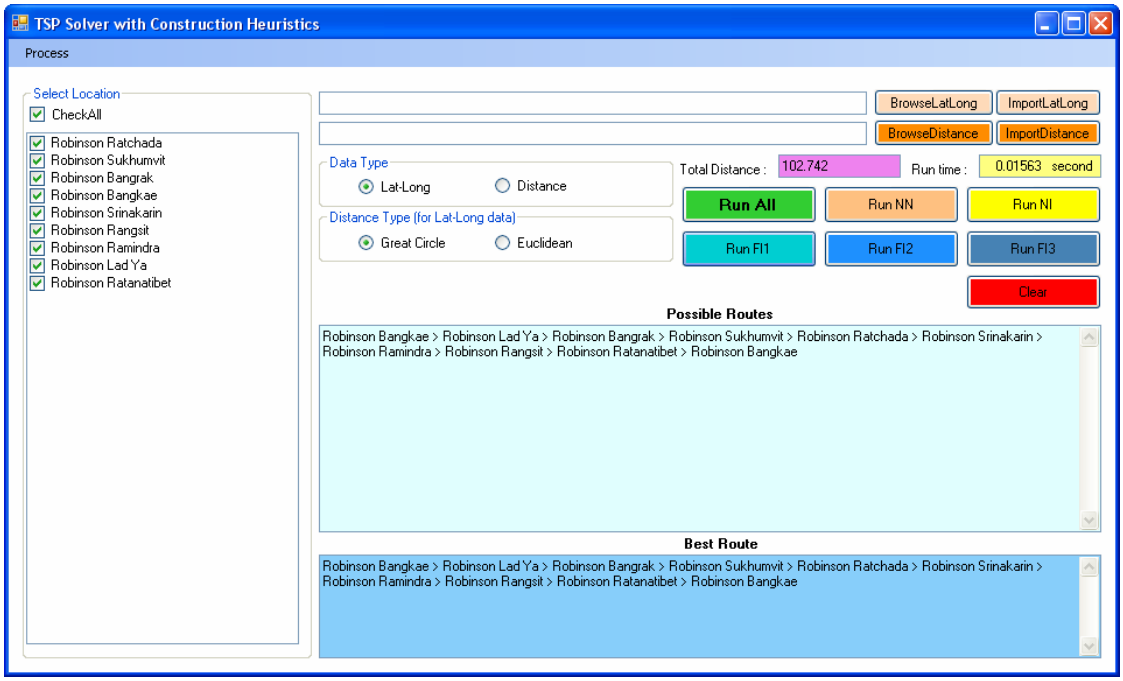


**Figure 3.12 Visual Studio 2008 user interface when “Run FI3” button is selected**

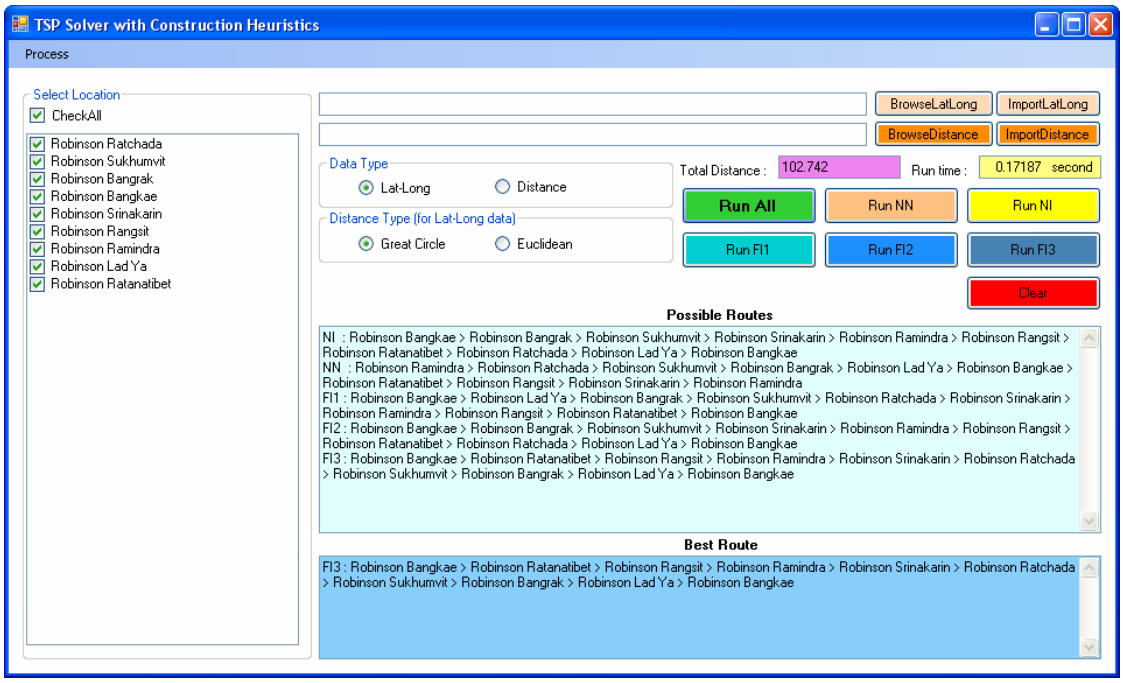


**Figure 3.13 Visual Studio 2008 user interface when "Run All" button is selected**

# CHAPTER IV
# RESULT AND DISCUSSION

## 4.1 Result Analysis

To compare the performance of the five algorithms i.e. NN, NI, FI1, FI2, and FI3, the empirical tests are conducted under MATLAB R2010b environment on Windows XP platform. Two types of experiments are conducted. The first one is the experiment to compare each of the heuristic solutions with the optimal solution obtained either from TSPLIB or other available applications i.e. Excel Solver 2003 and LINGO7. The second experiment is to compare solutions among these heuristics themselves with unknown optimal solutions available. In addition to the instances from TSPLIB, the real locations whose latitude-longitude coordinates obtained from Google Maps are also used in the tests. For each of the TSP instances, five runs are performed and the average elapsed time of each algorithm is recorded.

The first experiment is conducted by solving 23 selected TSP instances from TSPLIB with five algorithms. This test covers the instances ranging from 17 to 2392 places. The distance input of these TSP instances has one of the two formats i.e. a pair-wise distance matrix or a set of planar coordinates. The results of solution quality measured by the deviation from the optimal solutions, in percentage rounded to the nearest integer, are shown in Table 4.1. The first 21 TSP instances are symmetrical and the last two instances are asymmetrical. The optimal tours are known with their costs shown in the fourth column. The heuristic(s) giving the best solution among all heuristics are also displayed. Note that for those TSPs of which coordinates input are provided instead of the explicit distance matrix i.e. No. 8 – 21, Euclidean method is used to calculate the distance between any two nodes.

Considering the results in Table 4.1 in terms of solution quality, FI1 performs the best and FI2 comes the second. With the number of instances less than a hundred, FI1 gives the result within 10% of the optimal solution most of the times. For fri26 problem, FI1 even gives the optimal solution. NI is the least preferable choice on

average. Surprisingly, when the TSP instance becomes larger, saying more than a hundred nodes, FI2 and FI3 give similar solution quality to NN and NI. For the most part, Farthest Insertion algorithms outperform the Nearest Insertion and Nearest Neighbor algorithms.

**Table 4.1 Deviation from optimal solution of 5 algorithms for TSPLIB instances**

| No. | TSPLIB | *n* | Optimal Solution | Deviation from Optimal Solution | | | | | | Heuristic(s) with Min |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | NN | NI | FI1 | FI2 | FI3 | Min | |
| 1 | gr17 | 17 | 2085 | 4% | 11% | 1% | 1% | 4% | 1% | FI1, FI2 |
| 2 | gr24 | 24 | 1272 | 22% | 8% | 7% | 6% | 10% | 6% | FI2 |
| 3 | fri26 | 26 | 937 | 3% | 12% | 0% | 3% | 11% | 0% | FI1 |
| 4 | bays29 | 29 | 2020 | 6% | 13% | 1% | 6% | 12% | 1% | FI1 |
| 5 | dantzig42 | 42 | 699 | 24% | 19% | 9% | 9% | 17% | 9% | FI1 |
| 6 | swiss42 | 42 | 1273 | 13% | 20% | 6% | 14% | 11% | 6% | FI1 |
| 7 | gr48 | 48 | 5046 | 16% | 8% | 8% | 4% | 16% | 4% | FI2 |
| 8 | eli51 | 51 | 430 | 18% | 15% | 4% | 14% | 8% | 4% | FI1 |
| 9 | berlin52 | 52 | 7544 | 8% | 20% | 8% | 3% | 21% | 3% | FI2 |
| 10 | st70 | 70 | 679 | 12% | 15% | 9% | 5% | 11% | 5% | FI2 |
| 11 | eli76 | 76 | 545 | 12% | 15% | 7% | 10% | 12% | 7% | FI1 |
| 12 | pr76 | 76 | 108159 | 21% | 19% | 11% | 7% | 10% | 7% | FI2 |
| 13 | rd100 | 100 | 7910 | 19% | 20% | 9% | 11% | 15% | 9% | FI1 |
| 14 | kroA100 | 100 | 21285 | 16% | 21% | 10% | 15% | 16% | 10% | FI1 |
| 15 | kroC100 | 100 | 20751 | 14% | 24% | 5% | 5% | 10% | 5% | FI2 |
| 16 | kroD100 | 100 | 21294 | 17% | 21% | 5% | 9% | 12% | 5% | FI1 |
| 17 | eli101 | 101 | 642 | 15% | 16% | 7% | 13% | 16% | 7% | FI1 |
| 18 | lin105 | 105 | 14383 | 18% | 29% | 7% | 9% | 16% | 7% | FI1 |
| 19 | tsp225 | 225 | 3859 | 20% | 24% | 10% | 16% | 18% | 10% | FI1 |
| 20 | pcb442 | 442 | 50784 | 16% | 17% | 12% | 18% | 21% | 12% | FI1 |
| 21 | pr2392 | 2392 | 378063 | 21% | 24% | 13% | 28% | 30% | 13% | FI1 |
| 22 | ftv33 | 34 | 1286 | 24% | 22% | 7% | 13% | 13% | 7% | FI1 |
| 23 | ftv35 | 36 | 1473 | 13% | 15% | 8% | 12% | 8% | 8% | FI1 |
| Average | | | | 15% | 18% | 7% | 10% | 14% | 7% | FI1 |

The other performance measure in this study is time efficiency which is not easy to evaluate. It can be depend on the implementation e.g. how well the code is written, hardware and platform used. However, all heuristics have been tested on the same machine, platform, and environment. Due to this reason, the computational times in second(s) shown in Table 4.1 could be used for comparison purpose against one another. The numbers may not be so significant but the ratios can be considerable.

**Table 4.2 Computational times in second(s) of 5 algorithms for TSPLIB instances**

| No. | TSPLIB | *n* | NN | NI | FI1 | FI2 | FI3 |
|---|---|---|---|---|---|---|---|
| 1 | gr17 | 17 | 0.007 | 0.012 | 0.011 | 0.010 | 0.010 |
| 2 | gr24 | 24 | 0.011 | 0.012 | 0.011 | 0.011 | 0.011 |
| 3 | fri26 | 26 | 0.012 | 0.012 | 0.012 | 0.012 | 0.011 |
| 4 | bays29 | 29 | 0.015 | 0.013 | 0.012 | 0.012 | 0.012 |
| 5 | dantzig42 | 42 | 0.029 | 0.016 | 0.014 | 0.014 | 0.014 |
| 6 | swiss42 | 42 | 0.029 | 0.015 | 0.014 | 0.014 | 0.015 |
| 7 | gr48 | 48 | 0.039 | 0.016 | 0.015 | 0.015 | 0.015 |
| 8 | eli51 | 51 | 0.042 | 0.017 | 0.016 | 0.016 | 0.016 |
| 9 | berlin52 | 52 | 0.045 | 0.017 | 0.016 | 0.016 | 0.016 |
| 10 | st70 | 70 | 0.082 | 0.022 | 0.021 | 0.020 | 0.021 |
| 11 | eli76 | 76 | 0.097 | 0.024 | 0.024 | 0.022 | 0.022 |
| 12 | pr76 | 76 | 0.095 | 0.024 | 0.023 | 0.023 | 0.022 |
| 13 | rd100 | 100 | 0.175 | 0.034 | 0.032 | 0.032 | 0.031 |
| 14 | kroA100 | 100 | 0.172 | 0.034 | 0.032 | 0.033 | 0.031 |
| 15 | kroC100 | 100 | 0.171 | 0.032 | 0.033 | 0.031 | 0.031 |
| 16 | kroD100 | 100 | 0.169 | 0.035 | 0.036 | 0.033 | 0.032 |
| 17 | eli101 | 101 | 0.174 | 0.034 | 0.032 | 0.032 | 0.033 |
| 18 | lin105 | 105 | 0.188 | 0.036 | 0.034 | 0.034 | 0.035 |
| 19 | tsp225 | 225 | 1.043 | 0.148 | 0.139 | 0.136 | 0.136 |
| 20 | pcb442 | 442 | 5.857 | 0.819 | 0.724 | 0.705 | 0.727 |
| 21 | pr2392 | 2392 | 535.582 | 99.203 | 105.858 | 93.364 | 93.894 |
| 22 | ftv33 | 34 | 0.020 | 0.014 | 0.013 | 0.013 | 0.013 |
| 23 | ftv35 | 36 | 0.022 | 0.014 | 0.014 | 0.014 | 0.013 |
| | Average | | 24.730 | 4.572 | 4.869 | 4.300 | 4.325 |

Taking into consideration all instances in Table 4.2, on average, FI2 and FI3 give the shortest running time, followed by NI, FI1, and NN, respectively. If excluding pr2392 TSP instance, all three Farthest Insertion algorithms have comparable computational times. Nearest Neighbor algorithm gives the small running time for the smaller size problem, saying less than 30 nodes, and the running time grows dramatically when the problem size becomes larger.

Another set of the first experiment type is comparing the solutions from each heuristic versus optimal for the small-sized selected instances. This data set includes the subset of branches of the three selected department stores, namely Central group, The Mall Group, and Robinson. These instances are called small-sized in this study because it is solvable in Excel Solver 2003 within reasonable time. The instance names starting with ‘a’ indicate that the distance input is asymmetrical distance matrix obtained from the first directions suggested in Google Maps. Other instances have

distance input in the format of latitude-longitude coordinates and Great Circle method is then applied. The solutions of these selected TSP instances are shown in Table 4.3.

**Table 4.3 Deviation from optimal solution of 5 algorithms for real locations**

| No. | TSP instance | *n* | Optimal Solution | Deviation from Optimal Solution | | | | | | Heuristic(s) with Min |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | | NN | NI | FI1 | FI2 | FI3 | Min | |
| 1 | central7_1 | 7 | 65.2 | 3% | 7% | 0% | 0% | 3% | 0% | FI1, FI2 |
| 2 | central7_2 | 7 | 85.0 | 0% | 0% | 0% | 0% | 0% | 0% | All |
| 3 | central7_3 | 7 | 105.2 | 5% | 2% | 2% | 2% | 0% | 0% | FI3 |
| 4 | central7_4 | 7 | 49.0 | 16% | 9% | 0% | 0% | 0% | 0% | FI1, FI3 |
| 5 | themall6 | 6 | 457.8 | 3% | 0% | 0% | 0% | 0% | 0% | NI, FI1, FI2, FI3 |
| 6 | themall7 | 7 | 61.7 | 0% | 10% | 0% | 0% | 0% | 0% | All |
| 7 | themall8 | 8 | 458.7 | 0% | 1% | 0% | 0% | 0% | 0% | NN, FI1, FI2, FI3 |
| 8 | robinson9 | 9 | 102.7 | 15% | 4% | 0% | 4% | 0% | 0% | FI1, FI3 |
| 9 | central11 | 11 | 107.8 | 12% | 3% | 6% | 7% | 0% | 0.2% | FI3 |
| 10 | acentral7_1 | 7 | 85.3 | 4% | 0% | 4% | 4% | 4% | 0% | NI |
| 11 | acentral7_2 | 7 | 119.9 | 5% | 3% | 0% | 0% | 0% | 0% | FI1, FI2, FI3 |
| 12 | acentral7_3 | 7 | 142.1 | 6% | 2% | 0% | 3% | 3% | 0% | FI1 |
| 13 | acentral7_4 | 7 | 69.6 | 0% | 0% | 0% | 0% | 11% | 0% | NN, NI, FI1, FI2 |
| 14 | acentral11 | 11 | 151.8 | 12% | 6% | 3% | 9% | 5% | 3.3% | FI1 |
| 15 | robinson13 | 13 | 3596.07 | 6% | 0% | 0% | 0% | 5% | 0% | NI, FI2 |
| Average | | | | 6% | 3% | 1% | 2% | 2% | 0% | FI1 |

The optimal solutions are obtained from LINGO 7 Student version for the TSP problems with up to 7 locations. For the instances No. 7 – 9 and 11 are solved optimally in Excel Solver 2003.

Generally speaking, for small TSP problems of less than 10 nodes, if all five algorithms are applied and the best route is selected, it is highly possible to obtain the optimal solution. It is clearly seen that the five heuristics are quite effective for small problems. To emphasize the effectiveness of the implementation of these construction heuristic algorithms for small problems, the running time results are also shown in Table 4.4.

**Table 4.4 Computational times in second(s) for 5 algorithms for real locations**

| No. | TSP instance | *n* | NN | NI | FI1 | FI2 | FI3 |
|---|---|---|---|---|---|---|---|
| 1 | central7_1 | 7 | 0.003 | 0.010 | 0.009 | 0.009 | 0.009 |
| 2 | central7_2 | 7 | 0.003 | 0.010 | 0.009 | 0.009 | 0.009 |
| 3 | central7_3 | 7 | 0.003 | 0.010 | 0.009 | 0.009 | 0.009 |
| 4 | central7_4 | 7 | 0.003 | 0.010 | 0.009 | 0.009 | 0.009 |
| 5 | themall6 | 6 | 0.003 | 0.010 | 0.012 | 0.012 | 0.012 |
| 6 | themall7 | 7 | 0.004 | 0.010 | 0.012 | 0.012 | 0.012 |
| 7 | themall8 | 8 | 0.004 | 0.010 | 0.012 | 0.012 | 0.013 |
| 8 | robinson9 | 9 | 0.004 | 0.011 | 0.010 | 0.009 | 0.010 |
| 9 | central11 | 11 | 0.004 | 0.011 | 0.010 | 0.010 | 0.010 |
| 10 | acentral7_1 | 7 | 0.003 | 0.011 | 0.010 | 0.010 | 0.010 |
| 11 | acentral7_2 | 7 | 0.003 | 0.011 | 0.010 | 0.010 | 0.010 |
| 12 | acentral7_3 | 7 | 0.003 | 0.011 | 0.010 | 0.010 | 0.010 |
| 13 | acentral7_4 | 7 | 0.003 | 0.011 | 0.010 | 0.010 | 0.010 |
| 14 | acentral11 | 11 | 0.004 | 0.011 | 0.010 | 0.010 | 0.010 |
| 15 | robinson13 | 13 | 0.006 | 0.011 | 0.014 | 0.013 | 0.013 |

Comparing among five algorithms, insertion algorithms are not significantly different from one another. Although NN takes less time than all insertions algorithms about three times, it is so small and can then be neglected. Computational times are less than 1 second as well in Visual Studio 2008 application.

The second experiment is to run the real TSP instances of more than 13 nodes with results shown in Table 4.5. This data set includes the subset of branches of four selected department stores and hypermarkets i.e. Central group, Robinson, Carrefour and Tesco Lotus. The distance input for these instances are latitude-longitude coordinates with Great Circle as distance calculation method.

**Table 4.5 Solutions of 5 algorithms for selected TSP instances**

| No. | TSP instance | *n* | NN | NI | FI1 | FI2 | FI3 | Heuristic(s) with Min |
|---|---|---|---|---|---|---|---|---|
| 1 | central15 | 15 | 2909.9 | 2904.1 | 2859.8 | 2859.9 | 2860.1 | FI1, FI2 |
| 2 | robinson22 | 22 | 3885.3 | 3976.3 | 3666.4 | 3665.2 | 3852.1 | FI2 |
| 3 | carrefour27 | 27 | 218.3 | 214.6 | 206.1 | 201.4 | 221.9 | FI2 |
| 4 | tesco37 | 37 | 300.5 | 301.9 | 283.8 | 299.1 | 281.2 | FI3 |
| 5 | tesco70 | 70 | 4897.5 | 4731.5 | 4531.8 | 4689.2 | 4734.4 | FI1 |

For the TSP instances in Table 4.5, FI1 and FI2 generally perform better than others. FI1 performs consistently well on average. The time taken for each TSP

instance is shown in Table 4.6. Farthest Insertions take more time but gain a better result in general.

**Table 4.6 Computational times in second(s) of 5 algorithms for selected TSP instances**

| No. | TSP instance | *n* | NN | NI | FI1 | FI2 | FI3 |
|---|---|---|---|---|---|---|---|
| 1 | central15 | 15 | 0.006 | 0.011 | 0.014 | 0.013 | 0.013 |
| 2 | robinson22 | 22 | 0.012 | 0.012 | 0.015 | 0.016 | 0.015 |
| 3 | carrefour27 | 27 | 0.017 | 0.013 | 0.018 | 0.017 | 0.017 |
| 4 | tesco37 | 37 | 0.024 | 0.014 | 0.013 | 0.013 | 0.013 |
| 5 | tesco70 | 70 | 0.122 | 0.022 | 0.048 | 0.046 | 0.047 |

Another test is to review the effectiveness of the Great Circle method for the true application. For the 11 Central department stores in Bangkok, the average road distance from Google Maps is around 41% greater than the distance obtained from the Great Circle formula. For the 13 Robinson department stores up-country, the road distance from Google Maps is around 34% greater than the distance from the Great Circle formula. It appears that the formula used to calculate distance is good for the set of locations that is sparse than the dense one.

Table 4.7 shows an example of the tour solution for 11 Central department stores with Great Circle formula calculation comparing with the true distance matrix obtained from the first suggest route from Google Maps. Without loss of generality, the names of the Central department stores are replaced by the numbers 1 to 11.

**Table 4.7 Solution comparison with the distance from Great Circle formula versus the road distance from Google Maps**

| | Distance calculation | Tour Solution for "central11" and "acentral11" |
|---|---|---|
| Optimal | Great Circle formula | 1 > 2 > 3 > 7 > 9 > 8 > 10 > 4 > 11 > 6 > 5 > 1 |
| | True distance | 1 > 2 > 3 > 7 > 9 > 8 > 10 > 4 > 6 > 11 > 5 > 1 |
| NN | Great Circle formula | 1 > 2 > 3 > 9 > 7 > 5 > 6 > 4 > 11 > 10 > 8 > 1 |
| | True distance | 1 > 2 > 9 > 8 > 10 > 5 > 6 > 4 > 11 > 7 > 3 > 1 |
| NI | Great Circle formula | 1 > 5 > 6 > 11 > 4 > 10 > 8 > 3 > 9 > 7 > 2 > 1 |
| | True distance | 1 > 2 > 9 > 8 > 10 > 5 > 6 > 4 > 11 > 7 > 3 > 1 |
| FI1 | Great Circle formula | 1 > 2 > 3 > 10 > 8 > 9 > 7 > 6 > 11 > 4 > 5 > 1 |
| | True distance | 1 > 2 > 7 > 3 > 9 > 8 > 10 > 4 > 6 > 11 > 5 > 1 |
| FI2 | Great Circle formula | 1 > 2 > 3 > 8 > 9 > 7 > 10 > 4 > 11 > 6 > 5 > 1 |
| | True distance | 1 > 2 > 3 > 10 > 8 > 9 > 7 > 6 > 11 > 4 > 5 > 1 |
| FI3 | Great Circle formula | 1 > 2 > 5 > 6 > 11 > 4 > 10 > 8 > 9 > 7 > 3 > 1 |
| | True distance | 1 > 5 > 6 > 11 > 4 > 10 > 3 > 8 > 9 > 7 > 2 > 1 |

## 4.2 Alternative Applications

In this study, two more applications are explored for solving TSP optimally i.e. Excel Solver and LINGO. Solver is the optimization software of Frontline Systems developed for Microsoft including Excel Solver and Premium Solver which is a basic upgrade for the standard Excel Solver. Excel Solver can solve up to 200 variables while Premium Solver can solve up to 2000 variables at speeds anywhere from three to 100 times faster than the standard Solver. Premium Solver is quoted at the price of $895 including software license $745 and annual support $150. However, this study applies the Premium Solver Student version which allows 200 variables limit. LINGO is also an optimization software of LINDO Systems, Inc. This study applies LINGO 7.0 Student version that allows 50 integer variables. The LINGO's current demo version available for free trial download, LINGO12, can solve up to 30 integer variables.

With the TSP formulation in Chapter 2, Excel Solver can solve up to 13 locations. The real TSP instances up to 13 locations are solved by Solver and the average solving time in seconds is shown in Table 4.8.

**Table 4.8 Average computational times in seconds (s), rounded to the nearest integer, of Excel Solver and Premium Solver for selected TSP instances**

| No. | TSP instance | *n* | Excel Solver | Premium Solver |
|---|---|---|---|---|
| 1 | themall6 | 6 | 5 | 2 |
| 2 | central7_1 | 7 | 44 | 17 |
| 3 | central7_2 | 7 | 13 | 6 |
| 4 | central7_3 | 7 | 65 | 23 |
| 5 | central7_4 | 7 | 4 | 2 |
| 6 | acentral7_1 | 7 | 40 | 10 |
| 7 | acentral7_2 | 7 | 9 | 7 |
| 8 | acentral7_3 | 7 | 60 | 19 |
| 9 | acentral7_4 | 7 | 3 | 2 |
| 10 | themall7 | 7 | 14 | 3 |
| 11 | themall8 | 8 | 17 | 4 |
| 12 | robinson9 | 9 | 57 | 158 |
| 13 | central11 | 11 | 622 | 1650 |
| 14 | acentral11 | 11 | 29 | 1500 |
| 15 | robinson13 | 13 | > 82800 | > 46800 |

Premium Solver solves faster for the TSP that has less than 8 locations. The experiment is run on “central11” problem, and Excel Solver can give the optimal solution but at cost of much time consuming. The attempt to solve 13 locations was done with “robinson13” TSP instance, but Excel Solver appears to become inefficient. For those TSP instances of less than 8 locations, LINGO can solve for optimal solutions in no time, or less than 0.0 second. It is interesting observing the time required to solve TSP optimally in Excel Solver. For example, solving “robinson9” TSP instance, Excel Solver takes 29 seconds on average to obtain the optimal solution of 102.7. On the same machine, FI1 and FI3 algorithms also give the same solution within less than a second (in both MATLAB and Visual Studio). Figures 4.1 and 4.2 show the Excel Solver interface on solving the TSP. To ensure the optimal solution for integer programming problems, set “Tolerance” to 0% in “Options” button in the “Solver Parameters” box.

Excel Solver takes approximately 15 hours to obtain an incumbent, a solution satisfying the integer constraints, of 3596.07, on “robinson13” instance. It takes at least 23 hours in total before ending the solving algorithms and confirms 3596.07 as the optimal solution. Again, it takes less than a second to get the same solution by NI and FI2.

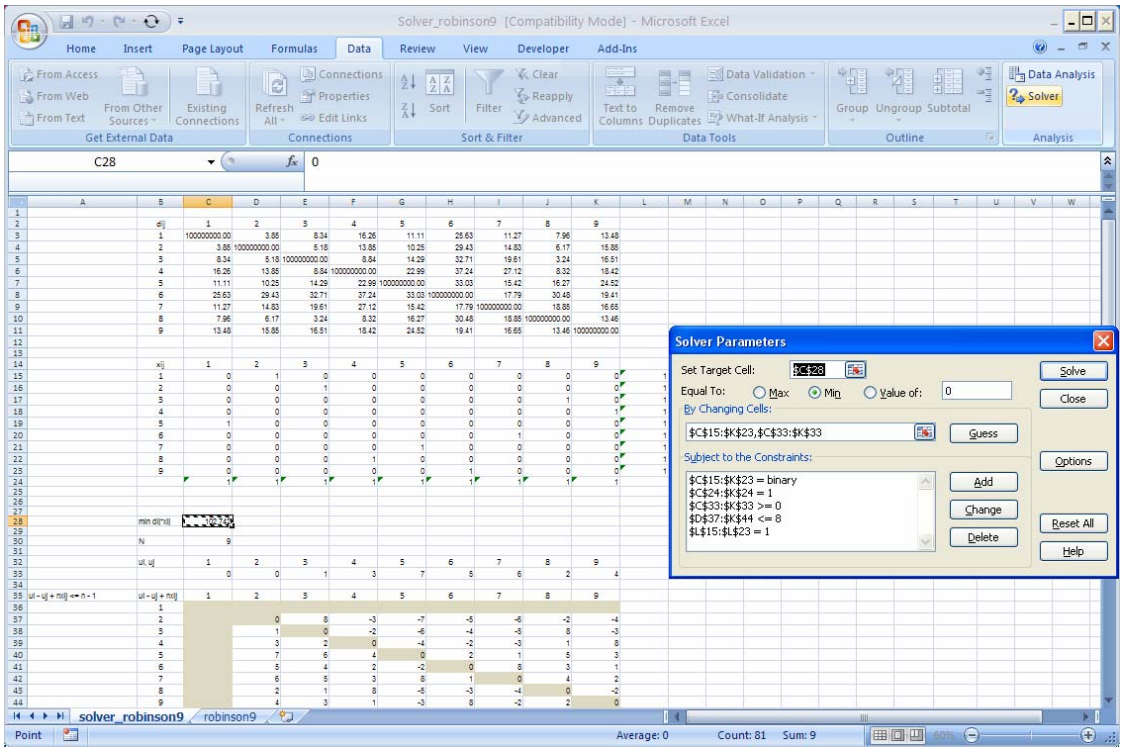


**Figure 4.1 Running Excel Solver on "robinson9" TSP instance**

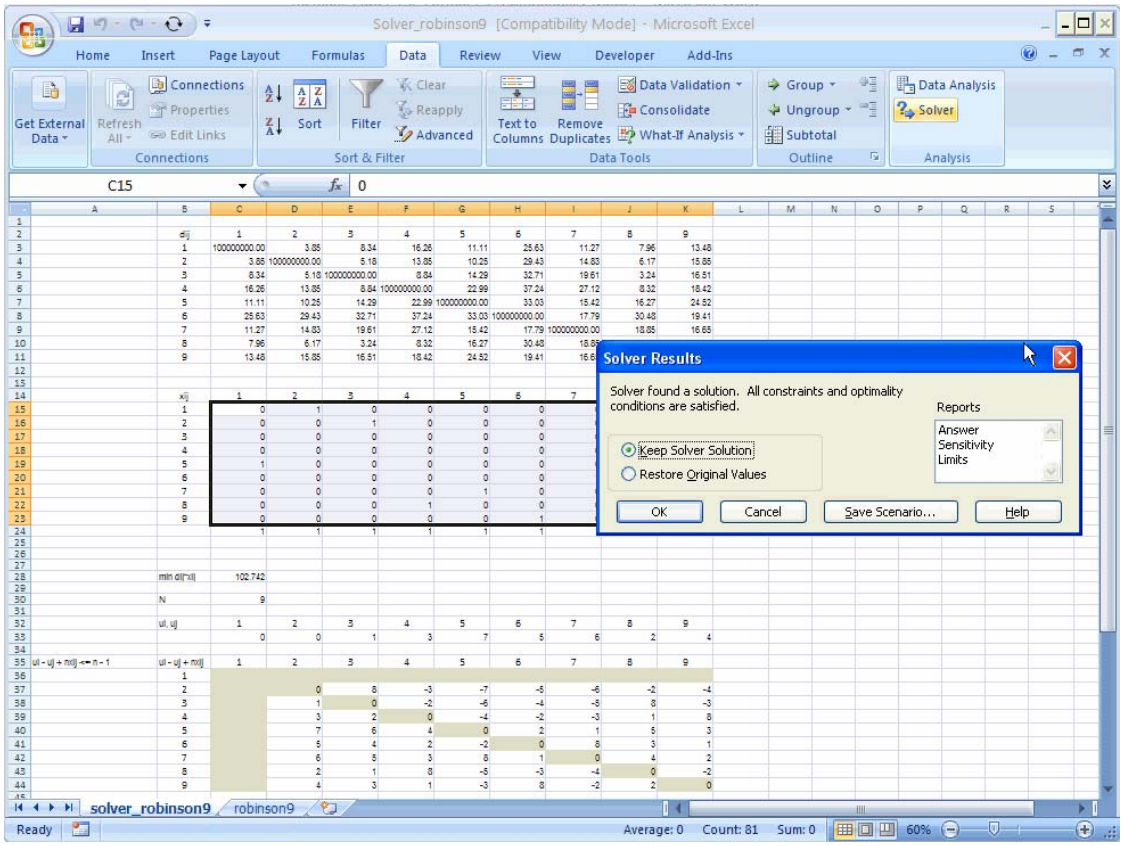


**Figure 4.2 Solver Results box showing an optimal solution found in Solver**

LINGO has a limit of 30 integer variables and 50 integer variables in demo version and student version, respectively. With the TSP formulation provided from LINGO "Samples" folder, we can solve the TSP with only 5 locations in demo version and up to 7 locations for Student version. Figure 4.3 shows the message box with the optimal solution when LINGO finishes solving a TSP. Figure 4.4 shows the error message when a number of integer variables exceed the capacity.

In practical application, LINGO and Excel Solver is capable for a small problem, saying, up to 6 to 7 locations and may not justify for even a little bigger problems. The application of the construction heuristics can be used instead with an acceptable error.

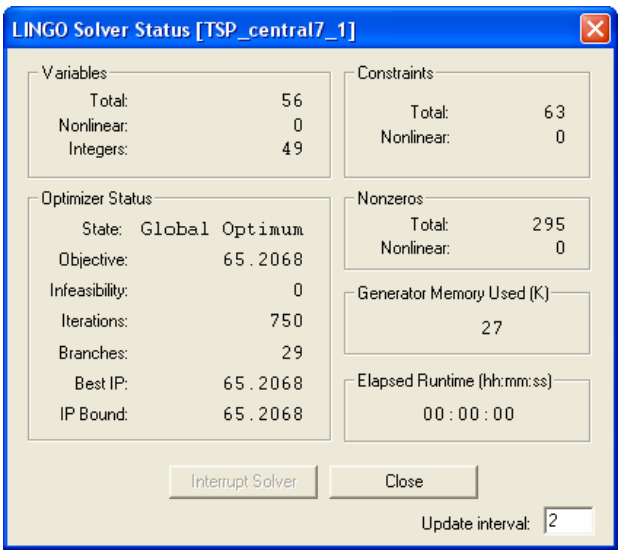


**Figure 4.3 LINGO Solver Status box showing the optimal value**

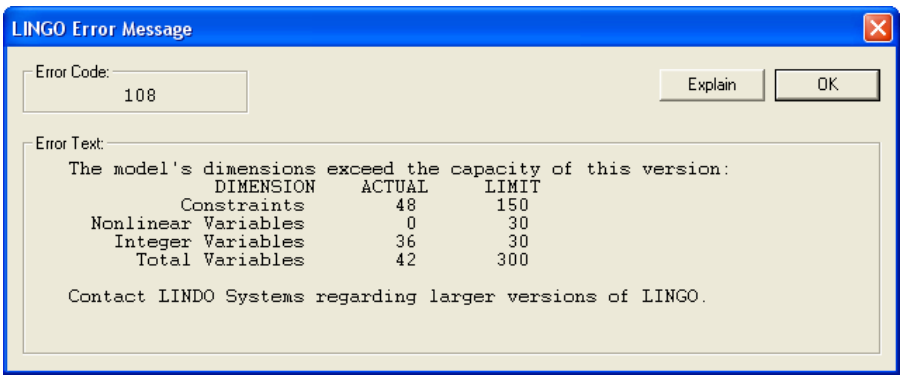


**Figure 4.4 LINGO Error Message box showing LINGO12 Demo capacity**

# CHAPTER V
# CONCLUSION AND RECOMMENDATION

## 5.1 Conclusion

The Traveling Salesman Problem (TSP) is basically the problem of finding the shortest route of a tour by visiting all the locations once and returning back to the starting location. A problem of this kind can be found in real applications ranging from a single salesman or a tourist to a large manufacturing company. However, this study focuses more on the application for general individuals or small companies who deal with small-sized problem. Thus, this study conducted the empirical test to review whether the simple basic heuristics are sufficiently justified for real applications of saying up to 50 locations.

In this study, three basic construction heuristics including Nearest Neighbor (NN), Nearest Insertion (NI), and Farthest Insertion (FI) are implemented and compared. The Farthest Insertion heuristics are actually implemented with three variations, namely Farthest Insertion 1 (FI1), Farthest Insertion 2 (FI2), and Farthest Insertion 3 (FI3). Thus, five heuristic algorithms are compared in both solution quality and computational time. The comparison is conducted with two set of data. The first data set is a collection of the 23 TSP instances from TSPLIB ranging from 17-city problem to 2392-point drilling problem with known optimal solution. The other data set is a collection of latitude-longitude coordinates of real locations which are three department stores and two hypermarkets. This data set comprises of Central, Robinson, The Mall Group department stores, as well as Carrefour and Tesco Lotus hypermarkets. For the second data set, the optimal solutions of the problems with up to 13 locations are acquired by Excel Solver or LINGO software. The optimal solutions of the problems with more than 13 locations are not acquired, so the solutions obtained from the heuristics are compared among one another.

From the test, Farthest Insertion gives the better solution quality, i.e. nearer to the optimal solution, than the Nearest Neighbor and the Nearest Insertion

heuristics for both data sets. Specifically, FI1 gives the best solution on average. With the test of 23 TSP instances from TSPLIB, FI1 gives the solution deviated from the optimal solution less than 10%. For the data set of up to 13 real locations, if applying both Nearest Insertions and Farthest Insertions, the best solution obtained is deviated from the optimal solution less than 4%, with even equal to the optimal solution most of the time. So, it is with strong confidence that the solution is good enough with small deviation from the optimal solution. By applying the basic TSP integer programming problem formulation in the Standard Excel Solver, it is considered not efficient for more than 10 locations and not even feasible for more than 13 locations since the number of variables exceeds the standard capacity limit. LINGO Student software is also applied with the second data set. With the same problem as the standard Excel Solver, there is a capacity limit for the number of integer variables. Hence, both insertion heuristics can be considered a good alternative to implement for real application with small-sized problems. The constructed tool can be sufficient for general individual users or small enterprise entrepreneurs that optimization software packages are not worth to acquire.

For real application, the distance between any two points may not be readily available, and the distance estimation may be helpful. If the latitude-longitude coordinates is known instead, then the Great Circle formula can be applied. This case, the distance estimation could be less than the true road distance by about 40% for the small area and about 30% for the large area.

The methodology described in this study from acquiring the distance from Google Maps to applying the nearest neighbor and insertion heuristics by MATLAB and Visual Studio 2008 to solve the problems could be used by any individuals or entrepreneurs such as direct-sales salespersons, traveling agency, laundry services,. This study also describes how to apply Google Maps to locate their new spots and use the Great Circle formula to estimate the distance.

## 5.2 Recommendation

This study aims to review the performance of the simple construction heuristics whether it can be justified for solving small-sized problems. Other heuristics can be implemented as an alternative application. Also, other metaheuristics can be used such as Genetics algorithm, but the parameters should be fine-tuned for the effective results.

In addition, the tool is constructed by the preference of the user and/or developer and the constructed tool in this study is mainly used to demonstrate the method with comparative purpose. In real practice, other tools or applications can be developed. Alternatively, other optimization software packages can be further reviewed whether its cost including acquired cost or purchasing cost, license and maintenance fee can be more worthwhile in practical use.

The real locations of the Central Group, Robinson, The Mall Group, Carrefour, and Tesco Lotus in Thailand are used in this study since these locations can be searched more easily in Google Maps. Other locations will be applied in real application which may have different characteristics. Even though the results of applying the heuristics are not expected to be far more different than those in this study, the details could be somewhat different. For example, to find the shortest tour of the cities across countries which may have far more distance on average than this study, it could be more accurate with the Great Circle formula distance estimation but further work shall be done to induce the conclusion.

# REFERENCES

Boyaci, A. (2007). *Traveling Salesman Problem Heuristics*. Retrieved October 9, 2010, from http://arman.boyaci.ca/files/tsp/homework-517-01.pdf.

Brest, J. and Zerovnik, J. (2005). *A Heuristic for the Asymmetric Traveling Salesman Problem*. Paper presented at the 6th Metaheuristics International Conference (MIC). Vienna, Austria.

Chungcharoen P. (2001). Application program for selection of traveling route using branch and bound technique. Bangkok: Mahidol University.

Cook, W. et. al. (2010). The Traveling Salesman Problem. Retrieved August 20, 2010, from http://www.tsp.gatech.edu/

Croes G. A. (1958). *A method for solving traveling-salesman problems*. Operations Research, 6(6):791 – 812.

Hahsler M. and Hornik K. (2010). *Introduction to TSP – Infrastructure for the Traveling Salesperson Problem*. Retrieved August 20, 2010, from http://cran.r-project.org/web/packages/TSP/vignettes/TSP.pdf

Hexa Software Development Center. (2003). *Geographical Distance Calculations*. Retrieved August 20, 2010, from http://www.zipcodeworld.com/docs/distance.pdf

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

Mutah University. (n.d.) *NP Complete Theory*. Retrieved October 31, 2010, from http://www.mutah.edu.jo/userhomepages/CS252/npcomplete.html

Nilsson C. *Heuristics for the Traveling Salesman Problem*. Retrieved August 20, 2010, from http://www.ida.liu.se/~TDDB19/reports_2003/htsp.pdf

Ozdemirel, N. E., (n.d.). *Week Two: Basic Searh and Construction Heuristics* [Powerpoint Slides]. Retrieved August 20, 2010, from http://www.ie.metu.edu.tr/~ie505/LectureNotes/Week%202%20(basic%20search%20&%20construction%20heuristics).pdf

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization: Algorithms and Complexity*. New York: Dover Publications, Inc.

Rosenkrantz, D. J., Stearns R. E., and Lewis P. M. (1977). *An analysis of several heuristics for the traveling salesman problem*. SIAM Journal on Computing, 6(3): 563 – 581.

Taha, H. A. (2003). *Operations Research: An Introduction* (7th ed.). New Jersey: Prentice Hall.

Tanthawichian, P. (2010). *Thailand Internet 2009 Snapshot by Truehits.net Statistics*. Retrieved August 20, 2010, from http://www.slideshare.net/pawoot/thailand-internet-truehits-awards2009

Universiteit Utrecht. (n.d.). *The Traveling Salesperson Problem Algorithms and Networks* [Powerpoint Slides]. Retrieved August 20, 2010, from www.cs.uu.nl/docs/vakken/an/an-tsp.ppt

Wikipedia. (n.d.). Travelling Salesman Problem. Retrieved October 31, 2010, from http://en.wikipedia.org/wiki/Travelling_salesman_problem

Winston, W. L. (2004). *Operations Research Applications and Algorithms* (4th ed.). Thompson Learning.

# APPENDIX

**Table A.1 Latitude-longitude coordinates of Robinson department stores**

| No. | Branch | Latitude | Longitude |
|---|---|---|---|
| 1 | Robinson Ratchada | 13.770589 | 100.572152 |
| 2 | Robinson Sukhumvit | 13.738213 | 100.559441 |
| 3 | Robinson Bangrak | 13.719606 | 100.515474 |
| 4 | Robinson Bangkae | 13.712803 | 100.433898 |
| 5 | Robinson Srinakarin | 13.701436 | 100.646423 |
| 6 | Robinson Rangsit | 13.997037 | 100.616398 |
| 7 | Robinson Ramindra | 13.840080 | 100.648155 |
| 8 | Robinson Lad Ya | 13.746055 | 100.502930 |
| 9 | Robinson Ratanatibet | 13.866940 | 100.496417 |
| 10 | Robinson Udornthani | 17.405387 | 102.799617 |
| 11 | Robinson Phuket | 7.877836 | 98.394451 |
| 12 | Robinson Nakhon Si Thammarat | 8.437681 | 99.969559 |
| 13 | Robinson Hadyai | 7.003118 | 100.469228 |
| 14 | Robinson Chiang Mai | 18.769391 | 98.975516 |
| 15 | Robinson Ubonratchathani | 15.240718 | 104.852547 |
| 16 | Robinson Sriracha | 13.168294 | 100.930863 |
| 17 | Robinson Ratchaburi | 13.534961 | 99.810584 |
| 18 | Robinson Chanthaburi | 12.599195 | 102.115600 |
| 19 | Robinson Ayutthaya | 14.335942 | 100.610889 |
| 20 | Robinson Jungceylon | 7.889624 | 98.300136 |
| 21 | Robinson Chonburi | 13.326761 | 101.003146 |
| 22 | Robinson Khonkaen | 16.432592 | 102.824937 |

**Table A.2 Latitude-longitude coordinates of The Mall Group department stores**

| No. | Branch | Latitude | Longitude |
|---|---|---|---|
| 1 | The Mall Ramkhamhaeng | 13.759561 | 100.610046 |
| 2 | The Mall Tha Pra | 13.715163 | 100.478897 |
| 3 | The Mall Ngamwongwarn | 13.859997 | 100.545802 |
| 4 | The Mall Bang Kae | 13.751391 | 100.407486 |
| 5 | The Mall Bangkapi | 13.778234 | 100.642147 |
| 6 | The Emporium | 13.736133 | 100.568590 |
| 7 | Siam Paragon | 13.745852 | 100.534097 |
| 8 | The Mall Nakorn Ratchasima | 14.980404 | 102.076147 |

**Table A.3 Latitude-longitude coordinates of Carrefour hypermarkets**

| No. | Branch | Latitude | Longitude |
|---|---|---|---|
| 1 | Carrefour Klong 3 | 14.033546 | 100.664297 |
| 2 | Carrefour Chaengwattana | 13.788738 | 100.504303 |
| 3 | Carrefour Bang Kae | 13.716263 | 100.447454 |
| 4 | Carrefour Bang Bon | 13.678993 | 100.436325 |
| 5 | Carrefour Bangpakok | 13.683377 | 100.493011 |
| 6 | Carrefour Bangpo | 13.806461 | 100.528851 |
| 7 | Carrefour Bang Yai | 13.698028 | 100.518723 |
| 8 | Carrefour Prachauthit | 13.611990 | 100.510837 |
| 9 | Carrefour Petchkasem | 13.708013 | 100.367215 |
| 10 | Carrefour Rama 2 | 13.649847 | 100.420548 |
| 11 | Carrefour Rama 4 | 13.719098 | 100.569011 |
| 12 | Carrefour Rangsit | 13.959601 | 100.619557 |
| 13 | Carrefour Ratchadapisek | 13.794349 | 100.574209 |
| 14 | Carrefour Rattanathibet | 13.861062 | 100.503294 |
| 15 | Carrefour Ram Intra | 13.862398 | 100.618769 |
| 16 | Carrefour Romklao | 13.792296 | 100.735826 |
| 17 | Carrefour Ladprao | 13.809972 | 100.568936 |
| 18 | Carrefour Lumlukka Klong 4 | 13.932890 | 100.682471 |
| 19 | Carrefour Srinakarintr | 13.648836 | 100.641275 |
| 20 | Carrefour Suanluang | 13.686350 | 100.668151 |
| 21 | Carrefour Sukhapiban 1 | 13.818617 | 100.653477 |
| 22 | Carrefour Sukhapiban 3 | 13.717029 | 100.572560 |
| 23 | Carrefour Suwintawong | 13.817304 | 100.719877 |
| 24 | Carrefour Sam Rong | 13.649120 | 100.641063 |
| 25 | Carrefour Nongchok | 13.855536 | 100.854814 |
| 26 | Carrefour Issaraparb | 13.732806 | 100.492705 |
| 27 | Carrefour Onnuch | 13.709889 | 100.601206 |

**Table A.4 Latitude-longitude coordinates of Tesco Lotus hypermarkets**

| No. | Branch | Latitude | Longitude |
|---|---|---|---|
| 1 | Tesco Lotus Bangkapi | 13.769455 | 100.643831 |
| 2 | Tesco Lotus Bangkae | 13.713066 | 100.418923 |
| 3 | Tesco Lotus Bangna-Trad | 13.653394 | 100.679277 |
| 4 | Tesco Lotus Bangpakok | 13.678198 | 100.501202 |
| 5 | Tesco Lotus Bangplee | 13.653394 | 100.679084 |
| 6 | Tesco Lotus Bangpoo | 13.654320 | 100.599548 |
| 7 | Tesco Lotus Bangyai | 13.824852 | 100.412258 |
| 8 | Tesco Lotus Chaengwattana | 13.895817 | 100.556378 |
| 9 | Tesco Lotus Charansanitwong | 13.788498 | 100.501666 |
| 10 | Tesco Lotus Fortune Town | 13.758310 | 100.565221 |
| 11 | Tesco Lotus Laksi | 13.879663 | 100.600184 |

| | | | |
|---|---|---|---|
| 12 | Tesco Lotus Minburi | 13.816939 | 100.728345 |
| 13 | Tesco Lotus Nakornprathom | 13.809297 | 100.052737 |
| 14 | Tesco Lotus Pathumtani | 14.020178 | 100.526549 |
| 15 | Tesco Lotus Pinklao | 13.776885 | 100.477897 |
| 16 | Tesco Lotus Prachachuen | 13.805979 | 100.535030 |
| 17 | Tesco Lotus Ramindra | 13.802077 | 100.613839 |
| 18 | Tesco Lotus Rangsit | 13.992933 | 100.614402 |
| 19 | Tesco Lotus Rangsit - Nakornnayok | 13.999242 | 100.680468 |
| 20 | Tesco Lotus Rangsit Klong 7 | 14.036158 | 100.732814 |
| 21 | Tesco Lotus Rattanatibet | 13.862234 | 100.518752 |
| 22 | Tesco Lotus Salaya | 13.776885 | 100.477897 |
| 23 | Tesco Lotus Seacon Square | 13.692672 | 100.647710 |
| 24 | Tesco Lotus Srinakarin | 13.620911 | 100.620349 |
| 25 | Tesco Lotus Sukhapiban 1 | 13.825680 | 100.659847 |
| 26 | Tesco Lotus Sukhumvit 50 | 13.705585 | 100.600323 |
| 27 | Tesco Lotus Ladprao on Paholyothin Rd. | 13.831621 | 100.570191 |
| 28 | Tesco Lotus Lumluka Klong 2 | 13.969360 | 100.643697 |
| 29 | Tesco Lotus Wanghin | 13.824102 | 100.590236 |
| 30 | Tesco Lotus Rama 1 | 13.748741 | 100.524444 |
| 31 | Tesco Lotus Rama 2 | 13.647947 | 100.419288 |
| 32 | Tesco Lotus Rama 3 | 13.696852 | 100.541615 |
| 33 | Tesco Lotus Rama 4 | 13.718418 | 100.568488 |
| 34 | Tesco Lotus Krabi | 8.107452 | 98.940897 |
| 35 | Tesco Lotus Kanjanaburi | 14.005290 | 99.545767 |
| 36 | Tesco Lotus Kon Kaen | 16.400768 | 102.815147 |
| 37 | Tesco Lotus Chanthaburi | 12.616720 | 102.101534 |
| 38 | Tesco Lotus Chon Buri | 13.340463 | 100.973116 |
| 39 | Tesco Lotus Chaiyaphum | 15.812090 | 102.024692 |
| 40 | Tesco Lotus City Park - Bangplee | 13.653658 | 100.678893 |
| 41 | Tesco Lotus Trang | 7.542641 | 99.618852 |
| 42 | Tesco Lotus Nakornpathom | 13.809264 | 100.052855 |
| 43 | Tesco Lotus Nakhon Si Thammarat | 8.398239 | 99.976525 |
| 44 | Tesco Lotus Navanakorn | 14.122616 | 100.617267 |
| 45 | Tesco Lotus Baan Pong | 13.793841 | 99.919890 |
| 46 | Tesco Lotus Pathumthani | 14.020188 | 100.526576 |
| 47 | Tesco Lotus Pakchong | 14.521107 | 100.922779 |
| 48 | Tesco Lotus Fang | 19.910849 | 99.204381 |
| 49 | Tesco Lotus Pattaya (North) | 12.951066 | 100.893086 |
| 50 | Tesco Lotus Pattaya (South) | 12.906261 | 100.896305 |
| 51 | Tesco Lotus Phitsanulok | 16.817875 | 100.300941 |
| 52 | Tesco Lotus Phuket | 7.904850 | 98.368988 |
| 53 | Tesco Lotus Mahachai | 13.577530 | 100.271621 |
| 54 | Tesco Lotus Rayong | 12.684484 | 101.269830 |
| 55 | Tesco Lotus Ratchaburi | 13.537167 | 99.819447 |
| 56 | Tesco Lotus Lopburi | 14.787787 | 100.677523 |
| 57 | Tesco Lotus Lamai | 9.473895 | 100.042625 |
| 58 | Tesco Lotus Samui | 9.534582 | 100.041740 |
| 59 | Tesco Lotus Saraburi | 14.521107 | 100.922285 |
| 60 | Tesco Lotus Sampran | 13.743885 | 100.216033 |
| 61 | Tesco Lotus Suphan Buri | 14.469812 | 100.131009 |

| | | | |
|---|---|---|---|
| 62 | Tesco Lotus Nongkai | 17.872728 | 102.741544 |
| 63 | Tesco Lotus Hua Hin | 12.574691 | 99.955424 |
| 64 | Tesco Lotus Hat Yai | 7.007824 | 100.491273 |
| 65 | Tesco Lotus Ayutthaya | 14.337043 | 100.610977 |
| 66 | Tesco Lotus Udon Thani | 17.431499 | 102.787389 |
| 67 | Tesco Lotus Ubon Ratchathani | 15.252615 | 104.849291 |
| 68 | Tesco Lotus Chiang Mai Kad Kamtiang | 18.810821 | 98.997392 |
| 69 | Tesco Lotus Chiang Mai Hangdong | 18.759532 | 98.972241 |
| 70 | Tesco Lotus Korat | 14.978894 | 102.071043 |

# BIOGRAPHY

| | |
|---|---|
| **NAME** | Vilasinee Leowarin |
| **DATE OF BIRTH** | 6 September 1978 |
| **PLACE OF BIRTH** | Bangkok, Thailand |
| **INSTITUTIONS ATTENDED** | Chulalongkorn University, 1996-1999<br>Bachelor of Engineering (Industrial Engineering)<br>Georgia Institute of Technology, 2001-2003<br>Master of Science (Industrial and System Engineering)<br>Mahidol University, 2009-2010<br>Master of Business Administration (Business Modeling and Analysis) |
| **HOME ADDRESS** | 338/13 Soi Lad Prao 80, Lad Prao Road,<br>Wangthonglang, Bangkok 10310<br>Tel. 02-539-8042<br>E-mail : winnyleowarin@yahoo.com |
| **EMPLOYMENT ADDRESS** | 1 Q-House Lumphini<br>Sathorn Road, Bangkok 10120 |