Old Dominion University

ODU Digital Commons

Engineering Management & Systems Engineering Theses & Dissertations

Engineering Management & Systems Engineering

Spring 2011

# A Hybrid Lehmer Code Genetic Algorithm and Its Application on Traveling Salesman Problems

Jun Zhang
*Old Dominion University*

Follow this and additional works at: https://digitalcommons.odu.edu/emse_etds

Part of the Operational Research Commons, and the Systems Engineering Commons

Recommended Citation
Zhang, Jun. "A Hybrid Lehmer Code Genetic Algorithm and Its Application on Traveling Salesman Problems" (2011). Doctor of Philosophy (PhD), Dissertation, Engineering Management & Systems Engineering, Old Dominion University, DOI: 10.25777/hw0b-3z55
https://digitalcommons.odu.edu/emse_etds/140

This Dissertation is brought to you for free and open access by the Engineering Management & Systems Engineering at ODU Digital Commons. It has been accepted for inclusion in Engineering Management & Systems Engineering Theses & Dissertations by an authorized administrator of ODU Digital Commons. For more information, please contact digitalcommons@odu.edu.

# A HYBRID LEHMER CODE GENETIC ALGORITHM AND ITS APPLICATION ON TRAVELING SALESMAN PROBLEMS

by

Jun Zhang
M.E. July 2007, Nanchang University, China

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

ENGINEERING MANAGEMENT

OLD DOMINION UNIVERSITY
May 2011

Approved by:

Shannon Bowling (Director)

Resit Unal (Member)

Ariel Pinto (Member)

Leonardo Bedoya-Valencia
(Member)

## ABSTRACT

A HYBRID LEHMER CODE GENETIC ALGORITHM AND ITS APLICATION ON TRAVELING SALESMAN PROBLEMS

Jun Zhang
Old Dominion University, 2011
Director: Dr. Shannon R. Bowling

Traveling Salesman Problems (TSP) is a widely studied combinatorial optimization problem. The goal of the TSP is to find a tour which begins in a specific city, visits each of the remaining cities once and returns to the initial cities such that the objective functions are optimized, typically involving minimizing functions like total distance traveled, total time used or total cost.

Genetic algorithms were first proposed by John Holland (1975). It uses an iterative procedure to find the optimal solutions to optimization problems.

This research proposed a hybrid Lehmer code Genetic Algorithm. To compensate for the weaknesses of traditional genetic algorithms in exploitation while not hampering its ability in exploration, this new genetic algorithm will combine genetic algorithm with 2-opt and non-sequential 3-opt heuristics. By using Lehmer code representation, the solutions created by crossover parent solutions are always feasible.

The new algorithm was used to solve single objective and multi-objectives Traveling Salesman Problems. A non Pareto-based technique will be used to solve multi-objective TSPs. Specifically we will use the Target Vector Approach. In this research, we used the weighted Tchebycheff function with the ideal points as the reference points as the objective function to evaluate solutions, while the local search heuristics, the 2-opt and non-sequential 3-opt heuristics, were guided by a weighted sum function.

This dissertation is dedicated to my parents, my wife and my kids: Leo and Aaron.

## ACKNOWLEDGMENTS

I would like to sincerely thank my advisor, Dr. Shannon Bowling, for your guidance and support in the three and half years of my PhD studies. Our numerous academic meetings and discussions helped me develop the ideas in this dissertation. Without your help and encouragement the completion of my dissertation would not have been possible. In addition to being my mentor, you are also a good friend. Your friendship has made my PhD experience at Old Dominion University a very pleasant journey. I enjoyed those warm moments when we exchanged our views on life and found that we shared a lot in our values. You will definitely continue to be my mentor and good friend for the rest of my life.

I would also like to thank Dr. Ariel Pinto for your assistance in my PhD research and for the trouble you took in writing those reference letters for me. The time and effort you spent reviewing my initial proposal, your constructive comments on it and all the other help you gave me are highly appreciated.

I am very grateful to Dr. Resit Unal, too, for agreeing to be one of my committee members, after Dr. Kady, who was on my dissertation committee, left. As one of your students, I learned a lot from your courses, “Robust Engineering Design” and “Reliability and Maintainability”, and I really appreciate the time and effort you put into reviewing my dissertation.

I would also like to thank Dr. Leonardo Bedoya-Valencia for your comments on my initial proposal and for the papers you sent me. These papers were really helpful for my research. And I appreciate the time and effort you spent reviewing my proposal and dissertation.

I am grateful to my friends, Mahmoud T. Khasawneh, Elkin Rodriguez and Nevan Shearer, for your friendship and support. I really enjoyed the days we spent on course study, projects and attending conferences. I value our friendship and hope it will be a lifetime one.

Finally, my thanks go to my wife, Hong, and my two children, Leo and Aaron. Hong, you are a great wife. You devoted all of your time to our family and supported me unconditionally during my PhD studies. Leo and Aaron, you are the best kids I could have. You have brought so much love and fun to my life. You made my PhD experience so much more enjoyable. Hong, Leo and Aaron, I love you with all my heart.

# TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

# CHAPTER 1

# INTRODUCTION

Morse and Kimball (1951) defined Operations Research (OR) as "a scientific method of providing executive departments with a quantitative basis for decisions regarding the operations under their control." OR uses methods to get optimal or near optimal solutions to complex problems. These methods include mathematical modeling, statistics and algorithms, etc. Hillier and Lieberman (2005) summarized the usual phases of an operation research study as the following:

1. Define the problem and collect the data.
2. Formulate mathematic model to represent the problem.
3. Develop a procedure to find the solutions to the problem from the model.
4. Test and refine the model.
5. Prepare for the application of the model.
6. Implement the application.

## 1.1 Background

Searching for optimal or near optimal solutions to optimization problems is the subject matter of much research in the field of operations research. There are several reasons that make it difficult to solve real-world optimization problems; these include:

- The search space is too large to perform an exhaustive search.
- The problems are so complicated that we need to use simplified models. But the results from simplified models are essentially useless.
- The evaluation functions are noisy or time dependent. This requires finding not just a single solution but a series of solutions.

- The possible solutions are so heavily constrained that it is difficult to construct even one feasible answer, let alone an optimal solution.

When attacking hard complex optimization problems, especially combinatorial optimization problems, classical optimization methods may fail to be effective and efficient. From the 1970s a number of metaheuristics have been proposed for solving these kinds of problems, among which are genetic algorithm (Holland, 1975), simulated annealing (Kirkpatrick, Gelatt & Vecchi, 1983), tabu search (Glover, 1989), ant colony algorithms (Colorni, Dorigo & Maniezzo, 1991) and particle swan optimization (Kennedy & Eberhart, 1995).

Combinatorial optimization is defined as the mathematical study of finding an optimal arrangement, grouping, ordering, or selection of discrete objects usually finite in numbers (Osman and Laporte, 1996). For many combinatorial optimization problems it is computationally impossible to find the optimal solution. Usually what can practically be produced are near-optimal solutions.

The term metaheuristic was first introduced by Glover (1986). It derives from the composition of two Greek words: heuristic, which means "to find" and the suffix Meta, which means "beyond, in an upper level." A metaheuristic is a heuristic method to solve optimization problems. Much of the development of metaheuristics comes from observing nature and implementing simple rules to solve complex problems. It can be defined as a high-level framework or method which is specialized to solve optimization problems, or a high-level strategy that helps other optimization methods in the process of searching for feasible solutions. Osman and Laporte (1996) defined a metaheuristic as " an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, learning strategies are used to structure information in order to find efficiently near-optimal solutions."

Exploration and exploitation of the solution spaces are the two competing goals which direct the design of a metaheuristic. The success of a metaheuristic depends on the good balance between exploration and exploitation. Exploration is needed to make sure that the solution space is searched enough to make a reliable estimate to the global optimal. Exploitation is also important because the improvement of the current solution often produces a better solution (Talbi, 2002). Generally, a metaheuristic is applied to complex problems which no specific general algorithms or methods can solve. The most common problems that metaheuristics solve are combinational optimization problems.

Although in theory metaheuristics can solve any optimization problem, especially complex problems which no general algorithm can solve, according to the "no free lunch" theorems (Wolpert & Macready, 1997), there is no optimization method that is perfect enough to solve all optimization problems efficiently. The performance of any algorithm over one class of problems is offset by its performance over another class. This is one of the reasons that a number of metaheuristics which focus on solving certain kinds of optimization problems are proposed, such as genetic algorithm (Holland, 1975), simulated annealing (Kirkpatrick, Gelatt & Vecchi, 1983), tabu search (Glover, 1989), ant colony (Colorni, Dorigo & Maniezzo, 1991), particle swarm optimization (Kennedy and Eberhart, 1995), etc., and variants for metaheuristics which introduce some changes to the original algorithm to improve its performance were also proposed (Grefenstette, 1986; see also Chiang & Russell, 1997; Li & Li, 2008; Wang & Wang, 2008).

Over the years, research on hybrid metaheuristics has risen considerably in combinatorial optimization. This research attempts to combine the best features from different metaheuristics to develop more powerful hybrid implementations than the original metaheuristics. One of the most common formats of the hybrid metaheuristics is: the population-based metaheuristics, like genetic algorithm, ant colony, particle swarm etc., which are powerful in exploring the solution space, were combined with local search metaheuristics, like hill climbing, simulated annealing, tabu search etc., which are more powerful in terms of exploitation to develop more powerful hybrid metaheuristics (Suh & Van Gucht, 1987; see also Fleurent & Ferland, 1994; Kim, Hayashi & Nara, 1995; Chen & Flann, 1994; Chen et al., 1995).

Talbi (2002) categorized all hybrid metaheuristics into four hierarchy categories: Low–level relay hybrid, low-level teamwork hybrid, high-level relay hybrid and high-level teamwork hybrid. He also categorized all hybrid metaheuristics using flat classification: homogeneous versus heterogeneous, global versus partial, specialist versus general. Figure 1.1 shows the hierarchical and flat classification of hybrid metaheuristics (Talbi, 2002).

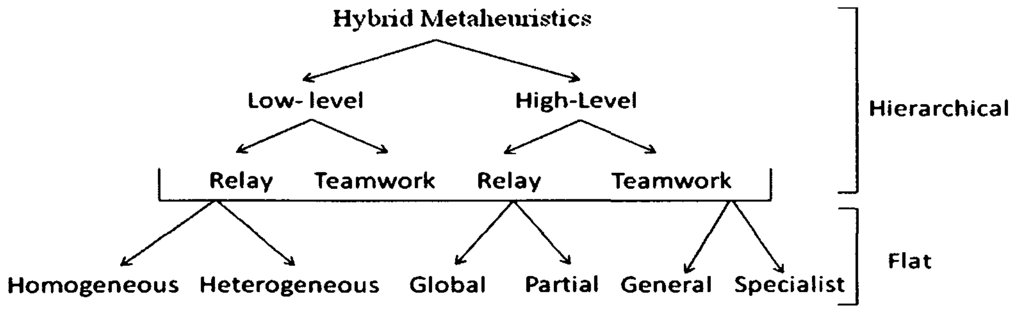


Figure 1. Classification of Hybrid Metaheuristic (Talbi, 2002)

Traveling Salesman Problems (TSP) is a widely studied combinatorial optimization problem. The goal of the Traveling Salesman Problems is to find a tour which begins in a specific city, visits each of the remaining cities once and returns to the initial cities such that the objective functions are optimized, typically involving minimizing functions like total distance traveled, total time used or total cost. The simplest TSP involves finding a shortest path that visits n cities and returns to the initial point and the distances between cities are symmetric while in multi-objective TSP, the goal is to simultaneously optimize distances, cost, times or other objectives.

TSP is NP-hard even with only a single objective. For multi-objective TSP, it has the difficulty of the TSP itself and the difficulty of multiple objectives (Ehrgott, 2000). Therefore, heuristics which provide sub-optimal solutions are widely used to tackle multi-objective TSP, such as tabu search (Hansen, 2000), genetic algorithm (Jaszkiewicz, 2002), particle swarm (Shi, 2007), evolutionary algorithm (Jozefowiez, 2008), etc.

As a population based algorithm, GA deals with multiple solutions in one single simulation run. Because of this, GA can maintain a diverse set of solutions. This makes GA promising in dealing with multi-objective TSP. Many variant GAs were proposed to attack TSP and multi-objective TSP (Fonseca & Fleming, 1993; see also Chatterjec, Carrera & Lynch, 1996; Merz & Freisleben, 2002; Deb et al., 2002; Jaszkiewicz, 2002; Samanlioglu, Ferrell & Kurz, 2008). The performance of GAs on TSP depends very much on the encoding methods and the genetic operators, crossover and mutation. To overcome the problem of infeasible solutions coming from the crossover that uses the natural representation, repairing methods were proposed, such as partially mapped crossover (Goldberg & Lingle, 1985; Oliver, Smith & Holland, 1987), edge recombination crossover (Whitley et al., 1989), etc. Another way to solve the creating infeasible solutions problem is to change the encoding method. For example, random keys (Bean, 1994; Samanlioglu, Ferrell & Kurz, 2008), and Lehmer code (Mantaci & Rakotondrajao, 2001; Martin, 1990) were used to code the solutions for TSP.

2-opt heuristic was first introduced by Croes (1958). It involves in breaking the route by deleting two edges and reconnecting the broken paths in the other possible way. If the 2-opt heuristic results in an improved tour, this change will be kept. Otherwise, the tour would not change. Figure 2 shows how 2-opt heuristic works. Please note that this picture is a schematic. If the distances were as shown in the figure, the tour will not change because the 2-opt will not result in a shorter tour. The Genetic Algorithms incorporated with 2-opt heuristic has been used to optimize TSP and Multi-objective TSP (Merz & Freisleben, 2002; see also Deb et al., 2002; Jaszkiewicz, 2002; Nilsson, 2003; Ghoseiri & Sarhadi, 2007; Samanlioglu, Ferrell & Kurz, 2008).

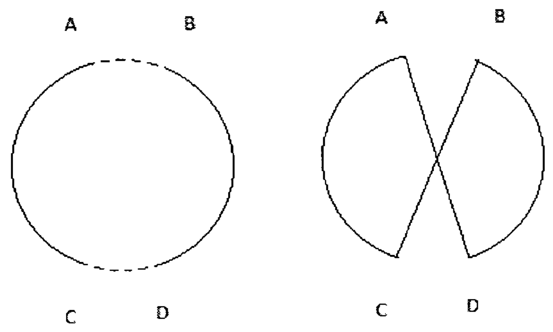


Figure 2. 2-opt Move (Original Tour Left, Resulting Tour Right)

3-opt heuristics involves in breaking the solution route by deleting three edges and reconnecting the broken paths in the other possible ways. Solutions that are 3-opt are also 2-opt.

## 1.2 Problem Statement

A meta-heuristics can be a high-level framework or method which is specialized to solve optimization problems, or a high-level strategy that helps other optimization methods in the process of searching for feasible solutions. But according to the "no free lunch" theorems (Wolpert & Macready, 1997), the performance of any optimization method over one class of problems is offset by its performance over another class. Hybrid meta-heuristics combine different meta-heuristics to produce more powerful implementations than the original ones.

Over the past decades, genetic algorithm attracted a lot of attention as a global optimization technique for complex optimization problems. Genetic algorithms were first proposed by John Holland (1975). They use an iterative procedure to find the optimal solutions to optimization problems and are categorized as a global search method. Genetic algorithms have been applied to solving optimization problems like numerical function optimization, scheduling, cognitive modeling, transportation problems, travel salesman problems, graph coloring problems, database query optimization, etc. (Bennett, Ferris & Ioannidis, 1991; see also De Jong, 1985; Goldberg, 1989; Vignaux & Michalewicz, 1991).

Genetic algorithm took clues from nature: genetic inheritance and fitness survival. As a population based optimization method, genetic algorithm is powerful in exploration but weak in exploitation. The hybrid genetic algorithm which combined genetic algorithm and local search meta-heuristics, such as hill climbing, simulated annealing, tabu search, etc. can be more efficient and effective than the original genetic algorithm and local search meta-heuristics. One of the weaknesses with genetic algorithm is that sometimes it converges towards local optimal.

In (Rugolph, 1994), the convergence properties of canonical genetic algorithm with mutation and crossover operators, proportional reproduction applied to static optimization problems were analyzed. It concluded that "a canonical genetic algorithm will never converge to the global optimal regardless of the initialization, operators and objective function. But the variants of canonical genetic algorithm that always maintain the best solution in the population are shown to converge to the global optimal due to the irreducibility of the underlying original non-convergent canonical genetic algorithm" (Rugolph, 1994).

This research proposes a Lehmer code Genetic Algorithm to solve single objective and multi-objective TSPs. To compensate for the weakness of traditional Genetic Algorithms in exploitation while not hampering its ability in exploration, local search heuristic—2-opt heuristic and non-sequential 3-opt heuristic was incorporated into the new Genetic Algorithms. Whenever a new solution was produced, no matter if it is produced by crossover or mutation, 2-opt heuristic and/or non-sequential 3-opt heuristic will be used to improve it until a local optimal solution is obtained. This Genetic Algorithm will benefit from using the local search heuristics, 2-opt and non-sequential 3-opt heuristics because they are powerful in exploitation. Therefore, these Genetic Algorithms will have both good exploration and exploitation ability. The algorithms will converge towards a solution quicker. Ideally, this solution will be a global optimal solution or near optimal solution.

Traditional Genetic Algorithms use direct representation of the solutions. That is, for the Traveling Salesman Problem, the solution was directly coded by the numerical representation of the cities. This makes it difficult for the traditional Genetic Algorithms to maintain feasibility from parents to offspring when solving many optimization problems, like multiple machine scheduling, traveling salesman problems, etc. Crossover is a genetic operator used to vary the programming of chromosomes from the two individuals of the fittest to form the next generation. Traditional Genetic Algorithm uses one point crossover (Holland, 1975). Suppose we have a traveling salesman problem that has 6 cities. A candidate for this problem is a permutation of these 6 cities. Two such permutations are 1→ 3 →2→4→5→6→1 and 6→ 4 →2→3→1→5→6. In traditional Genetic Algorithm, the genetic representation of these two sequences are the permutations $x = (1,3,2,4,5,6,1)$ and $x' = (6,4,2,3,1,5,6)$. A one-point crossover will divide each permutation at the crossover point and exchange certain segments of the two permutations. Suppose the crossover point is the fourth place of the permutation. From the Figure 1, we can see that the resulting sequences are 1→ 3 →2→4→1→5→6 and 6→ 4 →2→3→5→6→1. Both of them are infeasible. So the crossing over of two feasible solutions does not result in feasible solutions.

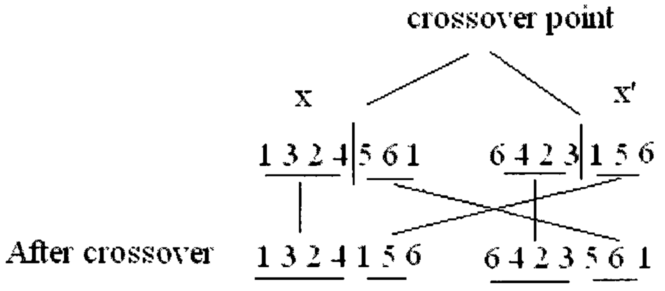


Figure 3. Traditional Single Point Crossover (Infeasible Solution)

To overcome the difficulty in maintaining feasibility from parent to offspring, Bean (1994) proposed the Random Keys Genetic Algorithm. Not like traditional Genetic Algorithm using direct chromosomal representation of the candidate solution, the Random Keys Genetic Algorithm uses chromosomal representation in a soft manner. It encodes the candidate solution with random numbers. The values of these random numbers range from 0 to 1. The random numbers are used as the sort keys to encode the candidate solutions. This encoding technique will eliminate the feasibility problem.

Although the random key genetic algorithm can solve the infeasible solutions problem, it has its own difficulties. First, when implemented on a large problem, the sorting process to determine the ranks is expensive. Moreover, the encoding process does not preserve the adjacency of cities in a given tour when crossover with random keys (Chatterjee, Carrera & Lynch, 1996).

This research will use Lehmer code to encode the permutation of a candidate solution. Lehmer code can code each permutation $\Pi_n$ of n numbers with a function $LC(\Pi\ ):\{1,...,n\}\rightarrow\{1,...,n-1\}$ to a special sequence of n-1 numbers (Kromer, Platos and Snasel, 2009). Lehmer code of a permutation can be expressed by using an inversion table. Consider a sequence of n numbers $x=(x_1x_2...x_n)$ . An inversion is a pair $(x_i,x_j)$ such that $i<j$ and $x_i>x_j$. For $i\in\{1,...,n\}$, let $d_i$ count the number of inversions with i as the smaller index. Then the sequence $(d_1d_2...d_n)$ is called inversion table of permutation x. $0\leq d_i\leq n-i$ for $i=1,...,n$.

The evolutionary multi-objective optimization methods can be classified into two types: the Pareto-based technique and non Pareto-based technique (Samanlioglu, Ferrell & Kurz, 2008). In the Pareto-based techniques, the selection is directed by the Pareto dominance and Pareto ranking. The multi-objective genetic algorithm proposed by Fonseca and Fleming (Fonseca & Fleming, 1993), the niched Pareto genetic algorithm (Horn, Nafpliotis & Goldberg, 1994), NSGA II (Deb et al., 2002), SPEA II (Zitzler, Laumanns & Thiele, 2001), etc., belong to this type of technique.

In the non Pareto-base techniques, the selection does not directly rely on the Pareto dominance and Pareto ranking, such as vector evaluated genetic algorithm (Schaffer, 1984), target vector approach (Coello, 2001), memetic random key genetic algorithm (Samanlioglu, Ferrell & Kurz, 2008), etc.

According to (Samanlioglu, Ferrell and Kurz, 2008), the main advantage of Pareto-based techniques is that these methods don't need to normalize objective functions, set reference points and specify weighting coefficients for each objective function according to its importance. But the Pareto-based techniques also have some disadvantages. First, the Pareto ranking does not work for hybrid meta-heuristics with local search because many local moves do not influence the rank of a solution. In some cases, change of a rank of a solution may need to change the objective function value a lot. But this may not be achieved by local move. And for solutions which have been already ranked 1, local improvement is not possible (Jaszkiewicz, 2002). Another problem with the Pareto-based techniques is with the comparability of the solutions. According to (Knowles and Corne, 2004), the Pareto-based techniques may be suited for problems with only two or three objective functions. When working on the multi-objective optimization problems with four or more objective functions, the Pareto-based technique may cause many problems because many solutions will be incomparable.

This research presents a hybrid Lehmer code genetic algorithm, which combines genetic algorithm with 2-opt heuristics and non sequential 3-opt heuristics. And this algorithm will be used to solve single objective Traveling Salesman Problems and multi-objective Traveling Salesman Problems that have up to four or more objective functions. And this research will use a non Pareto-based technique. Specifically we will use the

Target Vector Approach that was also used in (Coello, 2001), (Samanlioglu, Ferrell and Kurz, 2008), etc. In Target Vector Approach, the goal is to minimize the distance between the generated solution and the target vector. In this research, we use the weighted Tchebycheff function with the ideal points as the reference points.

The format of this dissertation is as follows; there will be nine chapters in the dissertation, including:

The first chapter is the introduction. In this chapter, background is given and the problem statement will be discussed.

The second chapter is a literature review. Literature on related topics will be summarized in this chapter.

The third chapter is methodology. This is the main part of the dissertation. This chapter will discuss how to develop and implement the hybrid Lehmer code genetic algorithm and how it is applied on a multi-objective Traveling Salesman Problem.

Chapter four will test the performance of the new algorithm proposed in chapter three. In this chapter, I will use the method proposed in chapter three to solve some bench mark TSPs and compare the performance of the new genetic algorithms with the performances of other meta-heuristics.

Chapter five will summarize the results for chapter four. We can get a good understanding about how the Lehmer code can be used in genetic algorithms to solve multi-objective TSPs. And we will summarize the advantages and disadvantages of the proposed genetic algorithms on solving multi-objective meta-heuristics.

Chapter six will discuss future work. This chapter will discuss the potentials of the other techniques that can be used to improve the performance of the hybrid Lehmer code genetic algorithms on solving multi-objective TSPs and the potential to use this new genetic algorithm to solve other combinatorial optimization problems.

## 1.3 Contributions

In this research, after investigating metaheuristics, especially Genetic Algorithms and Random Keys Genetic Algorithms and their applications on the Traveling Salesman Problem, a new Hybrid Lehmer Code Genetic Algorithm was proposed. This new Genetic Algorithm used Lehmer code to represent the potential solutions to solve the infeasible solutions problem associated with traditional Genetic Algorithms when solving discrete optimization problems. And 2-opt and non-sequential 3-opt heuristics were embedded into the new algorithm to conduct a local search. This provided an alternative way to use Genetic Algorithm to solve discrete optimization problems.

This new Genetic Algorithm was implemented by using Matlab, and experiments were conducted to test its performance on single objective and multi-objective TSPs. The results showed that the new Genetic Algorithm had some advantages on some bench mark single-objective TSPs over the newly proposed methods, more specifically Hansen and Samanlioglu's methods (Hansen, 2000; Samanlioglu, Ferrell & Kurz, 2008) even with small population size and fewer generations. And the results of this new Genetic Algorithm on multi-objective TSPs were comparable with those from the methods proposed in the new literatures.

Another advantage of the new Genetic Algorithm over traditional Genetic Algorithms and Random Keys Genetic Algorithms is easy implementation. No fixing procedure is required as in traditional Genetic Algorithms when solving discrete optimization problems.

# CHAPTER 2

# REVIEW OF LITERATURE

Traveling Salesman Problem (TSP) is a well studied combinatorial optimization problem. Many exact and approximate heuristics and meta-heuristics were proposed to solve this problem. It has also served as the bench mark problem to test the efficiency of the newly proposed heuristics and meta-heuristics.

## 2.1 Traveling Salesman Problem

The Traveling Salesman Problem is a well studied and important combinatorial optimization problem. The idea of the Traveling Salesman Problem is to find the shortest tour that visits each city exactly once and returns to the start city, given a list of cities. The origin of the Traveling Salesman Problem was not clear. But the Traveling Salesman Problem was first formulated as a mathematical problem in 1930 by Karl Menger.

Mathematically, the TSP can be defined as the following:

In the graph $\mathrm{G} = (\mathrm{V}, \mathrm{C})$, V is the set of nodes, or cities, C is the "cost matrix", where $c_{ij}$ represents the cost of going from city $i$ to $j$, $i, j \in V$. The goal is to find the permutation $(\mathrm{i}_1, i_2, \ldots, i_n)$ of the integers from 1 to n such that the total cost $c_{i_1 i_2} + c_{i_2 i_3} + \ldots + c_{i_n i_1}$ is minimized.

The traveling salesman problem is a NP-complete problem. There is no polynomial-time algorithm that is capable of solving it exactly (Karp, 1977). Homaifar (1992) states "one approach which would certainly find the optimal solution of any TSP is the application of exhaustive enumeration and evaluation." But in most situations, conducting exhaustive enumeration and evaluation will take a long time. And obviously we need to find an algorithm that gives us a solution in a short time. This means that we probably need to sacrifice optimality to get a good solution in a shorter time.

The algorithms for solving the traveling salesman problem can be classified into two classes: exact algorithm and approximate algorithms (Helsgaun, 2000). The most direct method would be the method that tries all permutations and finds the shortest path. The running time for this approach is within a polynomial factor of $O(n!)$. Various branch-and-bound algorithms belong to the exact algorithms. These algorithms are inefficient concerning the running time, especially when solving the traveling salesman problem with a large number of cities.

So many approximate heuristics were proposed to solve the travel salesman problem. Ghosh et al. (2007) introduced tolerance-based greedy algorithms to solve traveling salesman problem. The nearest neighbor algorithm is one of the first algorithms used to find a solution for the traveling salesman problem (Karp, 1977; Gutin, Yeo & Zverovich, 2002). Several genetic-based algorithms were proposed to solve the traveling salesman problem (Grefenstette, Gopal & Rosmaita, 1985; see also Jog, Suh & Gucht, 1989; Oliver, Smith & Holland, 1987; Seniw, 1991). Dorigo and Gambardella (1997) proposed an artificial ant colony algorithm that is capable of solving the traveling salesman problem. Shi, Zhou, Wang, Wang and Liang (2007) presented a discrete particle swarm optimization algorithm for traveling salesman problem. Ghoseiri and Sahadi (2008) present a memetic algorithm to solve the symmetric traveling salesman problem.

## 2.2 Multi-objective Traveling Salesman Problem

The multi-objective optimization (MOP) problem can be defined as the following:

$$\begin{aligned} \min \quad & f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\} \\ s.t. \quad & x \in S \end{aligned} \tag{1}$$

where $k \geq 2$, $x = (x_1, x_2, \ldots, x_n)$ is the decision variable vector, $S$ is the feasible solution space, $f(x)$ is the objective vector.

A MOP solution is a set of the non-dominated solutions called the Pareto Set (PS).

**Definition 1** A solution ($x^* \in S$) dominates a solution ($x \in S, x \neq x^*$), $x^* \succ x$ if and only if $\forall i \in \{1,2,\ldots,n\}$ $f_i(x) \leq f_i(x^*)$ and $\exists i \in \{1,2,\ldots,n\}$ $f_i(x) < f_i(x^*)$.

**Definition 2** A solution ($x^* \in S$) is efficient if there do not exist a solution ($x \in S$) that dominates it.

**Definition 3** A solution $x^* \in S$ is weakly efficient if there does not exist a solution ($x \in S, x \neq x^*$) such that $f_i(x) < f_i(x^*)$.

Weighted $L_p$ norms are defined as the following (Hansen and Jaszkiewicz, 1998):

$$L_p(z^1, z^2, \Lambda) = (\sum_{j=1}^{J} \lambda_j \left| z_j^1 - z_j^2 \right|^p)^{1/p} \quad p \in \{1,2,\ldots\} \tag{2}$$

where $\Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_J], \lambda_j \geq 0$, is a weight vector.

For a multi-objective TSP, the general weighted $L_p$ norm is defined as

$$\begin{aligned} \min \quad & (\sum_{j=1}^{J} \lambda_j \left| f_j - z^* \right|^p)^{1/p} \\ & s.t. \quad x \in S \end{aligned} \tag{3}$$

where $\lambda_j \geq 0, j = 1,2,\ldots,J, \sum_{j=1}^{J} \lambda_j = 1, z^*$ is the reference point.

If we set the reference point as the global optimal solution for $f_j$, when $p = \infty$, we get the weighted Tchebycheff metric

$$\begin{aligned} \min \quad & \max\{\lambda_j (f_j - z^*)\} \\ s.t. \quad & x \in S \end{aligned} \tag{4}$$

When $p = 1$, we get the weighted sum function

$$\begin{aligned} \min \quad & \sum_{j=1}^{J} (\lambda_j (f_j - z^*)) \\ s.t. \quad & x \in S \end{aligned} \tag{5}$$

Multi-objective TSPs belong to the class of NP-complete problems even with only two objectives (Hansen, 2000). It has the difficulties of both the TSPs and the multi-objectives (Ehrgott, 2000). And even a single objective TSP belongs to the class of NP-hard problems.

There are many heuristics and meta-heuristics proposed to attack on multi-objective combinatorial optimization problems. Among these, Jaszkiewicz (Jaszkiewicz and Czyzak, 1998) proposed a Pareto simulated annealing algorithm to solve multi-objective combinatorial optimization problems. Jaszkiewicz (2002) presented a new genetic local search algorithm to solve multi-objective TSP. And he concluded that a local search guided by weighted linear function gave a better solution than guided by weighted Tchebycheff function. Hansen (2000) proposed a tabu search with a local search heuristic to solve multi-objective TSPs. He suggested that the heuristic gives better solution when using a substitute scalarizing function instead of the Tchebycheff function to guide the local search heuristic. Samanlioglu et al. (Samanlioglu, Ferrell & Kurz, 2008) present a memetic random key genetic algorithm embedded with a 2-opt heuristic to solve multi-objective TSP. In his research, the local search is guided randomly by either a weighted Tchebycheff function or a weighted sum function.

The evolutionary multi-objective optimization methods can be classified into two types: the Pareto-based technique and non Pareto-based technique (Samanlioglu, Ferrell and Kurz, 2008). For the Pareto-based techniques, the selection is directed by the Pareto dominance and Pareto ranking. The multi-objective genetic algorithm proposed by Fonseca and Fleming (1993), the niched Pareto genetic algorithm (Horn, Nafpliotis & Goldberg, 1994), NSGA II (Deb et al., 2002), SPEA II (Zitzler, Laumanns & Thiele, 2001), etc. belong to this type of technique.

In the non Pareto-base techniques, the selection does not directly rely on the Pareto dominance and Pareto ranking, like vector evaluated genetic algorithm (Schaffer, 1984), target vector approach (Coello, 2001), memetic random key genetic algorithm (Samanlioglu, Ferrell & Kurz, 2008), etc.

According to (Samanlioglu, Ferrell & Kurz, 2008), the main advantage of Pareto-based techniques is that these methods don't need to normalize objective functions, set reference points and specify weighting coefficients for each objective function according to its importance. But the Pareto-based techniques also have some disadvantages. First, the Pareto ranking does not work for hybrid meta-heuristics with local search because many local moves do not influence the rank of a solution. In some cases, change of a rank of a solution may need to change the objective function value a lot. But this may not be achieved by local move. And for solutions which have been already ranked 1, local improvement is not possible (Jaszkiewicz, 2002). Another problem with the Pareto-based techniques is with the comparability of the solutions. According to (Knowles and Corne, 2004), the Pareto-based techniques may be suited for problems with only two or three objective functions. When working on the multi-objective optimization problems with four or more objective functions, the Pareto-based technique may cause many problems because many solutions will be incomparable.

## 2.3 Meta-heuristics

Meta-heuristic is an active field of research with more and more new methods and the applications to specific problems being proposed. Some well-known metaheuristics are: evolution programming, genetic algorithm, simulated annealing, tabu search, artificial intelligence, ant colony algorithms, particle swan optimization and so on. Wolpert and Macready (Wolpert and Macready, 1997) explored the relationship between effective optimization algorithm and the problems they solve. They presented a number of “no free lunch” theorems which state that “for any algorithm, any elevated performance over one class of problems is offset by performance over another class.” So no optimization method is perfect enough to solve all optimization problems efficiently.

### 2.3.1 Genetic Algorithm

A genetic algorithm is an iterative procedure used to find exact or approximate solutions to optimization problems. It is categorized as a global search method. Genetic algorithm was first used by John Holland (1975). Genetic algorithm took clues from nature: genetic inheritance and Darwinian strife for survival. One distinguished character of genetic algorithm is the separation of the representation of the problem from the variables in which it was originally formulated.

Five general components are required in a genetic algorithm for a particular problem (Michalewicz, 1996):

- a genetic representation of the solution
- a way to create initial solutions

- a fitness function to evaluate the solution
- genetic operators to create offspring
- values for parameters (population size, probability of applying genetic operators, etc.)

In a genetic algorithm the representations (chromosomes or genotype) of candidate solutions (phenotypes) to an optimization problem evolve toward better solutions. Traditional genetic algorithm uses binary strings as a chromosome to represent real values of the decision variables (Holland, 1975). Suppose that the decision variable x takes values from a domain ($D = [a, c] \subseteq R$) and eight decimal places for the variable's value is required. Obviously, the domain should be cut into $(c - a) \cdot 10^8$ equal size range to achieve such precision. A representation having the variable coded as a binary string of length n satisfies the precision requirement, if n is the smallest integer which satisfies $(c - a) \cdot 10^8 \le 2^n - 1$ . And the following function shows how to convert a binary string $(b_1 b_2 \cdots b_n)_2$ into a real number x:

$$x = a_1 + (b_1 b_2 \cdots b_n)_2 \cdot \frac{c - a}{2^n - 1}$$

(6)

Genetic algorithm starts from a population of randomly selected individuals. In each generation, the fitness of every individual is evaluated, certain individuals are selected from the current population and modified (through the process of recombination and mutation) to form a new population of the next generation. Theoretically a genetic algorithm can run forever because it is a stochastic research method. In practice, the method stops when a certain termination criterion is reached. The criteria are: a satisfied solution is found, a fixed number of generations are reached, the allocated budget is reached, and better solutions are no longer produced.

Pseudo codes for a genetic algorithm are:

1. Randomly generate a population of individuals.
2. Compute and save the fitness for each individual in the current population.
3. Repeat until a stopping criteria is met
    - Select best ranked individuals to reproduce
    - Breed new generation through recombination and mutation
    - Evaluate the fitness of the offspring
    - Replace the worst ranked individuals with the offspring
4. Get the individuals with the highest global objective fitness.

Selection is the process in which the fittest individuals of the current generation are used to form the next generation. In (Michalewicz, 1996), a roulette wheel with slots sized according to fitness of each solution is proposed as follows:

- ➢ Calculate the total fitness of the population

$$F = \sum_{i=1}^{n} eval(v_i) \tag{7}$$

- ➢ Calculate the probability of selection $p_i$ for each solution $v_i$

$$p_i = \frac{eval(v_i)}{F} \tag{8}$$

- ➢ Calculate the cumulative probability $q_i$ for each solution

$$q_i = \sum_{s=1}^{i} p_s \tag{9}$$

Then we spin the roulette wheel n (n is the population size) times. Each time we select a solution for the next generation as follows:

- Generate a random number $r$ from the range $[0..1]$
- If $r < q_1$, select the first solution $v_1$; otherwise, select the $i$ -th solution $v_i$ if $q_{i-1} < r \leq q_i$

There are two genetic operators in genetic algorithm: crossover and mutation. Crossover (or recombination) is a genetic operator used to vary the programming of chromosomes from the two individuals of the fittest to form the next generation. Traditional genetic algorithm applies single point crossover. It involves the following steps (Beasley, Bull & Martin, 1993):

- Select two individuals that will exchange certain bits of their binary string with each other.
- Get the randomly selected crossover point and cut both of two individuals into two parts according to the crossover point.
- Swap over the two individuals to produce new binary strings (see figure 2.1).

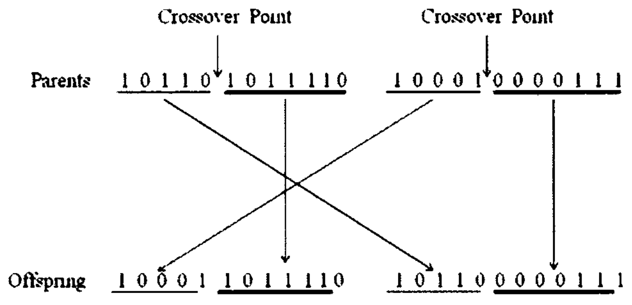


Figure 4. Single Point Crossover

Mutation is a genetic operator used to maintain genetic diversity from one generation to the next generation. Mutation provides a small amount of random search and ensures that no point in the search space has a zero probability to be searched (Beasley, Bull & Martin, 1993), thus improving the genetic algorithm's exploring ability. It involves a probability that an arbitrary bit in a genetic sequence changes from its original state. The normal procedure for a traditional genetic algorithm is: generate a random number from the range [0..1] for each bit of each solution; if this number is less than a predetermined number called mutation rate, the bit will change from 0 to 1, or from 1 to 0 (see figure 2.2).

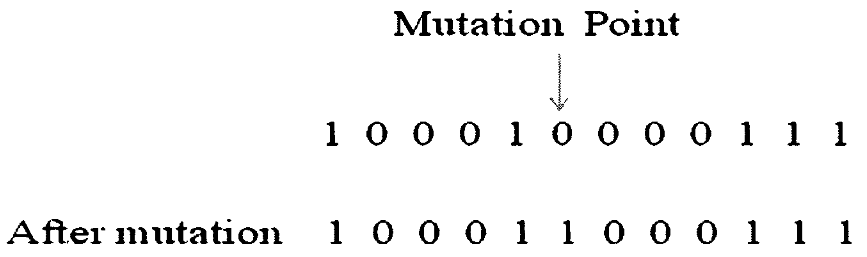


Figure 5. A Single Point Mutation

The strengths of the genetic algorithm are: the ability to evaluate many possible solutions simultaneously, easy implementation, good performance over a large number of problems (robust), good performance on NP-complete problems and the ability to be implemented on parallel processing (Choy, Lam & Lau, 1997-98).

The weakness of the genetic algorithm: sometimes converges towards local optima, has difficulty operating on dynamic data sets, cannot effectively solve problems with only single right/wrong measure fitness function, raises differences of opinion concerning the importance of crossover versus mutation and for specific problems, and other optimization algorithms may find better solutions than genetic algorithms.

## 2.3.2 Random Keys Genetic Algorithm

It is difficult for traditional Genetic Algorithm to maintain feasibility from parents to offspring when solving many optimization problems, like multiple machine scheduling, quadratic problem, traveling salesman problem, etc. (Bean, 1994). For example, Crossover (or recombination) is a genetic operator used to vary the programming of chromosomes from the two individuals of the fittest to form the next generation. Traditional Genetic Algorithm uses one point crossover (Holland, 1975). Suppose we have a traveling salesman problem that has 6 cities. A candidate for this problem is a permutation of these 6 cities. Two such permutations are 1→ 3 →2→4→5→6 and 6→ 4 →2→3→1→5. In traditional Genetic Algorithm, the genetic representation of these two sequences are the permutations $x = (1,3,2,4,5,6)$ and $x = (6,4,2,3,1,5)$ . A one point crossover will divide each permutation at the crossover point and exchange certain segments of the two permutations. Suppose the crossover point is the fourth place of the permutation. From the Figure 1, we can see that the resulting sequences are 1→ 3 →2→4→1→5 and 6→ 4 →2→3→5→6. Both of them are infeasible. So the crossing over of two feasible solutions does not result in feasible solutions.

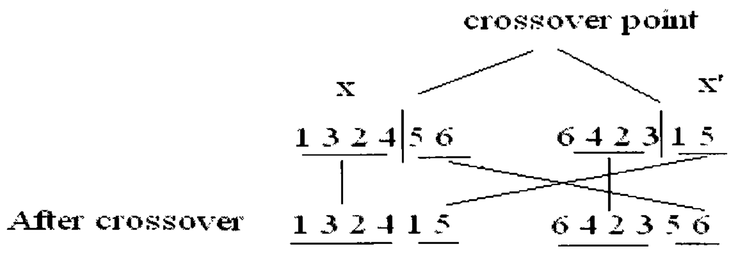


Figure 6. Single Point Crossover in Traditional Genetic Algorithms

To overcome the difficulty in maintaining feasibility from parent to offspring, Bean (1994) proposed the Random Keys Genetic Algorithm. Not like traditional Genetic Algorithm using direct chromosomal representation of the candidate solution, the Random Keys Genetic Algorithm uses chromosomal representation in a soft manner. It encodes the candidate solution with random numbers. The values of these random numbers range from 0 to 1. The random numbers are used as the sort keys to encode the candidate solutions. This encoding technique will eliminate the feasibility problem. For example, suppose that there is a single machine 6-jobs scheduling problem. The chromosome (0.45, 0.32, 0.58, 0.74, 0.65, 0.17) would represent the sequence 6→ 2 →1→3→5→4. In Random Keys Genetic Algorithm, the crossovers are executed on the chromosomes while not the sequence. Suppose that we have another chromosome (0.87, 0.66, 0.25, 0.14, 0.49, 0.94) that would represent the sequence 4→ 3→5→2→1→6. And we suppose the crossover point is after the fourth gene. By using the traditional single point crossover, we get the two offspring (see Figure 2): (0.45, 0.32, 0.58, 0.74, 0.49, 0.94) that represents the sequence 2 →1→5→3→4→ 6 and (0.87, 0.66, 0.25, 0.14, 0.65, 0.17) that represents the sequence 4→6→3→5→2→1. Both of the offspring are feasible solutions.

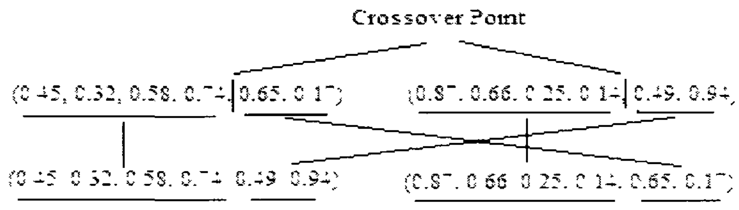


Figure 7. One Point Crossover for Random Keys Genetic Algorithms

Bean (1994) used the Random Keys Genetic Algorithm to solve the multiple machine scheduling problems, the resource allocation problem and the quadratic assignment problem to test its effectiveness. Chatterjee, Carrera and Lynch (1996) solved traveling salesman problems by using Random keys Genetic Algorithm. Snyder and Daskin (2006) presented a Random Keys Genetic Algorithm to solve generalized Traveling Salesman Problems. Samanlioglu et al. (Samanlioglu, Ferrell & Kurz, 2008) proposed a random-key genetic algorithm to solve the multi-objective traveling salesman problem.

While random key genetic algorithm can overcome the difficulty of maintaining feasibility from parents to offspring, there are two main difficulties with it (Chatterjee, Carrera & Lynch, 1996). First, when implemented on large problem, the sorting process to determine the ranks is expensive. Moreover, the encoding process does not preserve the adjacency of cities in a given tour when crossover with random keys. This makes the random key genetic algorithm very slow and inefficient.

### 2.3.3 Simulated Annealing

Simulated annealing is a generic probabilistic metaheuristic for the global optimization problem. It was derived from the annealing process of metals in which the crystalline configurations are dependent on the rate of the cooling process. A common use of simulated annealing is to find near globally minimum cost solutions to large optimization problems. Kirkpatrick, Gelatt and Vecchi (Kirkpatrick, Gelatt and Vecchi, 1983) were the first to come up and demonstrate applications of simulation techniques in a wide range of fields, from statistical physics to problems of combinatorial optimization.

At each step of the simulation algorithm a new state is constructed from the current state by giving random displacement to a randomly selected particle. If the energy of the new state is lower than that of the current state, then the displacement was accepted, therefore, the new state became the current state. However, if the energy of the new state was higher by m joules it became the current state with probability

$$\exp\left(\frac{m}{kT}\right).$$

(10)

This basic step can be mentioned indefinitely. The procedure was called *metropolis loop.* The generation of current states by applying this method led to a states distribution in which the probability of a given state with energy $e_i$ to the be the current state was

$$\frac{\exp(-e_i/kT)}{\sum_j \exp(-e_j/kT)} \qquad (11)$$

This is called the Boltzmann distribution, where k is Boltzmann's constant and T is the temperature.

Each iteration of the search process of simulated annealing moves from the current trial solution to an immediate neighbor. The selection of that immediate neighbor (candidate to be next trial solution) depends on certain rules, which represent the fundamentals of the simulated annealing search process. Those fundamentals are:

$Z_c$ = objective function value for the current trial solution

$Z_n$ = objective function value for current candidate to be the next trial solution

T = Temperature, a parameter that measures the tendency to accept the current candidate to be the next trial solution if this candidate is not an improvement on the current trial solution.

Now we will discuss the rule in which we apply those fundamental concepts for the selection of one of the immediate neighbors. Assuming the objective is *minimization*; the simulated annealing search process accepts or rejects a candidate solution to be the next trial solution as follows:

If $Z_n \leq Z_c$ always accept this candidate

If $Z_n > Z_c$ accept the candidate with the following probability:

Probability (Acceptance) = $\exp((Z_c - Z_n) / T_1)$

In the case of having a worse solution ($Z_n > Z_c$ in case of minimization), a probability of acceptance is introduced. This probability is compared to a random number (between 0 and 1) to determine if the current candidate solution will become the next trial solution.

If *random number* < Probability (Acceptance), accept the current candidate solution. Otherwise, reject.

The application of simulated annealing to optimization problems requires some preliminary steps:

1. Identify the analogues of the physics concepts in the optimization problem at hand.
    - Energy function becomes objective functions.
    - Particles configurations become parameter values configurations
    - Finding a low-energy configuration becomes looking for a near-optimal solution.
    - Temperature becomes the control parameter for the process.
2. Select an annealing schedule that consists of lowering a set of temperatures together with the amount of time that should be spent at each temperature.
3. Develop a way of generating and selecting new configurations.

Any optimization technique has strengths and weaknesses. Simulated annealing is no exception to that. The relative straightforwardness and the ability to solve many combinatorial solutions is an important strength of simulated annealing, however, there is always a need for long computer processing times. Table 2.1 lists some strengths and weaknesses of simulated annealing presented by research done in Antwerp University, Belgium.

| Strengths | Weaknesses |
|---|---|
| 1. Can deal with arbitrary systems and cost functions.<br>2. Statistically guarantees finding an optimal solution.<br>3. Is relatively easy to code, even for complex problems.<br>4. Generally gives a good solution. | 1. Repeatedly annealing with a schedule is very slow, especially if the cost function is expensive to compute.<br>2. For problems where the energy landscape is smooth, or there are few local minimum values, SA is overkill – simpler, faster methods will work better.<br>3. Heuristic methods, which are problem-specific or take advantage of extra information about the system, will often be better than general methods. But simulated annealing will often be comparable to heuristics.<br>4. Simulated annealing cannot tell whether it has found an optimal solution. |

Table 1. Advantages and Disadvantages of Simulated Annealing

## 2.3.4 Tabu Search

Tabu search is a metaheuristic which belongs to the class of local search techniques. Fred Glover (1989) is regarded as the one who created the method. According to Webster's dictionary, tabu is defined as "forbidden to profane use or contact because of what are held to be dangerous or supernatural powers" or "banned on grounds of morality or taste" or "banned as constituting a risk."

By using tabu, the method guides a local heuristic search procedure to explore the solution space beyond local optimality, thus improving the performance of the procedure. Glover and Laguna (1997) mentioned that "distinguished feature of Tabu Search is embodied in its exploitation of adaptive forms of memory, which equips it to penetrate complexities that often confound alternative approaches." They thought adaptive memory and responsive exploration are the general tenets of Tabu Search. Hertz, Taillard and Werra (1997) defined the Tabu Search procedure:

1. Choose an initial solution i in S. Set i*=i and k=0.
2. Set k = k + 1 and generate a subset $V^*$ of solution in $N(i,k)$ such that either one of the tabu conditions $tr(i,m) \in Tr$ is violated (r=1,...,t) or at least one of the aspiration conditions $ar(i,m) \in Ar(i,m)$ holds (r=1,...,a).
3. Choose a best $j = i \oplus m$ in $V^*$ (with respect to f or to the function f+) and set i=j.
4. If $f(i) < f(i^*)$ then set $i^* = i$.
5. Update tabu and aspiration conditions.
6. Stop if a stopping condition is met. Else go to Step 2.

They also defined the stopping criteria as:

1. The next iteration does not yield any solutions in the new neighborhood, $N(i,k+1) = \varnothing$.
2. Perform a set number of iterations, k.
3. The objective reaches a pre-specified threshold value.
4. Evidence can be given that a global optimum has been reached.

And because the aspiration criteria can allow a move that would be otherwise forbidden, it can affect the performance of Tabu Search. Glover and Laguna (1997) gave a basic aspiration criteria "consisting of removing a tabu classification from a trial move when the move yields solution better than the best obtained so far."

The strengths of Tabu Search:

1. Allows a move to an inferior solution to escape local optimums
2. Steers away from unpromising inferior solutions
3. Can be applied to both discrete and continuous optimization problems

The weaknesses of Tabu Search (Hertz, Taillard & Werra, 1997):

1. Overwhelmed by the number of parameters to be defined
2. Overwhelmed by the number of iterations
3. The performance depends on the settings of the various parameters.
4. The objective function is hard to evaluate or may not provide enough information to effectively drive the search to a more interesting area of the search space.

## 2.3.5 Ant Colony Optimization

Ant colony optimization is a metaheuristic technique for solving hard combinatorial optimization problems, like the Traveling Sales Problem. Macro Dorigo (1992) originally proposed it in his PhD thesis in 1992. The inspiration came from the ants which took shortest path through the communication among the ants by laying and following the pheromone. The ant colony optimization is based on the communication among a colony of agents, called ants, mediated by pheromone trails.

Dorigo (Dorigo, 1992; Dorigo & Caro, 1999) defined the framework of the Ant Colony Algorithm.

- A finite set $C = \{c_1, c_2, \dots, c_{N_c}\}$
- $L = \{l_{c_i c_j} | (c_i, c_j) \in \tilde{C}\}, |L| \leq N_C^2$ is a finite set of possible connection among the elements of $\tilde{C}$ , where $\tilde{C}$ is a subset of the Cartesian product $C \times C$ .
- $J_{c_i c_j} \equiv J(l_{c_i c_j}, t)$ is a connection cost function associated with each $l_{c_i c_j} \in L$
- $\Omega \equiv \Omega(C, L, t)$ is a finite set of constraints assigned over the elements of C and L.
- $s = (c_i, c_j, \dots, c_k, \dots)$ is a sequence over the elements of C. A sequence S is also called a state of the problem. If S is the set of all possible sequences, the set $\tilde{S}$ of all the sequences that are feasible with respect to the

constraint $\Omega(C,L,t)$, is a subset of S. The elements in $\tilde{S}$ define the problem's feasible states.

- A neighborhood structure is defined as follows: Given two states $s_1$ and $s_2$, if both $s_1$ and $s_2$ are in S, and the state $s_2$ can be reached from $s_1$ in one logical step, $s_2$ is said to be a neighbor of $s_1$. The neighbor of a state s is denoted by $N_s$ .

- ψ is a solution if it is an element of $\tilde{S}$ and satisfies all the problem's requirements.

- $J_\psi(L,t)$ is a cost associated to each solution ψ.

Ants of the colony have the following properties:

- An ant searches for minimum cost feasible solutions $\hat{J_\psi} = min_\psi J_\psi(L,t)$ .
- An ant k has a memory of $M^k$ that is used to store information on the path it followed so far. Memory is used to build a feasible solution, to evaluate the solution and to retrace the path backward.

- An ant k in state $s_r = (s_{r-1}, i)$ can move to any node j which is in its feasible neighborhood $N_i^k$ . The move is selected through a probabilistic decision rule which is the function of the values stored in a node local date structure that is obtained by a functional composition of node locally available pheromone trails and heuristic values, the ant's memory and the problem constraints.
- An ant k can be assigned a start state $s_s^k$ and one or more termination conditions $e^k$ . Ants start from the start state and move to a feasible neighbor state until at least one of the termination conditions is satisfied to build the solution in an incremental way.
- When ants move from node $i$ to neighbor node j, they update the pheromone trail $\tau_{ij}$ on the arc$(i,j)$. This is called online step-by-step pheromone update.
- Once a solution is built, the ant can retrace the same path backward and update the pheromone trails on the traversed arcs. This is called online delayed pheromone update.
- Once it has built a solution and it has retraced the path back to the source node, the ant dies and frees all the allocated resources.

Besides ants' activities, an ant colony optimization algorithm includes two additional procedures: pheromone trail evaporation and daemon actions. Pheromone evaporation is the process through which the pheromone deposited by previous ants decreases over time. It can avoid a too-rapid convergence to a suboptimal region. Daemon actions can be used to implement centralized actions which can not be performed by a single ant. The daemon can observe the path formed by each ant in the colony and choose to deposit extra pheromone on the components used by the ant that built the best solution. The pseudo code (Glover & Kochenberger, 2002) for ant colony optimization is in Figure 3.

```
Procedure ACO metaheuristic
    If not termination
        AntsActivity ( )
        PheromoneUpdate ( )
      DameonActions ( )
  End if
End Procedure
```

Figure 8. The Pseudo Code for Ant Colony Optimization

When the ant k is at the city i at time t, the probability that it will move to the city j is

$$p_{ij}^{k}(t) = \begin{cases} \dfrac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}}{\sum_{l \in N_i^k}[\tau_{il}(t)]^{\alpha}[\eta_{il}(t)]^{\beta}} & if\ j \in N_i^k \\ 0 & otherwise \end{cases}$$

(12)

Where

$\tau_{ij}$ is the amount of pheromone on edge$(i,j)$.

α is a parameter which controls the influence of$\tau(i,j)$.

$\eta_{ij}$ is the desirability of edge$(i,j)$. In traveling sales problem, $\eta_{ij} = 1/d_{ij}$, $d_{ij}$ is the distance between i and j.

$\beta$ is a parameter which controls the influence of $\eta_{ij}$.

In this way, we will favor the choice of edges which are short and have a greater amount of pheromone. The pheromone update is done by the following function:

$$\tau_{ij}(t+1) = (1-\rho)\cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t) \qquad (13)$$

Where $0 \leq \rho \leq 1$ is the pheromone trail evaporation rate. m is the number of ants. The parameter $\rho$ is used to avoid unlimited accumulation of the pheromone trails and let the algorithm forget the previous bad solution.

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{1}{L^{k}(t)} & \textit{if edge } (i,j) \textit{ is used by ant } k \\ 0 & \textit{otherwise} \end{cases} \qquad (14)$$

where $L^{k}(t)$ is the length of the $k$ th ant's tour.

Ant colony optimization can be applied to a wide range of combinatorial optimization problems. It is mainly used in these two fields: NP-hard problems and shortest path problems in which the properties of the problems' graph representation change over time. The ant colony optimization has an advantage over simulated annealing and genetic algorithm when the graph may change dynamically. It can be run continuously and adapt to changes in real time.

The first problem with ant colony optimization is difficult in definition. It is not easy to give a precise definition of what algorithm is or is not ant colony because the definition may change according to the authors and uses. Another weakness is the search may fall into a local optimum.

### 2.3.6 Particle Swarm Optimization

Particle swarm optimization is a population-based evolutionary computation algorithm for solving optimization problems. It was developed by Kennedy and Eberhart (Kennedy and Eberhart, 1995). It is derived from the research and simulation of the social behavior of a bird flock. Particle swarm optimization is similar to genetic algorithm in that it is also initialized with a population of random solutions. The difference between them is that in particle swarm optimization, each particle is also assigned a randomized velocity so it flows through the problem space (Eberhart and Shi, 2001).

Each particle keeps track of its coordinates in the problem space. The coordinates are associated with the best solution it has achieved so far. This value is called pbest. Another best value tracked is the overall best value and its location obtained by any particle in the population so far. It is called gbest.

Eberhart and Shi (2001) defined the process for implementing the global version of particle swarm optimization in the following:

1. Initialize a population of particles with random position and velocity in the problem space.
2. Evaluate each particle's desired optimization fitness function.
3. Update each particle's pbest and its location. If current value is better than pbest, set pbest value equal to the current value and its location equal to the current location.
4. Update gbest. If current value is better than gbest, then set gbest to the current value.
5. Change the velocity and position of each particle according to :

   i. $V_i(t+1) = wV_i(t) + C_1 rand_1(\,)(\hat{x}_i - x_i) + C_2 rand_2(\,)(\hat{g} - x_i)$ (15)

   ii. $X_i(t+1) = X_i(t) + V_i(t+1)$ (16)

   where $w$ is inertia weight, $C_1$ and $C_2$ are acceleration constants, $rand_1(\,)$ and $rand_2(\,)$ are two different random function range from 0 to 1. $\hat{x}_i$ is the location of each particle's local best value (pbest), $\hat{g}$ is the location of global best value (gbest).

6. Loop to step 2 until a criterion is met. The criterion is usually a sufficiently good fitness or a maximum number of iterations.

Particle swarm optimization has been used in a wide range of applications. Like other evolutionary optimization methods, particle swarm optimization can be applied to solve most optimization problems.

One of the reasons that particle swarm optimization is attractive is that there are very few parameters to adjust and it is easy to implement. Wang, Zhang, Zhou and Yin (2008) summarized that the advantages of particle swarm optimization are: simple structure, immediate accessibility for practical application, ease to implementation, quick convergence and robustness. But there are still problems with it. First its applications in solving global combinatorial optimization are limited and not as effective as in global continuous optimization. And on the other hand, the search in particle swarm optimization is mainly based on the local information. It is based on each particle's own best position information and the best global position information so far; therefore, the particle swarm optimization has no mechanism to get and use global information about the search space.

## 2.3.7 Harmony Search

Harmony search is a new heuristic algorithm which mimics the improvisation process of music player. Geem and Kim (2001) first discussed this new metaheuristic algorithm. Musical performers seek to find harmony, which is determined by an aesthetic standard, just as the optimization process seeks to find an optimal solution which is determined by an objective function. And still the pitch of each musical instrument determines the aesthetic quality; just as each decision variable determines the value of the objective function. In the algorithm, the harmony is analogous to the optimization solution and the improvisations are analogous to local and global search schemes (Lee & Geem, 2005). When a musician improvises one pitch, he usually chooses one of three options: 1) playing any one pitch from his memory; 2) playing an adjacent pitch of one pitch from his memory; 3) playing a totally random pitch from the possible sound range. Similarly, in a harmony search algorithm, each decision variable follows one of three rules to choose one value: 1) choosing any one value from the harmony search memory, 2) choosing an adjacent value of one value from the harmony search memory, 3) choosing totally random from the possible value range.

Lee and Geem (2005) defined the process of harmony search optimization as follows:

1. Initialize the optimization problem and parameters.
2. Initialize the harmony memory (HM) by randomly generating solution vectors and sorting by the values of the objective function. $x^1, x^2, \cdots, x^{HMS}$ ($HMS$ is the number of solution vectors in harmony memory.).
3. Improvise a new harmony from the HM. The new harmony vector, $x' = (x'_1, x'_2, \cdots, x'_N)$ is generated based on memory considerations, pitch adjustments and randomization.

$$x'_i \leftarrow \begin{cases} x'_i \in \{x_i^1, x_i^2, \cdots, x_i^{HMS}\} & with\ probability\ HMCR \\ x'_i \in X_i & with\ probability\ (1 - HMCR) \end{cases}$$

(17)

The HMCR is the probability of choosing one value from the historic values stored in the HM, while $(1 - HMCR)$ is the probability of randomly selecting one feasible value not limited to those stored in the HM. Each component of the new harmony vector, $x' = (x'_1, x'_2, \cdots, x'_N)$, is examined to determine if it should be pitch-adjusted.

Pitch adjusting decision for $x'_i \leftarrow \begin{cases} Yes \ with\ probability\ PAR \\ No \ \ with\ probability\ (1 - PAR) \end{cases}$

(18)

The pitch adjusting is only performed after a value is chosen from the HM.

$$x'_i \leftarrow x'_i \pm bw \cdot rand(0,1) \tag{19}$$

$bw$ is the distance bandwidth, the amount of maximum change for pitch adjustment.

The value $(1 - PAR)$ sets the rate of doing nothing.

4. Update the HM. If the harmony vector is better than the worst harmony in the HM in terms of the objective function value, the harmony replaces the worst harmony to be put into HM.
5. Repeat steps 3 and 4 until the termination criterion is satisfied.

Harmony search is a global search algorithm which can be easily applied to various optimization problems. The advantages of the harmony search are: when making a new vector, it considers all existing vectors rather than only two parents like the genetic algorithm. And harmony search does not require setting initial values for the decision variables. These advantages may help it in escaping local optima and finding better solutions. Harmony search gets into trouble when performing a local search for numerical application (Mahdavi, Fesanghary & Damangir, 2007).

## 2.4 Hybrid Meta-heuristics

Over the years, research on hybrid metaheuristics has risen considerably in combinatorial optimization. This research attempted to combine the best features from different metaheuristics to develop more powerful hybrid implementations than the original metaheuristics. The most common format of the hybrid metaheuristics is: the population-based metaheuristics, like genetic algorithm, ant colony, particle swarm etc., which are powerful in exploring the solution space were combined with local search metaheuristics, like hill climbing, simulated annealing, tabu search etc., which are more powerful in terms of exploitation to develop more powerful hybrid metaheuristics (Suh & Van Gucht, 1987; see also Fleurent & Ferland, 1994; Kim, Hayashi & Nara, 1995; Chen & Flann, 1994; Chen et al., 1995; Javadi and Tan, 2005; Li and Li, 2008).

According to (Raidl, 2006), the motivation behind hybrid metaheuristics is to obtain better performance metaheuristics that exploit and unite advantages of the individual pure metaheuristics.

Talbi (2002) categorized all hybrid metaheuristics into four hierarchy categories: Low–level relay hybrid metaheuristic, low-level teamwork hybrid, high-level relay hybrid and high-level teamwork hybrid.

In low-level relay hybrid metaheuristics, a metaheuristic is embedded into a single solution metaheuristic. For example, simulated annealing was combined with local search to solve the traveling salesman problem (Martin, Otto & Felten, 1992).

A metaheuristic embedded into a population-based metaheuristic forms low-level teamwork hybrid metaheuristic. For example, local search metaheuristics, like tabu search, simulated annealing, etc., had been embedded into genetic algorithm to form this kind of hybrid metaheuristics (Fleurent & Ferland, 1994; see also Thiel & Voss, 1994; Kim, Hayashi & Nara, 1995; Davis, 1985; Chen & Flann, 1994; Chen, Wang, Kao, Ouhyang & Chen 1995).

High-level relay hybrid metaheuristics involve several metaheuristics executed in a sequence. For example, in (Javadi & Tan, 2005), a hybrid intelligent genetic algorithm based on the combination of neural network and the genetic algorithm was proposed. In this algorithm, a neural network is used to improve the convergence of the genetic algorithm.

In high-level teamwork hybrid metaheuristics, several metaheuristics conduct a search in parallel. For example, some searches proposed distributed genetic algorithms in which a fixed number of subpopulations evolve competing solutions. Each one of the subpopulations is processed independently by a genetic algorithm. An extra operator, called migration, is proposed to produce exchange between the subpopulations (Tanese, 1989; see also Whitley & Starkweather, 1990; Sun & Wan, 1995; Herrera, Lozano & Moraga, 1999).

As a population based optimization method, genetic algorithm is powerful in exploring the solution space while weak in the exploitation of the solutions found. So genetic algorithms have been combined with local search heuristics that are powerful in exploitation to create more powerful hybrid metaheuristics.

There are several types of hybrid metaheuristics concerning genetic algorithms. Some hybrid metaheuristics are this type: a local search metaheuristic is embedded into genetic algorithm. The genetic algorithm is used as a global optimizer while its recombination operators (mutation and crossover) are augmented with the ability to perform a local search. Not like classical genetic algorithm using blind operators regardless of fitness of the original individual and the operated one, the hybrid genetic algorithms use heuristics as operator which consider an individual as the origin of its search, apply itself to the individual and replace the original individual with the enhanced one. The local search heuristics here can be hill climbing (Suh & Van Gucht, 1987; Jog, Suh & Van Gucht, 1989) , tabu search (Fleurent & Ferland, 1994; see also Thiel & Voss, 1994; Kim, Hayashi & Nara, 1995), greedy heuristics ( Davis, 1985), simulated annealing (Chen & Flann, 1994; Chen, Wang, Kao, Ouhyang and Chen 1995).

Another type of hybrid genetic algorithm is: genetic algorithm and other metaheuristics are executed in a sequence. As a population based optimization method, genetic algorithm is powerful in exploring the solution space. This means that genetic algorithm can quickly locate the high performance regions of the solution spaces. Once these high performance regions are located, it will be useful to use a local search metaheuristic to exploit these regions.

According to Talbi (2002), after a certain amount of time, the population of genetic algorithm is quite uniform. Thus, the process fell into a basin of attraction from which it has a low probability to escape. It will improve the performance of the metaheuristic to exploit the basin to get the optimal solution in the basin as efficient as possible. Because genetic algorithm is not good in exploitation, it will be efficient to use a local search metaheuristic, like hill climbing, tabu search, simulated annealing, etc. Mahfound and Goldberg (1995) used simulated annealing as local search metaheuristic to improve the population obtained by a genetic algorithm to get a more powerful hybrid metaheuristic. Nissen (1994) introduced hill climbing as a local search heuristic to improve the results obtained by genetic algorithm. In (Javadi & Tan, 2005), a hybrid intelligent genetic algorithm based on the combination of neural network and the genetic algorithm was proposed. In this algorithm, a neural network is used to improve the convergence of the genetic algorithm.

Another direction is using other metaheuristics, like greedy search, simulated annealing, etc., to generate initial population for genetic algorithm. For example, Lin, Kao and Hsu (1991) proposed a hybrid genetic algorithm which incorporated genetic algorithm into simulated annealing. This hybrid genetic algorithm started with simulated annealing and used genetic algorithm to augment the solution founded by simulated annealing.

Still another type of hybrid genetic algorithm is: several genetic algorithms perform a search in parallel. Potter and De Jong (1994) proposed a cooperative co-evolutionary genetic algorithm. In this algorithm, there are multiple interacting species and each species represents a subcomponent of a potential solution. And the evolution of each species is handled by a standard GA.

Some searches proposed distributed genetic algorithms in which a fixed number of subpopulations evolve competing solutions. Each one of the subpopulations is processed independently by a genetic algorithm. An extra operator, called migration, is proposed to produce exchange between the subpopulations (Tanese, 1989; see also Whitley & Starkweather, 1990; Sun & Wan, 1995; Herrera, Lozano & Moraga, 1999). In (Li & Li, 2008), Li and Li proposed a dual species genetic algorithm in which two subpopulations with the same size individuals have different characteristics, such as crossover rate and mutation operator. One subpopulation has a higher crossover rate while the other has a higher mutation rate. So the new hybrid genetic algorithm has both good exploration and exploitation ability.

## 2.5 Lehmer Code

Lehmer code can effectively represent a permutation. It was proposed by Lehmer (1960).

Lehmer code can code each permutation $\Pi_n$ of n numbers with a function $LC(\Pi\ ):\{1,...,n\} \to \{1,...,n-1\}$ to a special sequence of n-1 numbers (Kromer, Platos & Snasel, 2009). Lehmer code of a permutation can be expressed by using an inversion table. Consider a sequence of n numbers $x=(x_1x_2...x_n)$ . An inversion is a pair $(x_i, x_j)$ such that $i<j$ and $x_i > x_j$. For $i \in \{1,...,n\}$, let $d_i$ count the number of inversions with i as the smaller index. Then the sequence $(d_1d_2...d_n)$ is called an inversion table of permutation x. $0 \le d_i \le n-i$ for $i=1,...,n$.

For example, the permutation (3 4 5 2 6 1) can be coded into (2 2 2 1 1 0) by Lehmer code. When the genetic algorithm uses Lehmer code to encode the solutions, the offspring created by crossover of the parent solutions are always feasible. Moreover, it can preserve some edge information from the parent solutions to the offspring.

In (Kromer, Platos & Snasel, 2009), Kromer et al. presented a Lehmer code genetic algorithm and compared it with the other encoding methods. Pesko (2006) proposed a differential evolution algorithm with Lehmer code encoding the candidate solutions to solve the Traveling Salesman Problem.

By using Lehmer code representation, the difficulty for traditional Genetic Algorithms to maintain feasibility from crossover parent solutions to offspring solution will be solved automatically. The solutions getting from crossover will be always feasible. The following figure shows how Lehmer code representation crossover works.

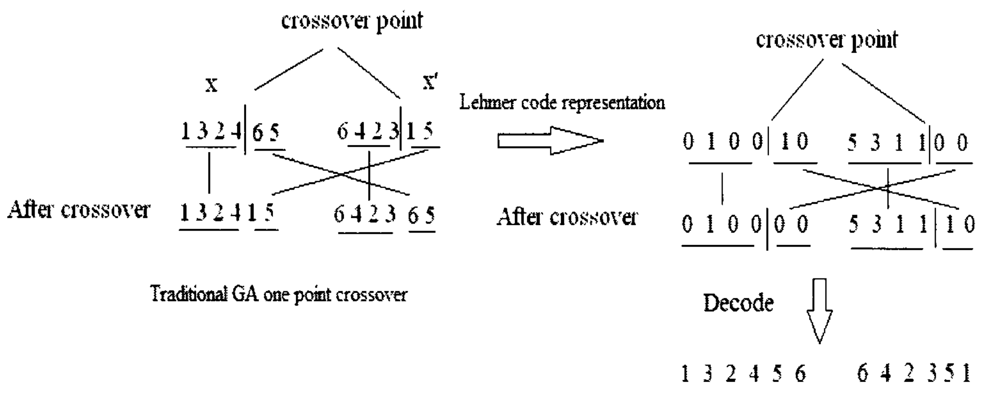


Figure 9. Lehmer Code One Point Crossover

From this figure, we can see that neither solution is feasible in the one point crossover of traditional GA. While in Lehmer code GA, both solutions getting from crossover of parent solutions which are the same with those in traditional GA are feasible. And we can see that with the Lehmer code representation, a certain part of the parent solutions information can be transferred to the offspring solutions. This means that a certain part of the schematic information can be reserved. As we know, this is one of the advantages that Genetic Algorithms has, while random keys Genetic Algorithms can't keep any edge information to transfer to the offspring. This can be one of the disadvantages of random keys Genetic Algorithms.

# CHAPTER 3

# METHODOLOGY

This research proposes developing a hybrid meta-heuristic consisting of combining genetic algorithms and local search heuristics, 2-opt heuristics and non sequential 3-opt heuristics. This method compensates for the weakness of a traditional genetic algorithm in exploitation while not hampering its ability in exploration. The new genetic algorithm will have both good exploration and exploitation ability because as a population based meta-heuristic, genetic algorithm is powerful in exploring the solution space and as local search heuristics, 2-opt heuristics and non sequential 3-opt heuristics are powerful in exploitation.

Lehmer code will be used to encode the candidate solutions to solve the infeasible solution problems for traditional Genetic algorithms brought by crossover when solving discrete optimization problems. By using Lehmer code representation, the solutions coming from crossover of parent solutions are always feasible solutions.

In this research, whenever a new solution was produced, no matter if it was produced by crossover of two solutions from the last generation or by mutation, 2-opt heuristics and/or non sequential 3-opt heuristics were used to improve this solution until a local optimal solution was obtained. The 2-opt heuristics and non sequential 3-opt heuristics were guided by a weighted sum of the objectives. The evaluation function was a weighted Tchebycheff metric with an ideal point.

The programming language, MATLAB, will be used to implement this new hybrid genetic algorithm.

## 3.1 Research Problems

There are several research problems that need to be addressed when creating the hybrid genetic algorithm. These include:

1. How to use the Lehmer code to encode the candidate solution?
2. What kind of local search heuristics will be used to perform a local search?
3. How to combine genetic algorithm and 2-opt heuristic?
4. How to evaluate the solutions?
5. How to test the performance of the new hybrid algorithm?

An approach to each of these research questions is given below.

**How to use the Lehmer code to encode the candidate solution?**

To overcome the difficulty in maintaining feasibility from parents to offspring for traditional genetic algorithms that use direct representation, Bean (1994) proposed random key genetic algorithms. But this encoding technique also has some disadvantages. For example, the sorting process to determine the ranks of the cities is time-consuming; the information about the adjacency cannot be preserved, this means that no schematics can be transferred from this generation to the next one.

In this research, I will use the Lehmer code to encode the solutions. For the initial solutions, I can definitely use Lehmer code to create the random permutations of n (n is the number of cities) to represent the initial solutions. But after the initial solutions are created, these solutions need to be evaluated, improved until local optimal solutions are obtained and sorted. So these solutions should be decoded to represent the path. Then these solutions need to be encoded by Lehmer code so the crossover and mutation operators will create feasible solution. And these coding and encoding processes will be expensive and time consuming. To make these processes simpler, this research will create the random permutations of n to represent the initial solutions directly using the natural representation. After these solutions are evaluated, improved and sorted, they will be encoded by Lehmer code to prepare for the crossover and mutation.

And as stated above, Lehmer code representation can not only guarantee the feasibility of the solutions getting from crossover of parent solutions, but also keep some parts of edge information. This makes it a better representation than random keys. Another possible advantage with Lehmer code representation in Genetic Algorithms will be the easy implementation of the genetic operator, mutation. This research will implement one point mutation.

To implement this mutation, first randomly select a number in [0, n-5] (n is the number of the cities). This number will indicate the position of the city that will mutate. For example, the number is j, the Lehmer code for this city is temp(j). Then create another random number in [0, 1]. This number will determine that the Lehmer code for selected city will be increased by 3 or be decreased by 3. If the random number is less than 0.5 and temp(j)-3>0 or temp(j)+3>n-j, the Lehmer code of the city will be changed to temp(j)-3. Or else, the Lehmer code for the selected city will be temp(j)+3.

**What kind of local search heuristics will be used to perform local search?**

There are many local search algorithms and their variants that we can select from. Therefore, we need to determine what kind of local search heuristics will be used in the new hybrid genetic algorithm. The idea is that we will use a local search algorithm to find local optimal solutions, a non-exhaustive list of local search techniques includes: nearest neighbor, greedy, 2-opt, 3-opt, k-opt, L-K, etc.

The 2-opt heuristic involves in deleting two edges from a tour and reconnecting the two paths created. Only if the resulting tour is shorter than the previous one will we make this change. Otherwise, we will keep the initial tour. This procedure will be continued until no improvement can be made. At this point, we can say a 2-opt local optimal solution is obtained.

According to (Joson & Mcgeoch, 1995), the 2-opt heuristic will often result in a tour with a length less than 5% above the Held-Karp bound. The 2-opt heuristic considers the pair-wise exchange. It involves selecting an edge $(c_1, c_2)$ and searching for another edge $(c_3, c_4)$, if $dis\tan ce(c_1, c_2) + dis\tan ce(c_3, c_4) > dis\tan ce(c_1, c_3) + dis\tan ce(c_2, c_4)$, the change will be accepted. This new solution will be served as the candidate solution to find another better solution until no better solution can be found. In the worst situation, each edge will be compared with the rest n-2 edges. Therefore, a simple implementation of 2-opt heuristic runs in $o(n^2)$. Although the other heuristics, like 3-opt, L-K opt, etc., can find better solutions than 2-opt, they are more complex than 2-opt. For example, a simple 3-opt runs in $o(n^3)$.

To take advantage of the exploitation ability of 3-opt heuristics and not increase CPU time too much, this research will also use non sequential 3-opt heuristics. We only picked up one of four possible 3-opt exchanges. After balancing the effectiveness and efficiency of all these local search heuristics, this research decides to choose 2-opt heuristics and non sequential 3-opt heuristics as the local search heuristic.

**How to combine genetic algorithm and 2-opt heuristics, non sequential 3-opt heuristics?**

As discussed above, there are many types of hybrid genetic algorithms, like embedding other local search meta-heuristic into genetic algorithm to augment the genetic operators of genetic algorithms, executing other meta-heuristics and genetic algorithm in a sequence, having distributed genetic algorithms, etc. In this research, 2-opt heuristics and non sequential 3-opt heuristics were embedded into genetic algorithms. Whenever a solution was gotten, 2-opt heuristics or non sequential 3-opt heuristics were used on this solution until a local optimal solution was obtained. For single objective TSPs, the 2-opt heuristics and non sequential 3-opt heuristics are guided by the objective values of the solutions. For multi-objective TSPs, 2-opt heuristics and non sequential 3-opt heuristics are directed by the weighted sum function.

**How to evaluate the solutions?**

In traditional genetic algorithm, solutions are evaluated according to the fitness values of the solutions. For single objective Traveling Salesman Problem, solutions are evaluated by the objective values of the solutions. For multi-objective TSP, each solution has multiple objective values. The goal is to find a set of the non-dominated solutions called the Pareto Set (PS). The evolutionary multi-objective optimization methods can be classified into two types: the Pareto-based technique and non Pareto-based technique.

This research will use non Pareto-based technique, Target Vector Approach, for multi-objective Traveling Salesman Problems. In Target Vector Approach, the goal is to minimize the distance between the generated solution and the target vector. And in this research, solution will be evaluated by the weighted Tchebycheff function with the ideal points as the reference points. The ideal points here are the optimal solutions for each objective function. And this research considers only fixed weight vector. To be more specific, the vector for each objective function is the same. For example, for a 4 objectives TSP, the weight for each objective function is 1/4.

**How to test the performance of the new genetic algorithm?**

To test the performance of the algorithm proposed by this research, first, the newly created algorithm will be used to work on some bench mark single objective TSPs. The bench mark single objective TSPs can be available at TSPIB (Reinelt, 1995). To be more specific, I will use this hybrid Lehmer code Genetic Algorithms to solve the TSPs: att48, kroA 100, kroB 100, kroC 100, kroD 100, kroE 100, kroA150, KroB150, KroA 200 and KroB 200. These problems have cities from 48, 100, 150 and 200. And their optimal solutions are known. All these make them good bench mark problems to test the performance of the heuristics and meta-heuristics like our algorithm.

And the performance of the new Genetic Algorithms will also be tested on solving multi-objective TSPs. I will compare the performance of my method on multi-objective TSPs with the methods proposed by Hansen (2002) and Samanlioglu et al. (Samanlioglu, Ferrell & Kurz, 2008). To make a good comparison, this research will test the performance of the algorithm proposed on the same Traveling Salesman problems as those used in (Hansen, 2000) and (Samanlioglu, Ferrell & Kurz, 2008). More specifically, I will use the set of Krolak instances with 100 cities from TSPLIB (Reinelt, 1995). The Krolak instances include 5 instances, kroA 100, kroB 100, ..., kroE 100. For the multi-objective TSP problem of this research, each instance correspond to the cost matrix of one objective function, for example, for a three objective TSP problem, kroA 100 corresponds to the cost matrix of objective function 1, kroB 100 corresponds to that of objective function 2, kroC 100 to that of objective function 3. There are two main advantages by doing this. First, the tours will be within the same scale of range for the different objectives. This makes the range scaling unnecessary. Furthermore, the optimal value to each problem is known. So we get the exact ideal point for each objective function. This makes the implementation of the weighted Tchebycheff function very easy.

Moreover, to make the comparison between the method of this research and the methods proposed by Hansen (2002) and Samanlioglu, Ferrell and Kurz (2008) valid, the method proposed by this research will work on TSP problems that have two objectives to up to five objectives and each problem will be run for 30 times.

## 3.2 Procedure and Pseudo Code

As mentioned above, the hybrid genetic algorithm will combine genetic algorithm and 2-opt heuristics and non sequential 3-opt heuristics.

The procedure for the new genetic algorithm:

1. Representation

This research will use the Lehmer code to represent the chromosome. Traditional genetic algorithm uses binary strings as a chromosome to represent real values of the decision variables (Holland, 1975). Michalewicz (1996) described the representation of a chromosome in traditional genetic algorithm.

Suppose we need to maximize a function of n variables, $f(x_1, x_2, \dots, x_k)$: $R^k \to R$, and each variable $x_i$ takes values from a domain $D_i = [a_i, b_i] \subseteq R$. We also suppose six decimal places for the variables' values are desirable. Obviously, each domain should be cut into $(b_i - a_i) \cdot 10^6$ equal size range to achieve such precision. A representation having the variable coded as a binary string of length $n_i$ satisfies the precision requirement, if $n_i$ is the smallest integer which satisfies $(b_i - a_i) \cdot 10^6 \le 2^{n_i} - 1$ . Thus, totally we need $n = \sum_{i=1}^{k} n_i$ binary bits to represent each chromosome (potential solution).

But it is difficult for traditional Genetic Algorithm to maintain feasibility from parents to offspring when solving many optimization problems, especially discrete optimization problems like TSP. For example, Crossover (or recombination) is a genetic operator used to vary the programming of chromosomes from the two individuals of the fittest to form the next generation. Traditional Genetic Algorithm uses one point crossover (Holland, 1975). Suppose we have a traveling salesman problem that has 6 cities. A candidate for this problem is a permutation of these 6 cities. Two such permutations are 1→ 3 →2→4→5→6→1 and 6→ 4 →2→3→1→5→6. In traditional Genetic Algorithm, the genetic representation of these two sequences are the permutations $x = (1,3,2,4,5,6,1)$ and $x' = (6,4,2,3,1,5,6)$. A one point crossover will divide each permutation at the crossover point and exchange certain segments of the two permutations. Suppose the crossover point is the fourth place of the permutation. From Figure 1, we can see that the resulting sequences are 1→ 3 →2→4→1→5→6 and 6→ 4 →2→3→5→6→1. Both of them are infeasible.

In this research, we will use the Lehmer code to represent the candidate solution. But doing this, all solutions creating by crossover two parent solutions will be feasible. To do this, we first randomly generate a population of permutation of n (n is the number of the cities) $v_i$ .

where $i = 1,2,\ldots,n$ *(n is the number of solutions in the population)*

Then we use the inversion table to change this initial solution to Lehmer code representation.

2. Use 2-opt heuristic to improve these initial solutions until all solutions obtained are local optimal solutions. This 2-opt heuristic will be guided by both a weighted sum function. Then these initial solutions will be sorted according to their evaluation values which are obtained by evaluating these solutions by using a Tchebycheff function.
3. Repeat the following step until a stopping criterion is met.
    - Copying the best p% of solutions from the current generation to the next generation. To avoid the early convergence of the solutions, these solutions are different from each other. To do this, before a solution is selected, it needs to be compared with the solutions that have already been selected to see if there is a solution that is the same as this one. Only the solution that is different from the solutions that have already been selected will be selected and kept into next generation.
    - Crossover using the classical crossover to form c% of the solutions for the next generation.

      First, pick up two solutions from the current generation to serve as the parent. To make the algorithm converge to good solutions, we only select the 50% best solutions to crossover with each other. This means that only good solutions can be crossed over with each other. This can be done by picking up two different random numbers from 1 to 0.5*n (n is the population size).

In this research, the classic one point crossover will be use. And as we discussed above, the Lehmer code representation can keep edge information from the parents to the offspring. To keep enough edge information from the parents to the offspring while not hampering the exploration ability of the algorithm, we set the crossover point with 70% and 90% of the number of the cities.

For example, the first two random numbers are 1 and 3. This means that solution 1 that is $v_1 = (x_{11}\, x_{12} \cdots x_{1m})$ will exchange certain bits with solution 3 that is $v_3 = (x_{31}, x_{32}, \cdots, x_{3m})$. Then pick up another random number from 0.7*m to 0.9*m (m is the number of the cities), and this number will serve as the crossover point. For example, the number is 75. The solutions 1 and 3 exchange the bits after the 75th bit with each other. Thus the new solutions are $v_1' = (x_{11}\, x_{12}\, x_{13} \cdots x_{376} \dots x_{3m})$ and $v_3' = (x_{31}\, x_{32}\, x_{33} \cdots x_{176} \dots x_{1m})$.

- Using 2-opt heuristics and non 3-opt heuristics on the newly created solutions

The solutions created by crossover will be served as initial solution and put into 2-opt heuristic to find local optimal solutions. Again, the 2-opt heuristic will be guided by a weighted sum function. And 20% best solutions created by crossover will be further improved by using non sequential 3-opt heuristics. This non sequential 3-opt heuristics will also be directed by a weighted sum function.

- Mutation to form another p% solutions for the next generation

In this research, mutation will be worked on the p% best solutions of the current generation to form the p% solutions for the next generation. Thus solutions for the next generation will be completely generated (p%+c%+p%=1).

We noticed that if the Lehmer code of a city in a route is changed by 3, there will be 4 cities changed in this route. In this research, to implement mutation operator, we first select the p% best solutions from the population of the current generation. For each selected solution, pick up a random number j between 1 and n-5 (n is the number of the cities). This number indicates the position of the city that will be changed. Then pick up a random number in $[0,1]$. This number tells us the selected Lehmer code will be increased or decreased by 3. Finally, use non sequential 3-opt heuristic to find the local optimal solution for the new solution created by mutation. Again, here the non sequential 3-opt heuristics is guided by a weighted sum function.

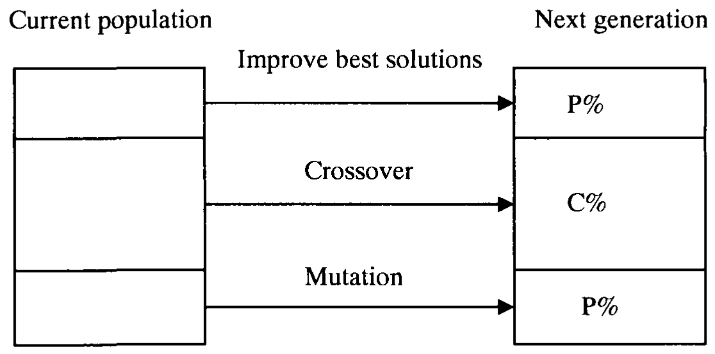


Figure 10. Generation Transition

- Evaluate the solutions using the weighted Tchebycheff function with ideal points.
- Sort the solutions according to the evaluation values getting from the above step.

4. Get the solution with best evaluation value.

This solution will be the optimal solution gotten by the new hybrid genetic algorithm.

The pseudo code for the hybrid genetic algorithm is in the following:

```
Begin /* hybrid genetic algorithm*
Generate initial population of solutions
Use Lehmer code encode the initial solutions
Find local optimal solutions by using 2-opt heuristic
Evaluate these local optimal solutions and sort them according to the fitness value.
While not finished Do
Copy certain part of solutions to form one part solutions for the next generation.
Breed new individuals through crossover and mutation
Find local optimal solutions for the newly generated solutions by using 2-opt heuristic and/or non sequential 3-opt heuristics
Evaluate these local optimal solutions and sort all the solutions for the next generation.
end
Get the best solution (this solution will be right on the first row of the solutions of last generation)
end
```

Figure 11. Pseudo code for the new Hybrid Lehmer code Genetic Algorithm

# CHAPTER 4

# IMPLEMENTATION ON SINGLE OBJECTIVE TSPS

## 4.1 Introduction

To test the performance of the new hybrid Lehmer code Genetic Algorithms, this Genetic Algorithms will be used on solving some single objective bench mark TSPs. These TSPs will be att48, kroA 100, kroB 100, kroC 100, kroD 100, kroA 150, kroB 150, kroA 200 and kroB 200. These TSPs were served as bench mark problems to test performance of many newly proposed heuristics and meta-heuristics on solving combinatorial problems. And their global optimal solutions are known. All these make them good bench mark problems in this research. They are available at TSPLIB (Reinelt, 1995).

For each bench mark problem, the algorithm will be run for 5 times to test the robustness of the algorithm so we can compare the results from this research with the results from Samanlioglu's (2006) research. The criteria used here will be the number of the optimal solutions obtained by using the new algorithm and the average relative excess.

## 4.2 Implementation

Since all these TSP problems are single objective, the solutions will be evaluated and sorted according to their objective values directly. And the fitness value will also guide the search of the local search algorithms, 2-opt and non sequential 3-opt.

In this research, the new genetic algorithm will use Lehmer code represent the candidate solutions. By using Lehmer code, the solutions getting from the crossover of the parent solutions will be always feasible.

Except for the genetic operators, crossover and mutation, that were used in traditional Genetic Algorithms, we add another operator that is called elite. To implement this operator, the 20 percent best solutions from last generation are selected to keep it to the next generation. And to avoid the early converge to the local optimal solutions, these 20 percent best solutions are non repeatable. This means that no solution can be selected twice in each generation.

In this research, non sequential 3-opt heuristic will be used in this research. As we know, local optimal solutions getting from 3-opt heuristic are also 2-opt optimal. The following figure show the pseudo code for the 3-opt heuristic used in this research.

```
3-opt

for k=1:m (m is the number of cities)
   for j=1:m-4
       for l=1:m-2-j
 prior_change=summation of distances between z(k)and z(k+1), z(k+j) and z(k+j+1), z(k+j+l) and
z(k+j+l+1);
 post_change=summation of distances between z(k) and z(k+j+l), z(k+1) and z(k+j+1), z(k+j) and
z(k+j+l+1)
   if post_change-prior_change<0
     make the change and update the fitness value
   end
       end
     end
end
```

Figure 12. Pseudo Code for 3-opt Heuristic

The non sequential 3-opt heuristic will work the following way:

1. Randomly pick up 3 points from the selected path.
2. Compute the distances between each point and its successive point and sum up these distances. The summation is defined as pre-distance.
3. Recombine these 3 points and their successive points and get the summation of the distances of the newly created lines. The summation is defined as post-

distance.

4. Define the change of distance as: the post-distance – the pre-distance. If the change of the distance is less than 0, change the sequence of the selected path as the new created path and the fitness value as the fitness value plus change of the distance.
5. Repeat the step 2-4, until no improvement can be made.

The following figure shows how the 3-opt works.

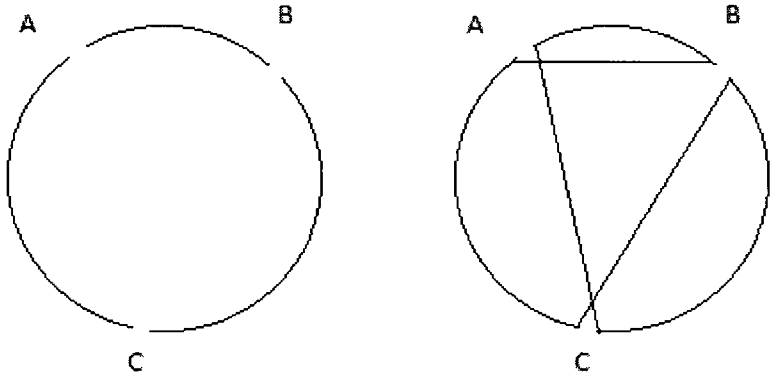


Figure 13. Non Sequential 3-opt Heuristics Move

By cutting a route into 3 parts, there will have 8 combinations for these 3 parts and only 4 combinations within these 8 combinations involve in exchanging 3 edges, AB'C', ACB', AC'B, A'BC' (A' is the reverse of A). To make it simpler, we only take into account one of these combinations AB'C'. And we called this non sequential 3-opt heuristic. In this research, solutions obtained by using non sequential 3-opt are also 2-opt optimal.

As stated above, the new hybrid Lehmer code Genetic Algorithms will use the Lehmer code to represent the permutation of the cities. By using Lehmer code representation, the offspring solutions created by crossover the parent solutions are always feasible. And the algorithm will also use traditional genetic operators: crossover and mutation. The crossover used here is the traditional one point crossover. First, randomly pick up two solutions. Then randomly pick up a number from 1 to n (n is the number of the cities). This random number will be served as the crossover point. Furthermore, switch certain parts of the two solutions according to the crossover point. At last, use 2-opt heuristic to find the local optimal solutions for the newly created offspring solutions.

For mutation, this research will use one point mutation. We noticed that if the Lehmer code of a city in a route is changed by 3, there will have 4 cities changed in this route. In this research, we first select the c% best solutions from the population of the current generation. For each selected solution, pick up a random number j between 1 and n-5 (n is the number of the cities). This number indicates the position of the city that will be changed. Then pick up a random number in[0,1], this number tells us the selected Lehmer code will be increased or decreased by 3. Finally, use non sequential 3-opt heuristic to find the local optimal solution for the new solution created by mutation. The following figure shows how one point mutation works.

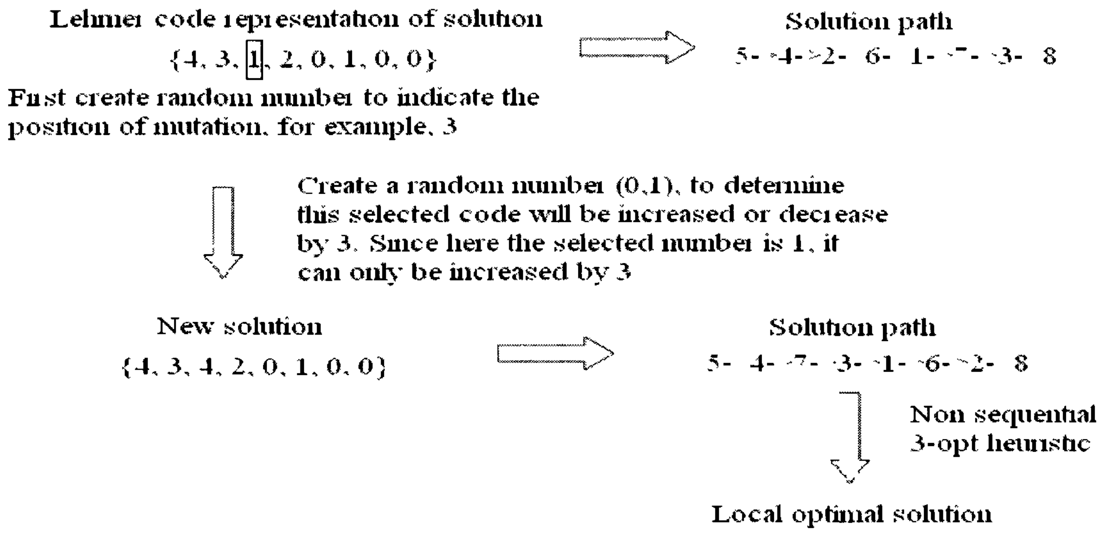

Lehmer code representation of solution
{4, 3, 1, 2, 0, 1, 0, 0}
First create random number to indicate the position of mutation, for example, 3
Solution path
5- 4- 2- 6- 1- 7- 3- 8
Create a random number (0,1), to determine this selected code will be increased or decrease by 3. Since here the selected number is 1, it can only be increased by 3
New solution
{4, 3, 4, 2, 0, 1, 0, 0}
Solution path
5- 4- 7- 3- 1- 6- 2- 8
Non sequential 3-opt heuristic
Local optimal solution

Figure 14. One Point Mutation Procedure

Whenever a solution is created, no matter if it is created by mutation or crossover of the parent solutions, 2-opt heuristic or non sequential 3-opt heuristic will work on it to find a local optimal solution. In this way, the solutions that comprise the population of each generation will be local optimal solutions. Thus, this new Genetic Algorithm will converge early. The 2-opt heuristic involves randomly picking up two edges in the current route and replacing them with another two edges that have same end points such that the resulting route has shorter distance. The following figure shows how 2-opt works.

```
2-opt
for k=1:m (m is the number of cities)
   for j=2:m-2
prior_change=summation of distances between z(k) and z(k+1),  z(k+j) and z(k+j+1)
post_change=summation of distances between z(k) and z(k+j),  z(k+1) and z(k+j+1)
   if post_change-prior_change<0
      make the change and update the fitness value
   end
      end
end
```

Figure 15. Pseudo Code for 2-opt Heuristics

In this chapter, the new algorithm will work on single objective TSP. So the evaluation of each solution will use its objective value directly. And solutions will be sorted according to their objective values. The 2-opt and 3-opt heuristics will be also guided by the objective values to search for local optimal solutions.

The following diagram (Figure 10) shows how the new hybrid Lehmer code Genetic Algorithms works on single objective TSP.

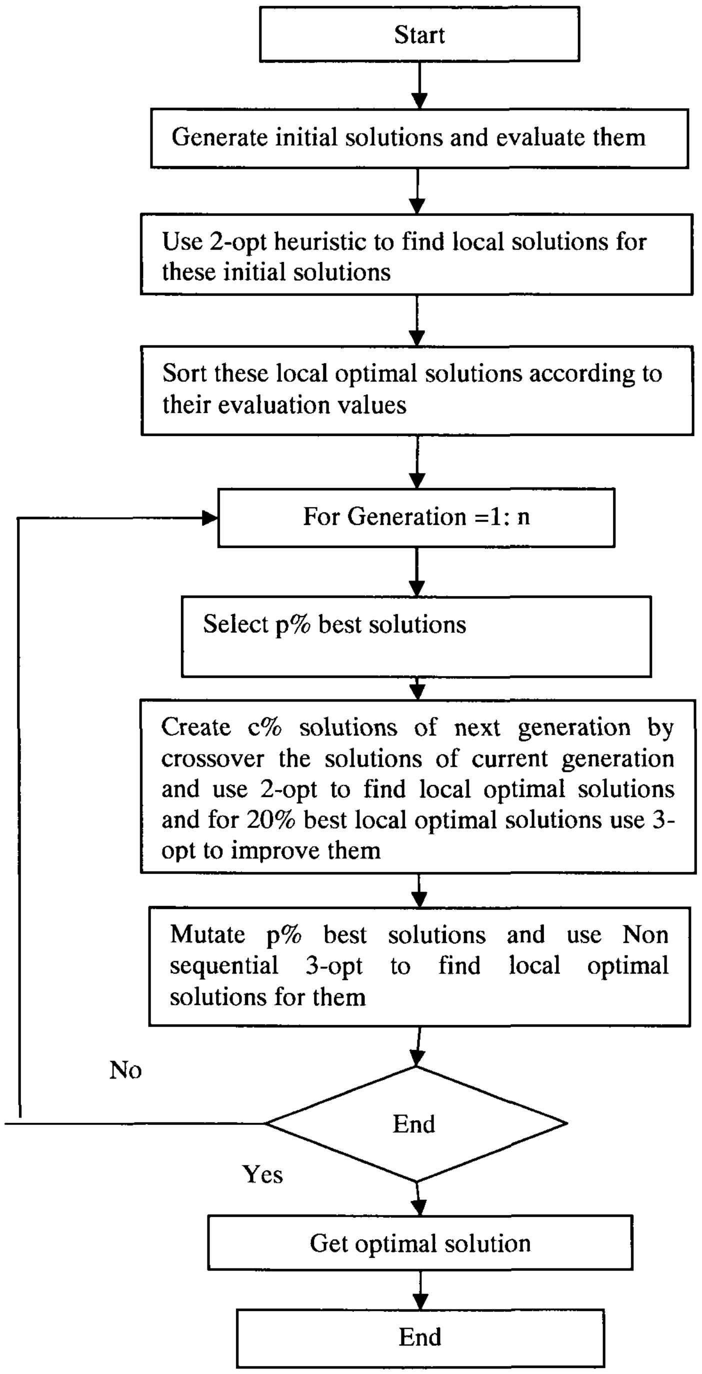

Start
Generate initial solutions and evaluate them
Use 2-opt heuristic to find local solutions for these initial solutions
Sort these local optimal solutions according to their evaluation values
For Generation =1: n
Select p% best solutions
Create c% solutions of next generation by crossover the solutions of current generation and use 2-opt to find local optimal solutions and for 20% best local optimal solutions use 3-opt to improve them
Mutate p% best solutions and use Non sequential 3-opt to find local optimal solutions for them
No
End
Yes
Get optimal solution
End

Figure 16. Process of the Hybrid Lehmer Code Genetic Algorithm

## 4.3 Experiment and Results

The new hybrid genetic algorithm will work on TSPs with cities from 48, 100,150 to 200. The following table shows the coordinates of the cities in Att48 that has 48 cities.

| | x | y | | x | y |
|---|---|---|---|---|---|
| 1 | 6734 | 1453 | 25 | 4307 | 2322 |
| 2 | 2233 | 10 | 26 | 675 | 1006 |
| 3 | 5530 | 1424 | 27 | 7555 | 4819 |
| 4 | 401 | 841 | 28 | 7541 | 3981 |
| 5 | 3082 | 1644 | 29 | 3177 | 756 |
| 6 | 7608 | 4458 | 30 | 7352 | 4506 |
| 7 | 7573 | 3716 | 31 | 7545 | 2801 |
| 8 | 7265 | 1268 | 32 | 3245 | 3305 |
| 9 | 6898 | 1885 | 33 | 6426 | 3173 |
| 10 | 1112 | 2049 | 34 | 4608 | 1198 |
| 11 | 5468 | 2606 | 35 | 23 | 2216 |
| 12 | 5989 | 2873 | 36 | 7248 | 3779 |
| 13 | 4706 | 2674 | 37 | 7762 | 4595 |
| 14 | 4612 | 2035 | 38 | 7392 | 2244 |
| 15 | 6347 | 2683 | 39 | 3484 | 2829 |
| 16 | 6107 | 669 | 40 | 6271 | 2135 |
| 17 | 7611 | 5184 | 41 | 4985 | 140 |
| 18 | 7462 | 3590 | 42 | 1916 | 1569 |
| 19 | 7732 | 4723 | 43 | 7280 | 4899 |
| 20 | 5900 | 3561 | 44 | 7509 | 3239 |
| 21 | 4483 | 3369 | 45 | 10 | 2676 |
| 22 | 6101 | 1110 | 46 | 6807 | 2993 |
| 23 | 5199 | 2182 | 47 | 5185 | 3258 |
| 24 | 1633 | 2809 | 48 | 3023 | 1942 |

Table 2. The Coordinates of Cities (ATT 48)

Att48 is a set of 48 cities (US state capitals) from TSPLIB. The distances between cities are Euclidean distance. So the goal of att48 is to minimize the distances of the route that visits each city once and return to the start point. The global optimal solution for this problem is known with the shortest distance: 10628. And the tour for the global optimal solution is: 1 → 8 → 38 → 31→ 44 → 18 → 7 → 28 → 6 → 37 → 19 → 27 →17→ 43 → 30 → 36 → 46 → 33 → 20 → 47 → 21 → 32 → 39 → 48 → 5 → 42 → 24 → 10 → 45 → 35 → 4 → 26 → 2 → 29 → 34 → 41 → 16 → 22 → 3 → 23 → 14 → 25 → 13 → 11 → 12 → 15 → 40 → 9 →1.

To implement the algorithm here, the initial parameters are set as the following: population size: 25 generations: 100, p% (percent of solutions being improved and mutated): 20%, c% (crossover rate): 60%. Each bench mark problem was run for 30 times. And the experiments were conducted computer with CPU, Intel Core 2 6600, 2.40 GHz, and 2 GB of RAM. And the algorithm was implemented in Matlab.

The relative excess over the best known solution is defined as

$$\text{relativeexcess} = \frac{\text{evalutaticn valuegettingfromour method - the best known evaluationvalue}}{\text{the best known evaluationvalue}}$$

The following table shows the results of the new genetic algorithm on Att48.

|  | Opt | Best Solution | Worst Solution | Average Relative Excess | Number of Opt |
|---|---|---|---|---|---|
| Att48 | 10628 | 10628 | 10628 | 0 | 30 |

Table 3. Results for Att48

From Table 2, we can see that the average relative excess is 0 and the number of optimal solution is 30. The new Genetic Algorithms seems robust when dealing with small single TSPs, like att48.

More experiments are executed on TSPs with the number of cities, 100, 150 and 200. These TSPs are the set of Krolak instances from TSP (Reinelt, 1995). They are KroA 100, KroB 100, KroC 100 and KroD 100 with 100 cities and KroA 150, KroB 2150, KroA 200 and KroB 200 with 200 cities. All these problems are generated from randomly placing cities in a rectangle measuring 4000 by 2000 and the using the rounded 2 dimensional Euclidean distance to generate the cost matrix (Hansen, 2000).

For all these TSPs, the same parameters will be used for the hybrid genetic algorithms. The new Genetic Algorithms worked on all these TSPs 5 runs. And for each run, the parameter was set to be the same, population size: 25, elite rate: 0.2, mutation rate: 0.2, crossover rate: 0.60 and generations: 100. The following table shows the results for TSPs with the 100 cities.

| | Opt | Best Solution | Worst Solution | Average Relative Excess | Number of Opt (out of 5) |
|---|---|---|---|---|---|
| KroA 100 | 21282 | 21282 | 21282 | 0.00% | 5 |
| KroB 100 | 22141 | 22141 | 22197 | 0.0524% | 4 |
| KroC 100 | 20749 | 20749 | 20749 | 0.00% | 5 |
| KroD 100 | 21294 | 21294 | 21294 | 0.00% | 5 |
| KroE 100 | 22068 | 22068 | 22106 | 0.0335% | 4 |

Table 4. Experiment Results for TSPs with 100 Cities

From the table, we can see that the new genetic algorithms proposed in this research can almost always find optimal solutions for all the selected TSPs with 100 cities with 4 to 5 out 5 runs. And the average relative excess for each TSP is very close to 0%, less than 0.1%. The average CUP time for this new Genetic Algorithms on solving single TSPs with 100 cities is around 398 second.

To make the new Genetic Algorithms more efficient, this means to make the new Genetic Algorithms running faster on single objective TSPs, we made a change to the algorithm. We only use 2-opt heuristics on the solutions created by mutation while not using non sequential 3-opt heuristics. We still got good results while the average CPU time was reduced to 152 seconds.

| | Opt | Best Solution | Worst Solution | Average Relative Excess | Number of Opt (out of 5) |
|---|---|---|---|---|---|
| KroA 100 | 21282 | 21282 | 21282 | 0.00% | 5 |
| KroB 100 | 22141 | 22141 | 22141 | 0.00% | 5 |
| KroC 100 | 20749 | 20749 | 20749 | 0.00% | 5 |
| KroD 100 | 21294 | 21294 | 21310 | 0.015% | 4 |
| KroE 100 | 22068 | 22068 | 22106 | 0.0335% | 4 |

Table 5. Another Experiment on TSPs with 100 Cities

Samanlioglu (2006) proposed a hybrid random key genetic algorithm to solve single TSPs. The population size used in her research was 300, 400, 500 and 1000 with generations 200 and 300, respectively. The following table shows the results from her research (Table 6).

| | N | X | Average Relative Excess | Average CPU Time (s) | Out of Five Replications |
|---|---|---|---|---|---|
| berlin52 | 300 | 200 | 0 | 2.4842 | 5o |
| | | 300 | 0 | 5.1092 | 5o |
| | 400 | 200 | 0 | 3.0814 | 5o |
| | | 300 | 0 | 4.2908 | 5o |
| | 500 | 200 | 0 | 6.7092 | 5o |
| | | 300 | 0 | 5.372 | 5o |
| | 1000 | 200 | 0 | 6.7722 | 5o |
| | | 300 | 0 | 7.75 | 5o |
| kroB100 | 300 | 200 | 0 | 15.4158 | 5o |
| | | 300 | 0.000867 | 16.9404 | 4o1t |
| | 400 | 200 | 0 | 21.588 | 5o |
| | | 300 | 0 | 37.053 | 5o |
| | 500 | 200 | 0 | 40.8218 | 5o |
| | | 300 | 0 | 25.5312 | 5o |
| | 1000 | 200 | 0.000714 | 62.9218 | 4o1t |
| | | 300 | 0 | 60.0842 | 5o |
| kroC100 | 300 | 200 | 0 | 14.225 | 5o |
| | | 300 | 0 | 13.2596 | 5o |
| | 400 | 200 | 0 | 18.05 | 5o |
| | | 300 | 0 | 14.3656 | 5o |
| | 500 | 200 | 0 | 21.053 | 5o |
| | | 300 | 0 | 18.7996 | 5o |
| | 1000 | 200 | 0 | 36.9718 | 5o |
| | | 300 | 0 | 35.3532 | 5o |
| kroD100 | 300 | 200 | 0.000282 | 21.472 | 3o2t |
| | | 300 | 0.000141 | 19.047 | 4o1t |
| | 400 | 200 | 0.000282 | 23.4812 | 3o2t |
| | | 300 | 0.000423 | 27.641 | 2o3t |
| | 500 | 200 | 0.000282 | 28.7062 | 3o2t |
| | | 300 | 0.000282 | 31.5592 | 3o2t |
| | 1000 | 200 | 0.000141 | 57.803 | 4o1t |
| | | 300 | 0.000141 | 56.5436 | 4o1t |
| kroE100 | 300 | 200 | 0.001133 | 24.5692 | 3o2t |
| | | 300 | 0.001251 | 21.147 | 2o3t |
| | 400 | 200 | 0.00029 | 21.2564 | 4o1t |
| | | 300 | 0.00048 | 22.8406 | 4o1t |
| | 500 | 200 | 0.001251 | 32.016 | 2o3t |
| | | 300 | 0.00048 | 24.8406 | 4o1t |
| | 1000 | 200 | 0 | 48.3846 | 5o |
| | | 300 | 0 | 45.1778 | 5o |

Table 6. Samanlioglu's Results for Single TSPs (2006)

Compared to the results from Samanlioglu's research, we can see that the results getting from the method proposed in this research are better than Samanlioglu's method even by using smaller population size and fewer generations. For example, for KroD 100, this research got 5 out of 5 optimal solutions with population size 25 and generation, 100 while Samanlioglu's research got 3 out of 5 optimal solutions with population size 300 and generation 300.

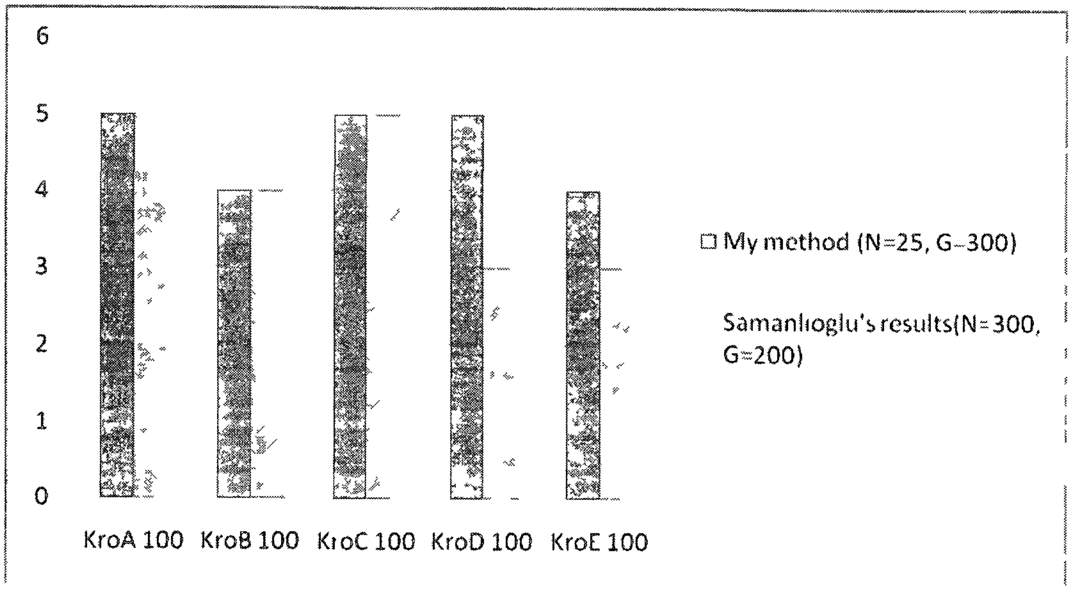


Figure 17. Comparison of Numbers of Optimal Solutions on TSPs with 100 Cities

Since in this research we programmed the code by using Matlab on the VCL (Virtual Computer Lab) computer with CPU, L5420, 2.50 GHz, and 2 GB of RAM while Samanlioglu's code was programmed by using C++ on a distributed computer system, we can't compare the CPU time for both methods.

To demonstrate the performance of the new hybrid Genetic Algorithms, more experiments were conducted on single objective TSPs with bigger sizes of cities. For these TSPs, the parameters are the exactly same as those used for the experiments with 100 cities, that is, population size: 25, generation: 100, c% (percent of solutions being improved and mutated): 20%, p% (crossover rate): 60%. And all these TSPs will be run for 5 times.

The following table shows the results for the new genetic algorithms on TSPs with 150 cities and 200 cites. We can see that the new genetic algorithms can still find the optimal solutions with probability. And the average relative excess for the TSPs with 150 and 200 cities is very low, less than 1% percent. This means that the performance of the proposed new hybrid Lehmer code Genetic Algorithms on medium size TSPs, such as TSPs with 150 cities and 200 cities, is also good. And it may need to use bigger population size or run longer generation to obtain better performance for the proposed method on bigger size TSPs.

| | Opt | Best Solution | Worst Solution | Average Relative Excess | Number of Opt (out of 5) |
|---|---|---|---|---|---|
| KroA 150 | 26524 | 26524 | 26598 | 0.1108% | 4 |
| KroB 150 | 26130 | 26130 | 26140 | 0.0115% | 4 |
| KroA 200 | 29368 | 26431 | 26590 | 0.4358% | 1 |
| KroB 200 | 29437 | 29455 | 29695 | 0.0591% | 1 |

Table 7. Results for Single TSPs with 150 and 200 Cities

From the above table, we can see that the algorithm also work well when dealing with bigger single TSPs with the same configuration. The average relative excess is less than 1%. To get better solutions on TSPs with more cities, the population size and the generation should be increased.

## 4.4 Conclusion

From the above experiments, we can see that the new hybrid Genetic Algorithm can easily find the optimal solution for small size TSPs, for example, att48, kroA 100, kroB 100, kroC 100 and kroD 100, and also the optimal solutions for medium size TSP, such as, kroA 150, kroB 150, KroA 200 and KroB 200.

Compared with the other Genetic Algorithms and hybrid Genetic Algorithms, the new hybrid Lehmer code Genetic Algorithms has some advantages on single TSP.

First, by using Lehmer code, the new hybrid Genetic Algorithms can easily overcome the difficulty with traditional Genetic Algorithms in keeping feasibility when crossover parent solutions create offspring. As we can see in the experiments, the crossover operator directly work on the Lehmer code represented parent solutions and no repairing process needed for the offspring solutions. All the solutions from crossover are feasible.

Furthermore, although the random keys Genetic Algorithms can also solve this difficulty, it loses the edge information at the same time when it solves the difficulty of keeping feasible solutions. This means, the random keys Genetic Algorithms can keep any schematic information from the parent solutions to the offspring. As we know, this schematic information is very important feature of Genetic Algorithms. And it's one of the reasons that makes Genetic Algorithms converge quicker. While by using Lehmer code representation, at least certain part of the edge information can be transferred from the parent solutions to the offspring. For example, the following figure shows how Lehmer code representation one point crossover keeps part of parent information into the offspring solutions.

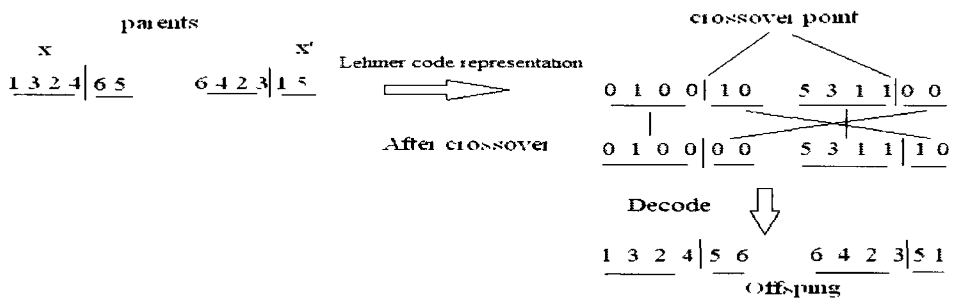


Figure 18. One Point Crossover of Lehmer Code GA (Keep Parent Information)

From the figure, we can see that the parent edge information, $1 \to 3 \to 2 \to 4$ and $6 \to 4 \to 2 \to 3$ were kept from the parent solutions to the offspring respectively.

Finally, the implementation of the new Genetic Algorithms is very easy compared to the random keys Genetic Algorithms. For example, for the encoding and evaluation of the solutions, the random keys Genetic Algorithms must sort all cities and get the ranks of all cities first. This means that at least an extra sort process is needed. And you don't know the neighbors for a specific city before you sort all cities. While in Lehmer code Genetic Algorithms, the position of each code is the rank of the city represented by the code. And the city represented by the code can be easily gotten from the list {1, 2, 3, ... , n} (n is the total number of the cities) according to the code of the city. The neighbors of this city are the cities right before and behind it.

Another advantage of this new Genetic Algorithm is in tuning up the solutions. For example, in this research, I used an improvement operator in this new hybrid Genetic Algorithms. The idea is to improve the solution by randomly changing the locations of 4 cities. This operator can be very simply implemented by randomly picking up a random number and decreasing or increasing the code which is located in the position of the random number by 3. The following figure shows how this works.

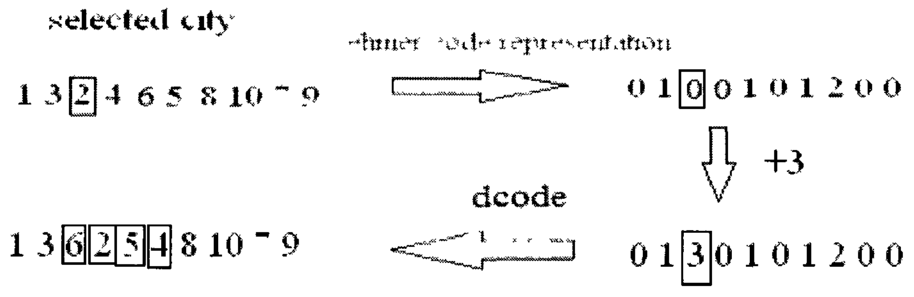


Figure 19. One Point Mutation

From this figure, we can see that three cities change position by simple add 3 for one of the cities' Lehmer code. While in random keys Genetic Algorithms, it is not so easy to implement the similar operator. For example, in Samanlioglu's Random Keys

Genetic Algorithms, 4 cities were randomly picked up and all the combinations of these cities were taken into account. Then 2-opt heuristics worked on these possible solutions and selected the best one. At last, repeated the above procedure for 10 times and selected the one that had the best fitness value.

Based on the above experiments and discussion, we can reach the conclusion that the new hybrid Lehmer code Genetic Algorithms is very robust when dealing with the single small and medium size of single objective TSPs. And it's very easy to implement this new hybrid Genetic Algorithms.

The future work will be to find ways to improve the performance of the algorithms on medium and large size of single objective TSPs. As we can see the algorithm performed not good even for medium size of single TSPs as it on small size problems. Another direction is to improve the efficiency of the algorithms. Since we used non sequential 3-opt heuristic as one of the local search heuristics and as we know, 3-opt heuristic is time consuming compared to 2-opt heuristic, the time expense for this new hybrid genetic algorithm is higher than Samanlioglu's random keys genetic algorithm which used only 2-opt heuristic as the local search method. The main reason is that the algorithm in this research was coded by using Matlab on the VCL computer with CPU, L5420, 2.50 GHz, and 2 GB of RAM while Samanlioglu's code was programmed by using C++ on a distributed computer system. So it leaves some room for us to improve the efficiency of the algorithm in the future.

# CHAPTER 5

# IMPLEMENTATION ON MULTI-OBJECITVE TSPS

The single objective TSPs belong to the class of NP-hard problems. It's hard to find a global optimal solution efficiently and effectively. While for multi-objective TSPs, the difficulty of the TPS itself and the difficulty of multiple objectives make it a NP-complete problem. It's much harder to find the global optimal solutions for multi-objective TSPs. Many research used the methods that were used to solve the single objective TSPs to solve multi-objective TSPs. But the performance was not as good as solving the single objective TSPs. And the solutions for multi-objective TSPs are highly depended on the preferences of the decision makers and the compromises between different objectives.

Many heuristics and meta-heuristics were proposed to attack the multi-objective TSPs. Within all these heuristics and meta-heuristics, the evolutionary meta-heuristics, such as Genetic Algorithms and all its variants, etc., are very promising. These meta-heuristics start from initial solutions and gradually improved to better solutions. Ideally, an acceptable solution can be found within a certain timeline. And these techniques deal with a population of solutions simultaneously. This makes them a good option to deal with multiple objectives since a compromise between different objectives can be made within a population of solutions.

The evolutionary multi-objective optimization methods can be classified into two types: the Pareto-based technique and non Pareto-based technique (Samanlioglu, Ferrell & Kurz, 2008). For the Pareto-based techniques, the selection is directed by the Pareto dominance and Pareto ranking. The multi-objective genetic algorithm proposed by

Fonseca and Fleming (1993), the niched Pareto genetic algorithm (Horn, Nafpliotis & Goldberg, 1994), NSGA II (Deb et al., 2002), SPEA II (Zitzler, Laumanns & Thiele, 2001), etc., belong to this type of technique.

In the non Pareto-base techniques, the selection does not directly rely on the Pareto dominance and Pareto ranking, like vector evaluated genetic algorithm (Schaffer, 1984), target vector approach (Coello, 2001), memetic random key genetic algorithm (Samanlioglu, Ferrell & Kurz, 2008), etc.

In this research, a new hybrid Lehmer code Genetic Algorithms will be proposed to solve the multi-objective TSPs. A non Pareto-based technique will be used for this new algorithm in solving multi-objective TSPs. More specifically, Target Vector Approach will be used in this research. In this approach, the goal is to minimize the distance between the generated solution and the target vector. This goal guides the new Genetic Algorithms to find solutions for multi-objective TSPs. Here we will use the weighted Tchebycheff function with the ideal points as the reference points. And the built-in local search technique, 2-opt heuristic and the non-sequence 3-opt will be guided by a weighted sum function.

## 5.1 Introduction

The main difference between single objective TSPs and multi-object TSPs is that for single the evaluation of the solution is solely dependent on the single fitness value of the solution while in multi-objective TSPs, the evaluation will depend on multiple fitness values. Therefore, the difficulties to solve multi-objective TSPs come from both the difficulty of TSPs itself and the difficulty of multiple objectives. As we know, even single TSP is a NP-hard problem. The multiple objectives make multi-objective TSPs NP-complete problems.

The multi-objective optimization problem (MOP) can be defined as the following:

$$\begin{aligned} \min \quad & f(x) = \{f_1(x), f_2(x), \ldots, f_k(x)\} \\ s.t. \quad & x \in S \end{aligned} \tag{20}$$

Where $k \geq 2$, $x = (x_1, x_2, \ldots, x_n)$ is the decision variable vector, $S$ is the feasible solution space, $f(x)$ is the objective vector.

Weighted $L_P$ norms are defined as the following (Hansen and Jaszkiewicz, 1998):

$$L_p(z^1, z^2, \Lambda) = (\sum_{j=1}^{J} \lambda_j \left| z_j^1 - z_j^2 \right|^p)^{1/p} \quad p \in \{1, 2, \ldots\} \tag{21}$$

Where $\Lambda = [\lambda_1, \lambda_2, \ldots, \lambda_J], \lambda_j \geq 0$, is a weight vector.

For a multi-objective TSP, the general weighted $L_P$ norm is defined as

$$\begin{aligned} \min \quad & (\sum_{j=1}^{J} \lambda_j \left| f_j - z^* \right|^p)^{1/p} \\ s.t. \quad & x \in S \end{aligned} \tag{22}$$

Where $\lambda_j \geq 0, j = 1, 2, \ldots, J, \sum_{j=1}^{J} \lambda_j = 1$, $z^*$ is the reference point.

If we set the reference point the global optimal solution for $f_j$, when $p = \infty$, we get the weighted Tchebycheff metric

$$\begin{aligned} \min \quad & \max\{\lambda_j (f_j - z^*)\} \\ s.t. \quad & x \in S \end{aligned} \tag{23}$$

When $p = 1$, we get the weighted sum function

$$\begin{aligned} \min \quad & \sum_{j=1}^{J} (\lambda_j (f_j - z^*)) \\ s.t. \quad & x \in S \end{aligned} \tag{24}$$

Many heuristics and meta-heuristics were proposed to solve single and multiple objectives TSPs, including Genetic Algorithms (Holland, 1975), Simulated Annealing (Kirkpatrick, Gelatt & Vecchi, 1983), Tabu Search (Glover, 1989), Ant Colony (Dorigo, 1992) and Particle Swarm (Kennedy & Eberhart, 1995), etc. Within all these techniques, genetic algorithm is very promising in solving TSPs including single and multiple objectives TSPs because of its ability to deal with a population of solutions simultaneously and evolve to better solutions, especially when it is combined with domain-specific local search heuristics, such as 2-opt, 3-opt, n-opt, etc. With its genetic operators, such as mutation and crossover, genetic algorithms can have good exploration and exploitation abilities. The mutation operator can keep diversity for genetic algorithm while crossover together with the elite selection procedure can let genetic algorithms converge to better solutions.

There are also many heuristics and meta-heuristic proposed to attack on multi-objective combinatorial optimization problems. Among these, Jaszkiewicz and Czyzak (1998) proposed a Pareto simulated annealing algorithm to solve multi-objective combinatorial optimization problems. Jaszkiewicz (2002) presented a new genetic local search algorithm to solve multi-objective TSP. And he concluded that local search guided by weighted linear function gave better solution than guided by weighted Tchebycheff function. Hansen (2000) proposed a Tabu Search with local search heuristic to solve multi-objective TSP. He suggested that heuristic of the Tchebycheff function gives better solution when using a substitute scalarizing function instead of the Tchebycheff function to guide the local search heuristic. Samanlioglu, Ferrell and Kurz (2008) present a memetic random key genetic algorithm embedded with a 2-opt heuristic to solve multi-objective TSP. In her research, the local search is guided randomly by either a weighted Tchebycheff function or a weighted sum function.

In this research, a new hybrid Lehmer code Genetic Algorithm which combined Genetic Algorithm with local search techniques, 2-opt and non sequential 3-opt. We used Lehmer code to represent the potential solutions for Genetic Algorithms. With the exploration ability of Genetic algorithm and the exploitation ability of local search, this new genetic algorithm is good in exploration and exploitation. And by using Lehmer code, the solutions created by mutation and crossover operators are always feasible. So an extra repairing step is not required in this new algorithm.

And to attack the multiple objectives difficulty associated with multi-objective TSPs, we used non Pareto-based technique. To be more specific, the Target Vector Approach will be used in this research. The goal of this technique is to minimize the distance between the generated solution and the target vector. This goal guides the new Genetic Algorithms to find good solutions for multi-objective TSPs.

In this research, 2-opt heuristics and non-sequence 3-opt heuristics were directed by a weighted sum function to find local optimal solutions.

## 5.2 Implementation

The implementation of the new hybrid Lehmer code on multi-objective TSPs is similar to the implementation of this algorithm on single objective TSPs. But since multi-objective TSPs is more complicated than the single objective TSPs, it's much more difficult to solve multi-objective TSPs than single objective TSPs. Additional procedures are needed to solve multi-objective TSPs.

The following diagram (Figure 10) shows how the new hybrid Lehmer code

Genetic Algorithms works on multi-objective TSP.

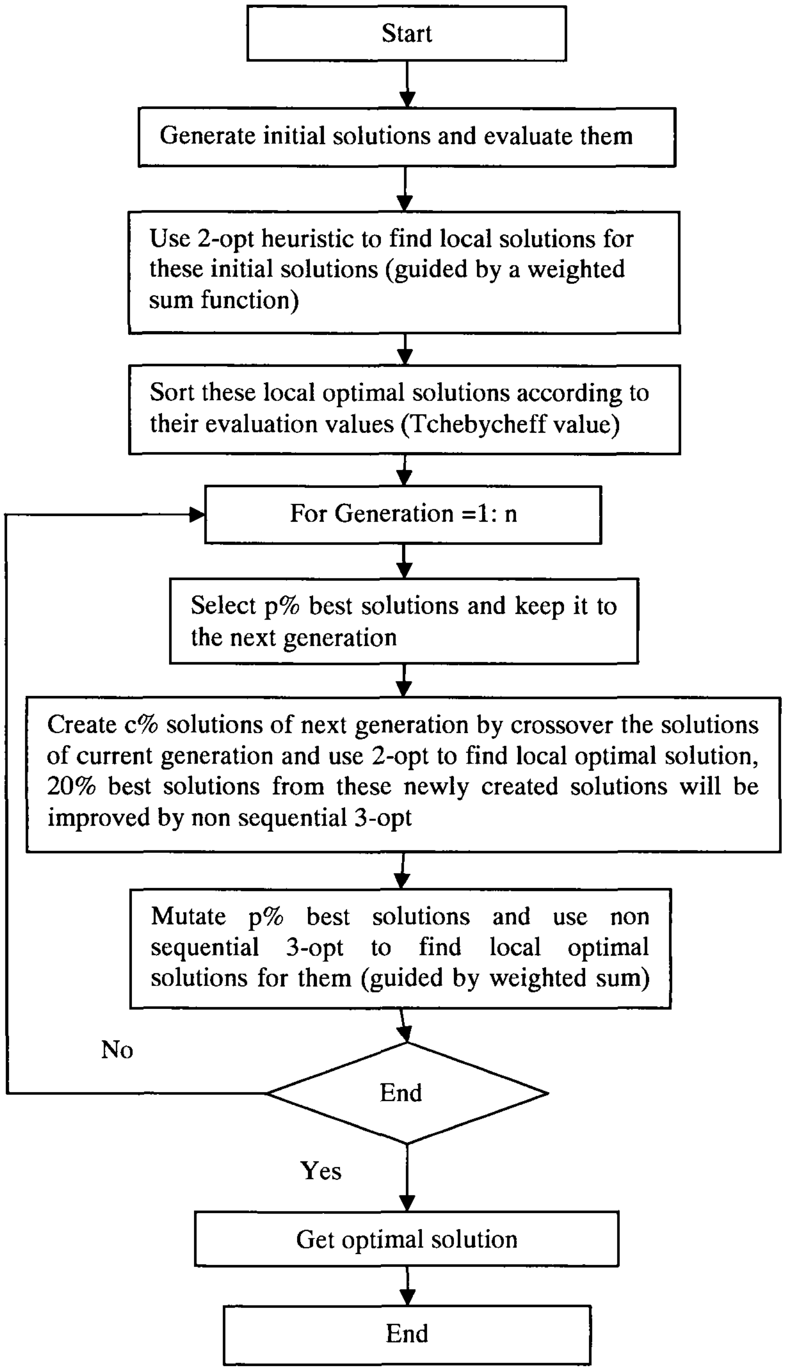

Start
Generate initial solutions and evaluate them
Use 2-opt heuristic to find local solutions for these initial solutions (guided by a weighted sum function)
Sort these local optimal solutions according to their evaluation values (Tchebycheff value)
For Generation =1: n
Select p% best solutions and keep it to the next generation
Create c% solutions of next generation by crossover the solutions of current generation and use 2-opt to find local optimal solution, 20% best solutions from these newly created solutions will be improved by non sequential 3-opt
Mutate p% best solutions and use non sequential 3-opt to find local optimal solutions for them (guided by weighted sum)
No
End
Yes
Get optimal solution
End

Figure 20. Multi-objective TSPs Process of the Hybrid Lehmer Code Genetic Algorithm

The procedure for the new genetic algorithm:

1. Initial solutions

To create initial solutions, we randomly created a population of permutations of n cities (n is the total number of the cities). And in this research, 2-opt heuristic was used to improve these initial solutions until local optimal solutions was obtained.

Unlike the 2-opt heuristic used in single objective TSPs, here the 2-opt heuristic was guided by weighted sum function. The 2-opt heuristic in single objective TSPs is solely guided by the single objective value. This means that change in the single objective value will determine if the change in the solution will be accepted. While in multi-objective TSPs, because of the existence of multiple objectives, whether a change will be accepted or rejected can't be determined by only one objective value. In this research, we used a weighted sum function to guide the 2-opt heuristics. This means that a change will be accepted if it can lead to a lower value of the weighted sum function.

After the population of initial solutions was created, they were sorted according to their evaluation values obtained by using the Tchebycheff function with an ideal point.

2. Representation

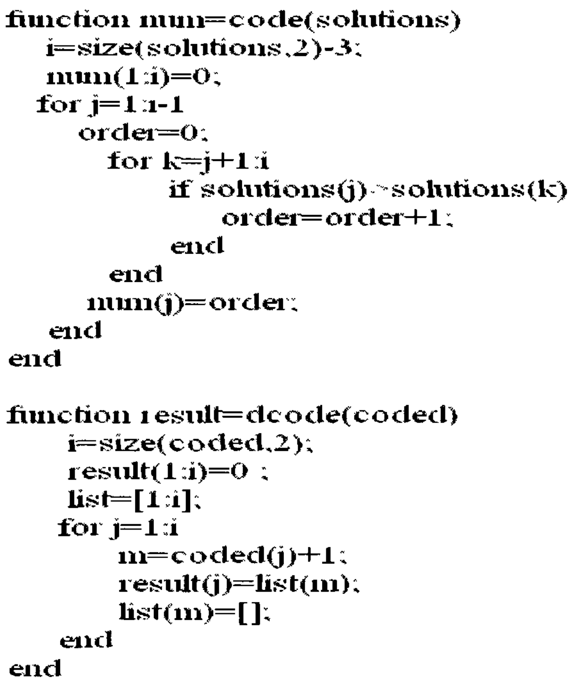
To prepare the initial solutions for the processing of genetic operators, these solutions should be represented by a certain representation method. As discussed above, it is difficult for traditional Genetic Algorithm to maintain feasibility from parents to offspring when solving many optimization problems, especially discrete optimization problems like TSP. The problem is with the representation of traditional Genetic Algorithms. When solving TSPs, traditional Genetic Algorithms used the permutation of the cities to represent the potential solutions. This will cause infeasible solutions problem when crossing over the current solutions to created offspring solutions.

For example, we have a traveling salesman problem that has 6 cities. A candidate for this problem is a permutation of these 6 cities. Two such permutations are 1→ 3 →2→4→5→6→1 and 6→ 4 →2→3→1→5→6. These two permutations served as two parent solutions. When we use one point crossover and set the third place as the crossover point, the resulting sequences are 1→3→2→3→1→5→6 and 6→4 →2→4→5→6→1. Neither of them is feasible. To solve this problem, the repairing procedure is need for traditional Genetic Algorithms.

While by using the Lehmer code representation, these two parent solutions are represented as (0, 1, 0, 0, 0, 0) and (5, 3, 1, 1, 0, 0). When we use the one point crossover and the same crossover point as in the traditional Genetic Algorithms, the resulting solutions are (0, 1, 0, 1, 0, 0) and (5, 3, 1, 0, 0, 0). These two solutions represent the sequence 1→3→2→5→4→6→1 and 6→4→2→1→3→5→1. Both solutions are feasible solutions. Crossover solutions represented by using Lehmer code always create feasible solutions and no repairing procedure is needed here.

And Lehmer code representation has an advantage over random key representation. One of the important features for Genetic Algorithms is that Genetic Algorithms can keep some parent information from the parents to the offspring. Although random key representation can overcome the infeasible solutions difficulty, it can't keep any edge information from the parent solutions to the offspring solutions. While in Lehmer code representation, the edge information before the crossover point can be preserved from the parent solution to the offspring solutions. For example, in the above example, the first three cities in the offspring solutions are exactly the same as those in the parent solutions while in random keys Genetic Algorithms, no any edge information can be preserved from the parents to the offspring.

Lehmer code of a permutation can be expressed by using inversion table. Consider a sequence of n numbers $x = (x_1 x_2 \ldots x_n)$. An inversion is a pair $(x_i, x_j)$ such that $i < j$ and $x_i > x_j$. For $i \in \{1,\ldots,n\}$, let $d_i$ count the number of inversions with i as the smaller index. Then the sequence $(d_1 d_2 \ldots d_n)$ is called inversion table of permutation x. $0 \le d_i \le n - i$ for $i = 1,\ldots,n$.


```
function num=code(solutions)
   i=size(solutions,2)-3;
   num(1:i)=0;
  for j=1:i-1
     order=0;
       for k=j+1:i
           if solutions(j)>solutions(k)
               order=order+1;
           end
       end
      num(j)=order;
  end
end

function result=dcode(coded)
     i=size(coded,2);
     result(1:i)=0 ;
     list=[1:i];
    for j=1:i
        m=coded(j)+1;
        result(j)=list(m);
        list(m)=[];
    end
end
```

Figure 21. Matlab code for Lehmer Presentation and Decoding Process

3. Repeat the following step until a stopping criterion is met.

- Copying the best p% of solutions from the current generation to the next generation and keep it to the next generation. And as discussed above in single objective TSPs, these p% best solutions are different with each other.

- Crossover using the classical one point crossover to form c% of the solutions for the next generation.

Randomly pick up two solutions from the current generation to serve as the parent solutions. This can be done by picking up 2 different random numbers for 1 to 0.5*k (k is the population size). This is the same as in the single objective TSPs, only 50 percent best solutions can crossover with each other. In this research, the classic one point crossover will be use. And since all solutions are local optimal solutions, to keep more edge information while not hampering the exploration ability, the crossover point is selected between 0.7m to 0.9m ( m is the number of cities).

For example, the first two random numbers are 1 to 3. This means that solution 1 that is $\vec{v}_1 = (\vec{x}_{11}\, x_{12} \cdots x_{1m})$ will exchange certain bits with solution 3 that is $\vec{v}_3 = (\vec{x}_{31}, x_{32}, \cdots, x_{3m})$.Then pick up another random number from 0.7*m to 0.9*m (m is the number of the cities). This number will serve as the crossover point. For example, the number is 75. Then the solutions 1 and 3 exchange the bits after the 75th cities. The new solution created will be $\vec{v}_1' = (\vec{x}_{11}\, x_{12}\, x_{13} \cdots x_{376} \cdots x_{3m})$ and $\vec{v}_3' = (\vec{x}_{31}\, x_{32}\, x_{33} \cdots x_{176} \cdots x_{1m})$. Use 2-opt heuristics on the newly created solutions to get local optimal solutions. Then these two newly created solutions will be compared with each other. The one with the better fitness value (evaluated by Tchebycheff function with an ideal point) will be kept in next generation. At last, non sequential 3-opt heuristics will work on 20% best solutions created by 2-opt heuristics. Again, the 2-opt heuristics and non sequential 3-opt heuristics will be guided by the weighted sum function.

- Mutation to form another c% solutions for the next generation

In this research, mutation will be worked on the p% best solutions of the current generation to form the p% solutions for the next generation. The solutions created by mutation will be put into non sequential 3-opt heuristic to find local optimal solutions. Again the non-sequential 3-opt heuristic was guided by weighted sum function.

Thus solutions for the next generation will be completely generated (p%+c%+p%=1).

To perform mutation, for each solution of the p% best solutions in the current generation, first we generate a random number j between 1 and m-5 (m is the number of cities). This number tells us the position of the number that will be mutated. Then pick up a random number in $[0,1]$, this number tells us the selected Lehmer code will be increased or decreased by 3.

After new solutions were created by mutation, non-sequential 3-opt heuristic was used to improve them until a local optimal solution was obtained for each solution. Again this non sequential 3-opt heuristic was guided by a weighted sum function.

- Sort the solutions according to the evaluation values from the above step.

Newly created Solutions were sorted according to their evaluation values from the weighted Tchebycheff function no matter if they were created by 2-opt heuristics and non sequential 3-opt heuristics that are guided by the weighted sum function or they were created by just copying p% best solutions of last generation.

4. Get the solution with the best evaluation value.

This best solution will be the optimal solution gotten by the new hybrid genetic algorithm. It can be obtained by just simply picking up the first solution in the last generation.

## 5.3 Experiment and Results

The new hybrid Lehmer code Genetic Algorithms will work on TSPs with 2 to 5 objectives TSP with the size of cities 100. To make an effective comparison, the proposed algorithm will be used to solve the same multi-objective TSPs as those used in (Hansen, 2000) and (Samanlioglu, Ferrell & Kurz, 2008). More specifically, I will use the set of Krolak instances with 100 cities from TSPLIB (Reinelt, 1995). The Krolak instances include 5 instances, kroA 100, kroB 100, ..., kroE 100. For the multi-objective TSP problems of this research, each instance correspond to the cost matrix of one objective function, for example, for a three objective TSP problem, kroA 100 corresponds to the cost matrix of objective function 1, kroB 100 corresponds to that of objective function 2, kroC 100 to that of objective function 3.

There are two main advantages in doing this. First, the tours will be within the same scale of range for the different objectives. This makes the range scaling unnecessary. Furthermore, the optimal value to each problem is known. So we get the exact ideal point for each objective function. This makes the implementation of the weighted Tchebycheff function very easy.

Moreover, to make the comparison between the method of this research and the methods proposed by Hansen (2002) and Samanlioglu et al. (2008) valid, the method proposed by this research will work on TSP problems that have two objectives to up to five objectives, and each problem will be run 30 times.

For all these multi-objective TSPs, the same parameters will be used for the hybrid genetic algorithms. First, the new genetic algorithms will work on all these TSPs

30 runs. And for each run, the parameter will be the same, population size: 25, improve rate: 0.2, mutation rate: 0.2, crossover rate: 0.60 and generations: 125.

And the experiments were conducted on the computer with CPU, Intel Core 2 6600, 2.40 GHz, and 2 GB of RAM. And the algorithm was implemented in Matlab.

The following table shows the best solution gotten from Hansen's method and the best solution obtained by using the proposed method on 3 objects TSP with 100 cities. From the table, we can see that the proposed method can find better solution than Hansen's method.

| | Hansen | Proposed Method |
|---|---|---|
| | 1-53-26-49-10-97-99-5921-72-38-88-22-94-70-56-75-80-65-79-47-67-40-85-68-30-100-81-69-73-3-29-46-12-27-35-86-62-20-55- 83-43-71-39-5-95-76-91-28-8-90-98-25-34-58-54-50-2-64-13-41-14-33-82-78-48-96-44-51-63-16-24-18-19-92-45-36-74-84-66-4-89-31-42-6-15-17-23-77-60-57-87-52-37-7-9-61-32-11-93 | 1-53-26-49-10-97-99-59-21-70-94-22-88-72-38-56-75-80-65-79-47-67-40-85-29-46-12-27-35-86-62-20-55-83-43-71-34-25-58-98-90-8-28-91-76-95-5-39-68-30-3-73-69-81-100-37-52-96-44-78-82-33-13-41-14-48-51-63-16-24-18-19-92-45-36-74-84-66-4-89-54-50-2-64-42-31-6-15-17-23-77-60-57-87-7-9-61-32-11-93 |
| Cost in KroA | 67274 | 67202 |
| Cost in KroB | 68054 | 68079 |
| Cost in KroC | 66751 | 66553 |
| W, $w_i$=1/k | (1/3,1/3,1/3) | (1/3,1/3,1/3) |
| Solution | 15334 | 15313 |

Table 8. Solutions Obtained by Hansen's Method and the Proposed Method

The following tables (Table 9, Table 10) shows the results of the new Genetic

Algorithms on multi-objective TSPs with N=25, G=125 and N=50, G=62, respectively (N is the population size, G is the generations).

| | Opt | Best Solution | Worst Solution | Average Relative Excess | Number of Opt (out of 30) |
|---|---|---|---|---|---|
| KroAB100 | 14256 | 14256 | 14417 | 0.5346% | 4 |
| KroABC100 | 15313 | 15313 | 22197 | 0.7715% | 1 |
| KroABCD 100 | 14241 | 14295 | 14431 | 0.8357% | 0 |
| KroABCE 100 | 14292 | 14292 | 14564 | 0.9328% | 1 |
| KroACDE 100 | 14088 | 14121 | 14310 | 0.8327% | 0 |
| KroABCDE100 | 12888 | 12936 | 13111 | 0.9846% | 0 |

Table 9. Results on Multi-objective TSPs (N=25, G=125)

| | Opt | Best Solution | Worst Solution | Average Relative Excess | Number of Opt (out of 30) |
|---|---|---|---|---|---|
| KroAB100 | 14256 | 14256 | 14417 | 0.5752% | 2 |
| KroABC100 | 15313 | 15313 | 22197 | 0.7580% | 1 |
| KroABCD 100 | 14241 | 14295 | 14431 | 0.7651% | 0 |
| KroABCE 100 | 14292 | 14292 | 14564 | 0.9118% | 1 |
| KroACDE 100 | 14088 | 14121 | 14310 | 0.7427% | 0 |
| KroABCDE100 | 12888 | 12936 | 13111 | 0.9539% | 0 |

Table 10. Results on Multi-objective TSPs (N=50, G=62)

The following table shows the comparison of results of this research and Samanlioglu on multi-objective TSPs with 2, 3, 4 and 5 objectives with population size 25, generation 125 and population size:50, generation 62, respectively.

| | N=25 G=125 | | | | N=50 G=62 | | | |
|---|---|---|---|---|---|---|---|---|
| | Samanlioglu | | This research | | Samanlioglu | | This research | |
| | ARE | R | ARE | R | ARE | R | ARE | R |
| KroAB100 | 0.452 | 3 | 0.5346 | 4 | 0.4003 | 2 | 0.5752 | 2 |
| KroABC100 | 0.7138 | 1 | 0.7715 | 1 | 0.723 | 0 | 0.758 | 1 |
| KroABCD100 | 0.6873 | 0 | 0.8357 | 0 | 0.5597 | 0 | 0.7651 | 0 |
| KroABCE100 | 0.785 | 1 | 0.9328 | 1 | 0.646 | 0 | 0.9118 | 1 |
| KroACDE100 | 0.7394 | 1 | 0.8327 | 0 | 0.5829 | 1 | 0.7427 | 0 |
| KroABCDE100 | 0.9469 | 1 | 0.9846 | 0 | 1.0027 | 0 | 0.9539 | 0 |

Table 11. Comparison of Results from This Research and Samanlioglu's Research

The relative excess over the best known solution is defined as

$$\text{relativeexcess} = \frac{\text{evalutation valuegettingfromour method- the best knownevaluationvalue}}{\text{the best knownevaluationvalue}}$$

From this table, we can see that the proposed Genetic Algorithm works well on multi-objective TSPs with objectives from 2 to up to 5. For all these multi-objective TSPs, the average relative excess is less than 1%. And the results getting from this research can find comparable results with Samanlioglu's research. For example, for KroAB 100, the proposed method found 4 out 30 optimal solutions while Samanlioglu's method 3. And for kroABCDE100, the average relative excess for this research is 0.9539% while Samanlioglu's method is 1.0027%.

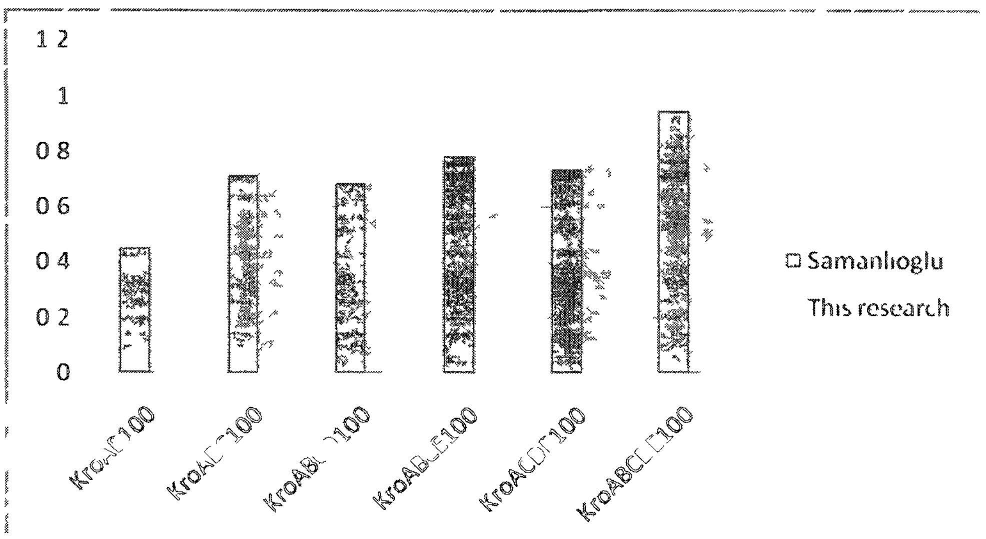


Figure 22 Comparison of ARE (N=25, G=125)

The above figure shows the comparison of the average relative excess of these two methods with population size 25, generation 125 We can see that the relative excesses are very close for these two methods

The figure below shows the comparison of the average relative excess of these two methods with population size 50, generation 62 We got similar results with the above. The results for both methods are very close.

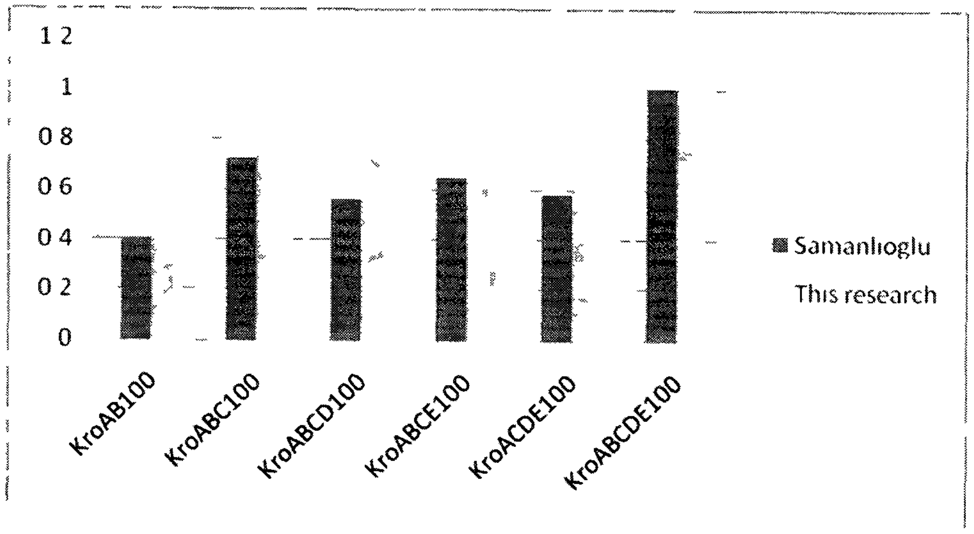


Figure 23. Comparison of ARE (N=50, G=62)

## 5.4 Conclusion

In this research, we proposed a hybrid Lehmer code Genetic Algorithm to solve multi-objective TSPs. This algorithm provided an alternative way to use Genetic Algorithms to solve discrete optimization problems. There are many researches using Random Keys Genetic Algorithms to solve discrete optimization problems, but not that many researches on Lehmer code Genetic Algorithms, especially using Lehmer code solving multi-objective combinatorial optimization problems.

The experiments showed that the proposed Lehmer code Genetic Algorithms worked well on multi-objective TSPs. The average relative excesses for all examples are less than 1%. And the results getting from the Lehmer code Genetic Algorithms are comparable to the results from Samanlioglu's Random keys Genetic Algorithms.

The new hybrid Lehmer code Genetic Algorithms has some advantages over Random keys Genetic Algorithms.

First, by using Lehmer code, the new hybrid Genetic Algorithms can easily overcome the difficulty with traditional Genetic Algorithms in keeping feasibility when crossover parent solutions to create offspring. As we can see in the experiments, the crossover operator directly work on the Lehmer code represented parent solutions and no repairing process needed for the offspring solutions. All the solutions obtained from crossover are feasible.

Furthermore, although the random keys Genetic Algorithms can also solve this difficulty, it loses the edge information at the same time when it solves the difficulty of keeping feasible solutions. This means, the random keys Genetic Algorithms can keep any schematic information from the parent solutions to the offspring. As we know, this schematic information is very important feature of Genetic Algorithms. And it's one of the reasons that makes Genetic Algorithms converge quicker. While by using Lehmer code representation, at least certain part of the edge information can be transferred from the parent solutions to the offspring. This is obvious in this research. In this new Genetic Algorithms, we set the crossover point between 0.7m and 0.9m (m is the number of the cities). This means that at least 70% of parent edge information will be kept from the parents to the offspring. For example, the following figure shows how Lehmer code representation one point crossover keeps part of parent information into the offspring solutions.

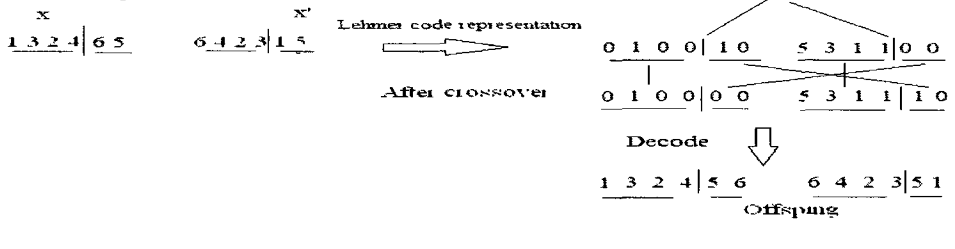


Figure 18. One Point Crossover of Lehmer Code GA (Keep Parent Information)

From the figure, we can see that the parent edge information, $1 \rightarrow 3 \rightarrow 2 \rightarrow 4$ and

6 → 4 → 2 → 3 were kept from the parent solutions to the offspring, respectively.

Finally, the implementation of the new Genetic Algorithms is very easy compared to the random keys Genetic Algorithms. For example, for the encoding and evaluation of the solutions, the random keys Genetic Algorithms must sort all cities and get the ranks of all cities first. This means that at least an extra sort process is needed. And you don't know the neighbors for a specific city before you sort all cities. While in Lehmer code Genetic Algorithms, the position of each code is the rank of the city represented by the code. And the city represented by the code can be easily gotten from the list {1, 2, 3, ... , n} (n is the total number of the cities) according to the code of the city. The neighbors of this city are the cities right before and behind it.

Another advantage of this new Genetic Algorithm is in tuning up the solutions. For example, in this research, I used an improvement operator in this new hybrid Genetic Algorithm. The idea is to improve the solution by randomly changing the locations of 4 cities. This operator can be very simply implemented by randomly picking up a number and decreasing or increasing the code which is located in the position of the random number by 3. The following figure shows how this works.

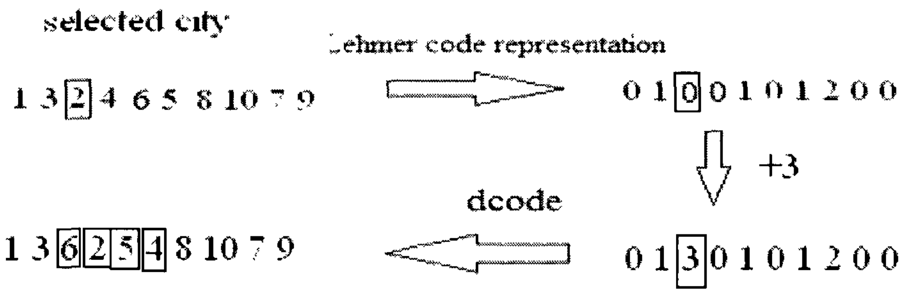


Figure 19. One Point Mutation

From this figure, we can see that three cities change position by simple adding 3 to one of the cities' Lehmer code. In random keys Genetic Algorithms, it is not so easy to implement the similar operator. For example, in Samanlioglu's Random Keys Genetic Algorithms, 4 cities were randomly picked up and all the combinations of these cities were taken into account. Then 2-opt heuristics worked on these possible solutions and selected the best one. Lastly, the above procedure was repeated 10 times and the one that had the best fitness value was selected.

Based on the above experiments and discussion, we can get the conclusion that the new hybrid Lehmer code Genetic Algorithms works well on multi-objective TSPs. This research provided an alternative way to use Genetic Algorithms to solve multi-objective combinatorial optimization problems. And it's very easy to implement this new hybrid Genetic Algorithm.

# CHAPTER 6

# FUTURE WORK AND CONCLUSION

## 6.1 Contributions of the Dissertation

This research proposed a new hybrid Lehmer code Genetic Algorithm to solve single objective and multi-objective TSPs. The major contributions of the dissertation are:

- After investigating the performance of traditional Genetic Algorithms and Random Keys Genetic Algorithms on Traveling Salesman Problems, a new Lehmer code Genetic Algorithms was proposed to solve both single objective and multi-objective TSPs. This provided an alternative way to use Genetic Algorithms to solve discrete optimization problems, especially Traveling Salesman Problems.
- Lehmer code was proposed to represent the potential solutions. By using Lehmer code representation, solutions created by using genetic operators are always feasible solutions. Another advantage by using Lehmer code representation is that certain parts of edge information can be retained from the parent solutions to the offspring.
- 2-opt and Non sequential 3-opt were proposed to conduct local search in this new Genetic Algorithm. By doing this, the new Genetic Algorithm has good exploitation ability while not increasing the computation time too much.
- The proposed Hybrid Lehmer code is very easy to implement compared to the traditional Genetic Algorithms and Random Keys Genetic Algorithms when solving discrete optimization problems. No additional fixing procedure is needed as in traditional Genetic Algorithms. And only traditional genetic operators: crossover and mutation were used.

- Matlab was used to implement this new hybrid Lehmer Code Genetic Algorithm.
- The new Genetic Algorithm was used to solve some bench mark single objective TSPs with up to 200 cities from the TSPLIB (Reinelt, 1995) and multi-objective TSPs with up to 5 objectives and 100 cities. The results getting from the proposed method were compared with the results from the newly proposed methods in the literature, more specifically; the results form Hansen and Samanlioglu's methods (Hansen, 2000) (Samanlioglu, Ferrell & Kurz, 2008).

The experiments showed that this new algorithm worked well on both single objective and multi-objective TSPs. Through the experiments and the comparison of the results from the proposed method and the results form Hansen and Samanlioglu's methods, we can see that the proposed method's performance on single objective TSPs is better than Samanlioglu's method even using smaller population size and fewer generations.

And for multi-objective TSPs, we can see that the proposed hybrid Genetic Algorithms got comparable solutions with Samanlioglu's method. And compared to Samanlioglu's method, the proposed algorithm is much easier for implementation.

## 6.2 Limitation and Future Work

But there is still some room to improve this new Genetic Algorithm:

- The results obtained from the new algorithm on multi-objective TSPs is only comparable to the results from the Random keys Genetic Algorithms while not much better than them.
- The performance of the new Genetic Algorithm on single objective TSPs with large number of cities is not as good as on that with small number of

cities.

- 3-opt is more time consuming compared to 2-opt. Although we only used non-sequential 3-opt in this new Genetic Algorithm, the efficiency of the algorithm was influenced a little bit because of the using of the 3-opt. In addition, we used to Matlab to implement the new Genetic Algorithm. As we know, Matlab is not good in dealing with loops. The efficiency of the algorithm was decreased further.
- Another limitation for this research is that we only used the new algorithm solving several single objective TSPs from TSPLIB and multi-object TSPs from Hansen and Samanlioglu's research. More experiments should be conducted on more single objective TSPs, multi-objective TSPs and other discrete optimization problems to test the robustness of the new Genetic Algorithm on discrete optimization problems.

The future work will be to find ways to improve the performance of the algorithm on multi- objective TSPs. As we discussed above, the Lehmer code Genetic Algorithms has some advantages over Random Keys Genetic Algorithms, such as, keeping edge information from parents to the offspring, easy implementation, etc.

One direction to improve the performance of this new algorithm is to add more complexity to the algorithms. As we discussed above, one of advantages of this new Genetic Algorithms is easy to implement. But this also means that there is some room for us to improve the performance of the algorithm. For example, this new Genetic Algorithms only use non sequential 3-opt heuristics. Maybe in the future, we can try to implement 3-opt heuristics to see if this can improve the performance of the new Genetic

Algorithms. This means that we will take into account of all the possible combination of the three parts of the router while not just taking into account of 1 of the 4 combinations. But this will definitely increase the running time of the algorithm.

And this new algorithm uses one point mutation. To implement this mutation, this algorithm only randomly changes 4 cities position and uses non sequential to improve it. But in Samanlioglu's random keys Genetic Algorithms 4 cities were randomly selected and all the combinations of these cities were improved by 2-opt. Then select the best one. This procedure was repeated 10 times. And the best solution for this repetition was selected as one of solutions in next generation. Maybe in the future, similar procedure can be implemented in the Lehmer code Genetic Algorithms proposed by this research. At least, we can repeat the one point mutation several times and select the best one to keep it to the next generation.

Another direction is to do sensitive analysis. We had already tested the algorithms by using different parameters. So far the parameters used in this research seem to have the best performance. But we did not try all the combinations of the parameters to test the performance of the algorithms because it takes time to run the model, especially if we need to run the algorithm 30 times. In the future, if we can run this algorithm in a distribute computer system, we may find a better combination of the parameters that can improve the performance of the algorithm.

And in the future, we can try to use the new Genetic Algorithms to solve the other optimization problems.

Lastly, we used Matlab to code the new Lehmer code Genetic Algorithms. As we know, Matlab does not perform well when dealing with loops. While in this algorithm, we need to use many loops for 2-opt and 3-opt heuristics. Maybe in the future, we can try to code this algorithm by using objective-oriented programming package, such as C++, Java, etc. This will greatly decrease the running time for this algorithm.

## REFERENCES

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The Traveling Salesman Problem: A Computational Study*. Princeton, New Jersey: Princeton University Press.

Bean, J. C. (1994). Genetic algorithms and random keys for sequencing and optimization. *ORSA Journal on Computing,* 6(2), 154–160.

Beasley, D., Bull, D. R., & Martin, R. R. (1993). An Overview of Genetic Algorithms: Part 1, Fundamentals. *University Computing,* 15(2), 58-69.

Bennett, K., Ferris, M.C., & Ioannidis, Y.E. (1991). A Genetic Algorithm for Database Query Optimization. *in Proceedings of the Fourth International Conference on Genetic Algorithms,* 400-407.

Bryant, K. (2000). *Genetic Algorithms and the Traveling Salesman Problem* (master's Thesis). Harvey Mudd College, Dept. of Mathematics.

Chiang, W. C., & Russell, R. A. (1997). A Reactive Tabu Search Meta-heuristic for the Vehicle Routing Problem with Time Windows. *INFORMS Journal on Computing,* 9(4), 417-430.

Chen, C.-C., Wang, L.-H., Kao, C.-Y., Ouhyoung, M. & Chen, W.-C. (1995). Molecular Binding: A Case Study of the Population-Based annealing Genetic Algorithms. *In IEEE International Conference on Evolutionary Computation ICEC'95.* Perth, Australia, 50–55.

Choy, K. L., Lam, E. W., & Lau, K.H. (1997-98). Application of fuzzied genetic algorithms in optimizing parameters in a manufacturing system for resource allocation. *Annual Journal of IIE (HK),* 11-16.

De Jong, K.A. (1975). *Analysis of the Behavior of a Class of Genetic Adaptive Systems* (Doctoral dissertation). University of Michigan, Ann Arbor.

De Jong, K.A. (1985). Genetic Algorithms: A 10 Year Perspective. *In Proceedings of the First International Conference on Genetic Algorithms and Their Applications,* 169-177.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms* (Doctoral dissertation). Politecnico di Milano, Italie.

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

Dorigo, M., & Caro, G. D. (1999). Ant Colony Optimization: A New Meta-Heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation*, 2, 1470-1477.

Eberhart, R., & Shi, Y. (2001). Particle swarm optimization: Development, Applications and Resources. *Proc. IEEE CEC*, 81-86.

Fonseca, C. M., & Fleming, P. J. (1993). Genetic algorithms for multiobjective optimization: formation, discussion and generation. *In Proceedings of the Fifth International Conference on Genetic Algorithms*, Morgan Kauffman, 416-423.

Garey, M. R., & Johnson, D. S. (1979*). Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, NY.

Geem, Z. W., & Kim, J. H. (2001). A New Heuristic Optimization Algorithm: Harmony Search, *Simulation*, 76(2), 60-68.

Ghoseiri, K., & Sahadi, H. (2008). A memetic algorithm for symmetric traveling salesman problem. *International Journal of Management Science and Engineering Management*, 3(4), 275-283.

Ghosh, D., Goldengorin, B., Gutin, G., & Jäger, G. (2007). Tolerance-based greedy algorithms for the traveling salesman problem. *Communications in DQM 10*, 52–70.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computer and Operations Research*, 13, 533-549.

Glover, F., & Laguna, M. (1997). *Tabu Search*. Norwell, Massachusetts: Kluwer Academic Publishers.

Glover, F., & Kochenberger, G. A. (2002). *Handbook of Meta-heuristics*. Norwell, Massachusetts: Kluwer Academic Publishers.

Goldberg, D. E. (1989). *Genetic Algorithm in Search, Optimization and Machine Learning*. Boston, Massachusetts: Addison-Wesley Longman.

Goldberg, D. E., & Lingle, R. (1985). Alleles, Loci and the Traveling Salesman Problem. *Proceedings of First International Conference on Genetic Algorithms*.

Grefenstette, J. J. (1986). Optimization of Control Parameters for Genetic Algorithms. *IEEE Transactions on Systems, Man and Cybernetics*, 16(1), 122-128.

Grefenstette, J. J., Gopal, R., Rosmaita, B. J., & Van Gucht, D. (1985). Genetic Algorithms for the Traveling Salesman Problem. *Proceedings of the First International Conference on Genetic Algorithms*, 160-165.

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

Hansen, M. P., & Jaszkiewicz, A. (1998). *Evaluation the quality of approximation to the non-dominated set*. IMM Technical Report IMM-REP-1998-7.

Hansen, M. P. (2000). Use of Substitute Scalarizing Functions to Guide a Local Search Based Heuristic: The Case of mo TSP. *Journal of Heuristics*, 6, 419-431.

Hatta, K., Wakabayashi, S., & Koide, T. (2001). Adaptation of Genetic Operators and Parameters of Genetic Algorithm Based on the Elite Degree of an Individual. *Systems and Computers in Japan*, 31(1), 29-37.

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

Herrera, F., Lozano, M., & Moraga, C. (1999). Hierarchical Distributed Genetic Algorithms. *International Journal of Intelligent Systems*, 14(11), 1099-1121.

Hertz, A., Taillard, E., & Werra, D. (1997). Tabu Search. In E. Aarts and J. K. Lenstra, (Eds.). *Local Search in Combinatorial Optimization* (pp.121-136). New York, NY: John Wiley & Sons, Ltd.

Hillier, F. S., & Lieberman, G. J. (2005). *Introduction to Operations Research (eighth edition)*. New York, NY: McGraw-Hill.

Homaifar, A., Guan, S., & Liepins, E. G. (1992). Schema analysis of the traveling salesman problem using genetic algorithms. *Complex Systems*, 6(2):183–217.

Holland, J. H. (1975). *Adaption in natural and artificial systems*. Ann Arbor, Michigan: U. of Michigan Press.

Javadi, A. A., & Tan, T. P. (2005). A hybrid intelligent genetic algorithm. *Advanced Engineering Informatics*, 19(4), 266-262.

Horn, J., Nafpliotis, N., & Goldberg, D. (1993). A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In *Proceedings of the First IEEE Conference on*

*Evolutionary Computation, IEEE World Congress on Computational Computation*, 1, 82-87.

Jog, P., Suh, J. Y., & van Gucht, D. (1989). The effects of population size, heuristic crossover, and local improvement on a genetic algorithm for the traveling salesman problem. In J. D. Schaffer (Eds.), *Proceedings of the Third International Conference on Genetic Algorithms* (pp. 110-115), San Mateo, CA: Morgan Kaufmann.

Johnson, D. S., & Mcgeoch, L.A. (1997). The Traveling Salesman Problem: A Case Study in Local Optimization. *Local Search in Combinatorial Optimization.* New York, NY: Wiley.

Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks* (pp. 1942-1948). NJ: Piscataway.

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4589), 671-680.

Lehmer, H. (1960). Teaching combinatorial tricks to a computer, In *Proceedings of Symposia in Applied Mathematics, Combinatorial Analysis*, 10, 179-193.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering*, 194(36-38), 3902-3933.

Li, J. H., & Li, M. (2008). Genetic Algorithm with Dual Species, In *Proceedings of the IEEE International Conference on Automation and Logistics* (pp. 2572-2575). Qingdao, China.

Lin, F. T., Kao, C. Y., & Hsu, C. C. (1991). Incorporating Genetic Algorithms into Simulated Annealing. In *Proceedings of the Fourth International Symposium on Artificial Intelligence* (pp. 290–297).

Mahdavi, M., Fesanghary, M., & Damangir, E. (2007). An improved harmony search algorithm for solving optimization problems. *Applied Mathematics and Computation*, 188 (2), 1567-1579.

Mahfoud, S. W., & Goldberg, D. E. (1995). Parallel Recombinative Simulated Annealing: A Genetic Algorithm. *Parallel Computing*, 21, 1–28.

Martin, O. C., Otto, S. W., & Felten, E. W. (1992). Large-Step Markov Chains for the TSP: Incorporating Local Search Heuristics. *Operation Research Letters*, 11, 219–224.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structure = Evolution Programs* (Third, Revised and Extended Edition). Berlin, Heidelberg, New York: Spring-Verlag.

Mishra, S. K. (2006). *Particle Swarm Method on Some Difficult Test Problems of Global Optimization*. MPRA Paper 1742, University Library of Munich, Germany. Retrieved from: http://ssrn.com/abstract=928538.

Morse, P. M., & Kimball, G.E. (1951). *Methods of Operations Research*. New York, NY: John Wiley & Sons.

Nissen, V. (1994). Solving the Quadratic Assignment problem with Clues from Nature. *IEEE Transactions on Neural Networks*, 5(1), 66–72.

Oliver, I. M., Smith, D. J., & Holland, J. R. C. (1987). A study of Permutation Crossover Operators on Traveling Salesman Problem, *Proceedings of Second International Conference on Genetic Algorithms*.

Osman, I. H., & Laporte, G. (1996). Meta-heuristics: A bibliography. *Annals of Operations Research*, 63, 513-623.

Pesko, S. (2006). Differential Evolution for Small TSPs with Constraints, In *Proceedings of 4th International Scientific Conference "Challenges in Transport and Communication"*.

Potter, M. A., & De Jong, K. A. (1994). A cooperative co-evolutionary approach to function optimization. In *Proceedings of the Third Parallel Problem Solving from Nature*, 249-257. Jerusalem, Israel.

Rudolph, G. (1994). Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions of Neural Networks*, 5(1), 96-101.

Singh, S. K., & Borah, M. (2009). A comparative study of Repulsive Particle Swarm Optimization and Simulated Annealing on some Numerical Bench Mark Problems. *International Journal of Computational Intelligence Research*, 5, 75-82.

Talbi, E. G., & Muntean, T. (1993). Hill-climbing, simulated annealing and genetic algorithms: a comparative study and application to mapping problems. *Proceeding of International Conference on System Science*, 565-573.

Tanese, R. (1989). Distributed genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 434–439.

Van den Bergh, F., & Engelbrecht, A. (2004). A Cooperative Approach to Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3), 225–239.

Vignaux, G. A., & Michalewicz, Z. (1991). A Genetic Algorithm for the Linear Transportation Problem, *IEEE Transaction on System, Man and Cybernetics*, 21(2), 445-452.

Wang, J., Zhang, Y., Zhou, Y., & Yin, J. (2008). Discrete Quantum-Behaved Particle Swarm Optimization Based on Estimated of Distribution for Combinatorial Optimization. *Proc. IEEE CEC*, pp. 897-904.

Wang, D. Y., & Wang, G. (2008). Parameters Optimization of Fuzzy Controller Based on Improved Particle Swarm Optimization, In *IEEE International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, pp. 917-921.

Wang, H., Wang, D., & Yang, S. (2009). A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems. *Soft Computing*, 13, 763-780.

Whitley, D., & Starkweather, T. (1990). Genitor II: a distributed genetic algorithm. *Journal of Experimental and Theoretical Artificial Intelligence*, 2, 189–214.

Wolpert, D. H., & Macready W. G. (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, 1(1), 67-82.

Yildiz, A. L. (2009). An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry. *Journal of Material Processing Technology*, 209, 2773-2780.

# APPENDIX

## COORDINATE FOR CITIES OF TSPS

KroA100

| | | |
|---|---|---|
| 1 | 1380 | 939 |
| 2 | 2848 | 96 |
| 3 | 3510 | 1671 |
| 4 | 457 | 334 |
| 5 | 3888 | 666 |
| 6 | 984 | 965 |
| 7 | 2721 | 1482 |
| 8 | 1286 | 525 |
| 9 | 2716 | 1432 |
| 10 | 738 | 1325 |
| 11 | 1251 | 1832 |
| 12 | 2728 | 1698 |
| 13 | 3815 | 169 |
| 14 | 3683 | 1533 |
| 15 | 1247 | 1945 |
| 16 | 123 | 862 |
| 17 | 1234 | 1946 |
| 18 | 252 | 1240 |
| 19 | 611 | 673 |
| 20 | 2576 | 1676 |
| 21 | 928 | 1700 |
| 22 | 53 | 857 |
| 23 | 1807 | 1711 |
| 24 | 274 | 1420 |
| 25 | 2574 | 946 |
| 26 | 178 | 24 |
| 27 | 2678 | 1825 |
| 28 | 1795 | 962 |
| 29 | 3384 | 1498 |
| 30 | 3520 | 1079 |
| 31 | 1256 | 61 |
| 32 | 1424 | 1728 |
| 33 | 3913 | 192 |
| 34 | 3085 | 1528 |
| 35 | 2573 | 1969 |
| 36 | 463 | 1670 |

| | | |
|---|---|---|
| 37 | 3875 | 598 |
| 38 | 298 | 1513 |
| 39 | 3479 | 821 |
| 40 | 2542 | 236 |
| 41 | 3955 | 1743 |
| 42 | 1323 | 280 |
| 43 | 3447 | 1830 |
| 44 | 2936 | 337 |
| 45 | 1621 | 1830 |
| 46 | 3373 | 1646 |
| 47 | 1393 | 1368 |
| 48 | 3874 | 1318 |
| 49 | 938 | 955 |
| 50 | 3022 | 474 |
| 51 | 2482 | 1183 |
| 52 | 3854 | 923 |
| 53 | 376 | 825 |
| 54 | 2519 | 135 |
| 55 | 2945 | 1622 |
| 56 | 953 | 268 |
| 57 | 2628 | 1479 |
| 58 | 2097 | 981 |
| 59 | 890 | 1846 |
| 60 | 2139 | 1806 |
| 61 | 2421 | 1007 |
| 62 | 2290 | 1810 |
| 63 | 1115 | 1052 |
| 64 | 2588 | 302 |
| 65 | 327 | 265 |
| 66 | 241 | 341 |
| 67 | 1917 | 687 |
| 68 | 2991 | 792 |
| 69 | 2573 | 599 |
| 70 | 19 | 674 |
| 71 | 3911 | 1673 |
| 72 | 872 | 1559 |
| 73 | 2863 | 558 |
| 74 | 929 | 1766 |
| 75 | 839 | 620 |
| 76 | 3893 | 102 |
| 77 | 2178 | 1619 |
| 78 | 3822 | 899 |
| 79 | 378 | 1048 |

```
          80    1178     100
          81    2599     901
          82    3416     143
          83    2961    1605
          84     611    1384
          85    3113     885
          86    2597    1830
          87    2586    1286
          88     161     906
          89    1429     134
          90     742    1025
          91    1625    1651
          92    1187     706
          93    1787    1009
          94      22     987
          95    3640      43
          96    3756     882
          97     776     392
          98    1724    1642
          99     198    1810
         100    3950    1558
EOF


KroA150
           1    1380     939
           2    2848      96
           3    3510    1671
           4     457     334
           5    3888     666
           6     984     965
           7    2721    1482
           8    1286     525
           9    2716    1432
          10     738    1325
          11    1251    1832
          12    2728    1698
          13    3815     169
          14    3683    1533
          15    1247    1945
          16     123     862
          17    1234    1946
```

| | | |
|---|---|---|
| 18 | 252 | 1240 |
| 19 | 611 | 673 |
| 20 | 2576 | 1676 |
| 21 | 928 | 1700 |
| 22 | 53 | 857 |
| 23 | 1807 | 1711 |
| 24 | 274 | 1420 |
| 25 | 2574 | 946 |
| 26 | 178 | 24 |
| 27 | 2678 | 1825 |
| 28 | 1795 | 962 |
| 29 | 3384 | 1498 |
| 30 | 3520 | 1079 |
| 31 | 1256 | 61 |
| 32 | 1424 | 1728 |
| 33 | 3913 | 192 |
| 34 | 3085 | 1528 |
| 35 | 2573 | 1969 |
| 36 | 463 | 1670 |
| 37 | 3875 | 598 |
| 38 | 298 | 1513 |
| 39 | 3479 | 821 |
| 40 | 2542 | 236 |
| 41 | 3955 | 1743 |
| 42 | 1323 | 280 |
| 43 | 3447 | 1830 |
| 44 | 2936 | 337 |
| 45 | 1621 | 1830 |
| 46 | 3373 | 1646 |
| 47 | 1393 | 1368 |
| 48 | 3874 | 1318 |
| 49 | 938 | 955 |
| 50 | 3022 | 474 |
| 51 | 2482 | 1183 |
| 52 | 3854 | 923 |
| 53 | 376 | 825 |
| 54 | 2519 | 135 |
| 55 | 2945 | 1622 |
| 56 | 953 | 268 |
| 57 | 2628 | 1479 |
| 58 | 2097 | 981 |
| 59 | 890 | 1846 |
| 60 | 2139 | 1806 |

| | | |
|---|---|---|
| 61 | 2421 | 1007 |
| 62 | 2290 | 1810 |
| 63 | 1115 | 1052 |
| 64 | 2588 | 302 |
| 65 | 327 | 265 |
| 66 | 241 | 341 |
| 67 | 1917 | 687 |
| 68 | 2991 | 792 |
| 69 | 2573 | 599 |
| 70 | 19 | 674 |
| 71 | 3911 | 1673 |
| 72 | 872 | 1559 |
| 73 | 2863 | 558 |
| 74 | 929 | 1766 |
| 75 | 839 | 620 |
| 76 | 3893 | 102 |
| 77 | 2178 | 1619 |
| 78 | 3822 | 899 |
| 79 | 378 | 1048 |
| 80 | 1178 | 100 |
| 81 | 2599 | 901 |
| 82 | 3416 | 143 |
| 83 | 2961 | 1605 |
| 84 | 611 | 1384 |
| 85 | 3113 | 885 |
| 86 | 2597 | 1830 |
| 87 | 2586 | 1286 |
| 88 | 161 | 906 |
| 89 | 1429 | 134 |
| 90 | 742 | 1025 |
| 91 | 1625 | 1651 |
| 92 | 1187 | 706 |
| 93 | 1787 | 1009 |
| 94 | 22 | 987 |
| 95 | 3640 | 43 |
| 96 | 3756 | 882 |
| 97 | 776 | 392 |
| 98 | 1724 | 1642 |
| 99 | 198 | 1810 |
| 100 | 3950 | 1558 |
| 101 | 3477 | 949 |
| 102 | 91 | 1732 |
| 103 | 3972 | 329 |

| | | |
|---|---|---|
| 104 | 198 | 1632 |
| 105 | 1806 | 733 |
| 106 | 538 | 1023 |
| 107 | 3430 | 1088 |
| 108 | 2186 | 766 |
| 109 | 1513 | 1646 |
| 110 | 2143 | 1611 |
| 111 | 53 | 1657 |
| 112 | 3404 | 1307 |
| 113 | 1034 | 1344 |
| 114 | 2823 | 376 |
| 115 | 3104 | 1931 |
| 116 | 3232 | 324 |
| 117 | 2790 | 1457 |
| 118 | 374 | 9 |
| 119 | 741 | 146 |
| 120 | 3083 | 1938 |
| 121 | 3502 | 1067 |
| 122 | 1280 | 237 |
| 123 | 3326 | 1846 |
| 124 | 217 | 38 |
| 125 | 2503 | 1172 |
| 126 | 3527 | 41 |
| 127 | 739 | 1850 |
| 128 | 3548 | 1999 |
| 129 | 48 | 154 |
| 130 | 1419 | 872 |
| 131 | 1689 | 1223 |
| 132 | 3468 | 1404 |
| 133 | 1628 | 253 |
| 134 | 382 | 872 |
| 135 | 3029 | 1242 |
| 136 | 3646 | 1758 |
| 137 | 285 | 1029 |
| 138 | 1782 | 93 |
| 139 | 1067 | 371 |
| 140 | 2849 | 1214 |
| 141 | 920 | 1835 |
| 142 | 1741 | 712 |
| 143 | 876 | 220 |
| 144 | 2753 | 283 |
| 145 | 2609 | 1286 |
| 146 | 3941 | 258 |

```
          147       3613        523
          148       1754        559
          149       2916       1724
          150       2445       1820
EOF

KroA
200

1           1357      1905
2           2650      802
3           1774      107
4           1307      964
5           3806      746
6           2687      1353
7           43        1957
8           3092      1668
9           185       1542
10          834       629
11          40        462
12          1183      1391
13          2048      1628
14          1097      643
15          1838      1732
16          234       1118
17          3314      1881
18          737       1285
19          779       777
20          2312      1949
21          2576      189
22          3078      1541
23          2781      478
24          705       1812
25          3409      1917
26          323       1714
27          1660      1556
28          3729      1188
29          693       1383
30          2361      640
31          2433      1538
32          554       1825
33          913       317
34          3586      1909
```

| | | |
|---|---|---|
| 35 | 2636 | 727 |
| 36 | 1000 | 457 |
| 37 | 482 | 1337 |
| 38 | 3704 | 1082 |
| 39 | 3635 | 1174 |
| 40 | 1362 | 1526 |
| 41 | 2049 | 417 |
| 42 | 2552 | 1909 |
| 43 | 3939 | 640 |
| 44 | 219 | 898 |
| 45 | 812 | 351 |
| 46 | 901 | 1552 |
| 47 | 2513 | 1572 |
| 48 | 242 | 584 |
| 49 | 826 | 1226 |
| 50 | 3278 | 799 |
| 51 | 86 | 1065 |
| 52 | 14 | 454 |
| 53 | 1327 | 1893 |
| 54 | 2773 | 1286 |
| 55 | 2469 | 1838 |
| 56 | 3835 | 963 |
| 57 | 1031 | 428 |
| 58 | 3853 | 1712 |
| 59 | 1868 | 197 |
| 60 | 1544 | 863 |
| 61 | 457 | 1607 |
| 62 | 3174 | 1064 |
| 63 | 192 | 1004 |
| 64 | 2318 | 1925 |
| 65 | 2232 | 1374 |
| 66 | 396 | 828 |
| 67 | 2365 | 1649 |
| 68 | 2499 | 658 |
| 69 | 1410 | 307 |
| 70 | 2990 | 214 |
| 71 | 3646 | 1018 |
| 72 | 3394 | 1028 |
| 73 | 1779 | 90 |
| 74 | 1058 | 372 |
| 75 | 2933 | 1459 |
| 76 | 3099 | 173 |
| 77 | 2178 | 978 |

| | | |
|---|---|---|
| 78 | 138 | 1610 |
| 79 | 2082 | 1753 |
| 80 | 2302 | 1127 |
| 81 | 805 | 272 |
| 82 | 22 | 1617 |
| 83 | 3213 | 1085 |
| 84 | 99 | 536 |
| 85 | 1533 | 1780 |
| 86 | 3564 | 676 |
| 87 | 29 | 6 |
| 88 | 3808 | 1375 |
| 89 | 2221 | 291 |
| 90 | 3499 | 1885 |
| 91 | 3124 | 408 |
| 92 | 781 | 671 |
| 93 | 1027 | 1041 |
| 94 | 3249 | 378 |
| 95 | 3297 | 491 |
| 96 | 213 | 220 |
| 97 | 721 | 186 |
| 98 | 3736 | 1542 |
| 99 | 868 | 731 |
| 100 | 960 | 303 |
| 101 | 1380 | 939 |
| 102 | 2848 | 96 |
| 103 | 3510 | 1671 |
| 104 | 457 | 334 |
| 105 | 3888 | 666 |
| 106 | 984 | 965 |
| 107 | 2721 | 1482 |
| 108 | 1286 | 525 |
| 109 | 2716 | 1432 |
| 110 | 738 | 1325 |
| 111 | 1251 | 1832 |
| 112 | 2728 | 1698 |
| 113 | 3815 | 169 |
| 114 | 3683 | 1533 |
| 115 | 1247 | 1945 |
| 116 | 123 | 862 |
| 117 | 1234 | 1946 |
| 118 | 252 | 1240 |
| 119 | 611 | 673 |
| 120 | 2576 | 1676 |

| | | |
|---|---|---|
| 121 | 928 | 1700 |
| 122 | 53 | 857 |
| 123 | 1807 | 1711 |
| 124 | 274 | 1420 |
| 125 | 2574 | 946 |
| 126 | 178 | 24 |
| 127 | 2678 | 1825 |
| 128 | 1795 | 962 |
| 129 | 3384 | 1498 |
| 130 | 3520 | 1079 |
| 131 | 1256 | 61 |
| 132 | 1424 | 1728 |
| 133 | 3913 | 192 |
| 134 | 3085 | 1528 |
| 135 | 2573 | 1969 |
| 136 | 463 | 1670 |
| 137 | 3875 | 598 |
| 138 | 298 | 1513 |
| 139 | 3479 | 821 |
| 140 | 2542 | 236 |
| 141 | 3955 | 1743 |
| 142 | 1323 | 280 |
| 143 | 3447 | 1830 |
| 144 | 2936 | 337 |
| 145 | 1621 | 1830 |
| 146 | 3373 | 1646 |
| 147 | 1393 | 1368 |
| 148 | 3874 | 1318 |
| 149 | 938 | 955 |
| 150 | 3022 | 474 |
| 151 | 2482 | 1183 |
| 152 | 3854 | 923 |
| 153 | 376 | 825 |
| 154 | 2519 | 135 |
| 155 | 2945 | 1622 |
| 156 | 953 | 268 |
| 157 | 2628 | 1479 |
| 158 | 2097 | 981 |
| 159 | 890 | 1846 |
| 160 | 2139 | 1806 |
| 161 | 2421 | 1007 |
| 162 | 2290 | 1810 |
| 163 | 1115 | 1052 |

```
164        2588     302
165        327      265
166        241      341
167        1917     687
168        2991     792
169        2573     599
170        19       674
171        3911     1673
172        872      1559
173        2863     558
174        929      1766
175        839      620
176        3893     102
177        2178     1619
178        3822     899
179        378      1048
180        1178     100
181        2599     901
182        3416     143
183        2961     1605
184        611      1384
185        3113     885
186        2597     1830
187        2586     1286
188        161      906
189        1429     134
190        742      1025
191        1625     1651
192        1187     706
193        1787     1009
194        22       987
195        3640     43
196        3756     882
197        776      392
198        1724     1642
199        198      1810
200        3950     1558
EOF
```

```
KroB100
          1      3140       1401
```

| | | |
|---|---|---|
| 2 | 556 | 1056 |
| 3 | 3675 | 1522 |
| 4 | 1182 | 1853 |
| 5 | 3595 | 111 |
| 6 | 962 | 1895 |
| 7 | 2030 | 1186 |
| 8 | 3507 | 1851 |
| 9 | 2642 | 1269 |
| 10 | 3438 | 901 |
| 11 | 3858 | 1472 |
| 12 | 2937 | 1568 |
| 13 | 376 | 1018 |
| 14 | 839 | 1355 |
| 15 | 706 | 1925 |
| 16 | 749 | 920 |
| 17 | 298 | 615 |
| 18 | 694 | 552 |
| 19 | 387 | 190 |
| 20 | 2801 | 695 |
| 21 | 3133 | 1143 |
| 22 | 1517 | 266 |
| 23 | 1538 | 224 |
| 24 | 844 | 520 |
| 25 | 2639 | 1239 |
| 26 | 3123 | 217 |
| 27 | 2489 | 1520 |
| 28 | 3834 | 1827 |
| 29 | 3417 | 1808 |
| 30 | 2938 | 543 |
| 31 | 71 | 1323 |
| 32 | 3245 | 1828 |
| 33 | 731 | 1741 |
| 34 | 2312 | 1270 |
| 35 | 2426 | 1851 |
| 36 | 380 | 478 |
| 37 | 2310 | 635 |
| 38 | 2830 | 775 |
| 39 | 3829 | 513 |
| 40 | 3684 | 445 |
| 41 | 171 | 514 |
| 42 | 627 | 1261 |
| 43 | 1490 | 1123 |
| 44 | 61 | 81 |

| | | |
|---|---|---|
| 45 | 422 | 542 |
| 46 | 2698 | 1221 |
| 47 | 2372 | 127 |
| 48 | 177 | 1390 |
| 49 | 3084 | 748 |
| 50 | 1213 | 910 |
| 51 | 3 | 1817 |
| 52 | 1782 | 995 |
| 53 | 3896 | 742 |
| 54 | 1829 | 812 |
| 55 | 1286 | 550 |
| 56 | 3017 | 108 |
| 57 | 2132 | 1432 |
| 58 | 2000 | 1110 |
| 59 | 3317 | 1966 |
| 60 | 1729 | 1498 |
| 61 | 2408 | 1747 |
| 62 | 3292 | 152 |
| 63 | 193 | 1210 |
| 64 | 782 | 1462 |
| 65 | 2503 | 352 |
| 66 | 1697 | 1924 |
| 67 | 3821 | 147 |
| 68 | 3370 | 791 |
| 69 | 3162 | 367 |
| 70 | 3938 | 516 |
| 71 | 2741 | 1583 |
| 72 | 2330 | 741 |
| 73 | 3918 | 1088 |
| 74 | 1794 | 1589 |
| 75 | 2929 | 485 |
| 76 | 3453 | 1998 |
| 77 | 896 | 705 |
| 78 | 399 | 850 |
| 79 | 2614 | 195 |
| 80 | 2800 | 653 |
| 81 | 2630 | 20 |
| 82 | 563 | 1513 |
| 83 | 1090 | 1652 |
| 84 | 2009 | 1163 |
| 85 | 3876 | 1165 |
| 86 | 3084 | 774 |
| 87 | 1526 | 1612 |

```
          88     1612      328
          89     1423     1322
          90     3058     1276
          91     3782     1865
          92      347      252
          93     3904     1444
          94     2191     1579
          95     3220     1454
          96      468      319
          97     3611     1968
          98     3114     1629
          99     3515     1892
         100     3060      155
EOF

KroB150
         1     1357     1905
         2     2650      802
         3     1774      107
         4     1307      964
         5     3806      746
         6     2687     1353
         7       43     1957
         8     3092     1668
         9      185     1542
        10      834      629
        11       40      462
        12     1183     1391
        13     2048     1628
        14     1097      643
        15     1838     1732
        16      234     1118
        17     3314     1881
        18      737     1285
        19      779      777
        20     2312     1949
        21     2576      189
        22     3078     1541
        23     2781      478
        24      705     1812
        25     3409     1917
        26      323     1714
        27     1660     1556
```

| | | |
|---|---|---|
| 28 | 3729 | 1188 |
| 29 | 693 | 1383 |
| 30 | 2361 | 640 |
| 31 | 2433 | 1538 |
| 32 | 554 | 1825 |
| 33 | 913 | 317 |
| 34 | 3586 | 1909 |
| 35 | 2636 | 727 |
| 36 | 1000 | 457 |
| 37 | 482 | 1337 |
| 38 | 3704 | 1082 |
| 39 | 3635 | 1174 |
| 40 | 1362 | 1526 |
| 41 | 2049 | 417 |
| 42 | 2552 | 1909 |
| 43 | 3939 | 640 |
| 44 | 219 | 898 |
| 45 | 812 | 351 |
| 46 | 901 | 1552 |
| 47 | 2513 | 1572 |
| 48 | 242 | 584 |
| 49 | 826 | 1226 |
| 50 | 3278 | 799 |
| 51 | 86 | 1065 |
| 52 | 14 | 454 |
| 53 | 1327 | 1893 |
| 54 | 2773 | 1286 |
| 55 | 2469 | 1838 |
| 56 | 3835 | 963 |
| 57 | 1031 | 428 |
| 58 | 3853 | 1712 |
| 59 | 1868 | 197 |
| 60 | 1544 | 863 |
| 61 | 457 | 1607 |
| 62 | 3174 | 1064 |
| 63 | 192 | 1004 |
| 64 | 2318 | 1925 |
| 65 | 2232 | 1374 |
| 66 | 396 | 828 |
| 67 | 2365 | 1649 |
| 68 | 2499 | 658 |
| 69 | 1410 | 307 |
| 70 | 2990 | 214 |

| | | |
|---|---|---|
| 71 | 3646 | 1018 |
| 72 | 3394 | 1028 |
| 73 | 1779 | 90 |
| 74 | 1058 | 372 |
| 75 | 2933 | 1459 |
| 76 | 3099 | 173 |
| 77 | 2178 | 978 |
| 78 | 138 | 1610 |
| 79 | 2082 | 1753 |
| 80 | 2302 | 1127 |
| 81 | 805 | 272 |
| 82 | 22 | 1617 |
| 83 | 3213 | 1085 |
| 84 | 99 | 536 |
| 85 | 1533 | 1780 |
| 86 | 3564 | 676 |
| 87 | 29 | 6 |
| 88 | 3808 | 1375 |
| 89 | 2221 | 291 |
| 90 | 3499 | 1885 |
| 91 | 3124 | 408 |
| 92 | 781 | 671 |
| 93 | 1027 | 1041 |
| 94 | 3249 | 378 |
| 95 | 3297 | 491 |
| 96 | 213 | 220 |
| 97 | 721 | 186 |
| 98 | 3736 | 1542 |
| 99 | 868 | 731 |
| 100 | 960 | 303 |
| 101 | 3825 | 1101 |
| 102 | 2779 | 435 |
| 103 | 201 | 693 |
| 104 | 2502 | 1274 |
| 105 | 765 | 833 |
| 106 | 3105 | 1823 |
| 107 | 1937 | 1400 |
| 108 | 3364 | 1498 |
| 109 | 3702 | 1624 |
| 110 | 2164 | 1874 |
| 111 | 3019 | 189 |
| 112 | 3098 | 1594 |
| 113 | 3239 | 1376 |

```
     114     3359     1693
     115     2081     1011
     116     1398     1100
     117      618     1953
     118     1878       59
     119     3803      886
     120      397     1217
     121     3035      152
     122     2502      146
     123     3230      380
     124     3479     1023
     125      958     1670
     126     3423     1241
     127       78     1066
     128       96      691
     129     3431       78
     130     2053     1461
     131     3048        1
     132      571     1711
     133     3393      782
     134     2835     1472
     135      144     1185
     136      923      108
     137      989     1997
     138     3061     1211
     139     2977       39
     140     1668      658
     141      878      715
     142      678     1599
     143     1086      868
     144      640      110
     145     3551     1673
     146      106     1267
     147     2243     1332
     148     3796     1401
     149     2643     1320
     150       48      267
EOF

Kro200
       1     3140     1401
       2      556     1056
       3     3675     1522
```

| | | |
|---|---|---|
| 4 | 1182 | 1853 |
| 5 | 3595 | 111 |
| 6 | 962 | 1895 |
| 7 | 2030 | 1186 |
| 8 | 3507 | 1851 |
| 9 | 2642 | 1269 |
| 10 | 3438 | 901 |
| 11 | 3858 | 1472 |
| 12 | 2937 | 1568 |
| 13 | 376 | 1018 |
| 14 | 839 | 1355 |
| 15 | 706 | 1925 |
| 16 | 749 | 920 |
| 17 | 298 | 615 |
| 18 | 694 | 552 |
| 19 | 387 | 190 |
| 20 | 2801 | 695 |
| 21 | 3133 | 1143 |
| 22 | 1517 | 266 |
| 23 | 1538 | 224 |
| 24 | 844 | 520 |
| 25 | 2639 | 1239 |
| 26 | 3123 | 217 |
| 27 | 2489 | 1520 |
| 28 | 3834 | 1827 |
| 29 | 3417 | 1808 |
| 30 | 2938 | 543 |
| 31 | 71 | 1323 |
| 32 | 3245 | 1828 |
| 33 | 731 | 1741 |
| 34 | 2312 | 1270 |
| 35 | 2426 | 1851 |
| 36 | 380 | 478 |
| 37 | 2310 | 635 |
| 38 | 2830 | 775 |
| 39 | 3829 | 513 |
| 40 | 3684 | 445 |
| 41 | 171 | 514 |
| 42 | 627 | 1261 |
| 43 | 1490 | 1123 |
| 44 | 61 | 81 |
| 45 | 422 | 542 |
| 46 | 2698 | 1221 |

| | | |
|---|---|---|
| 47 | 2372 | 127 |
| 48 | 177 | 1390 |
| 49 | 3084 | 748 |
| 50 | 1213 | 910 |
| 51 | 3 | 1817 |
| 52 | 1782 | 995 |
| 53 | 3896 | 742 |
| 54 | 1829 | 812 |
| 55 | 1286 | 550 |
| 56 | 3017 | 108 |
| 57 | 2132 | 1432 |
| 58 | 2000 | 1110 |
| 59 | 3317 | 1966 |
| 60 | 1729 | 1498 |
| 61 | 2408 | 1747 |
| 62 | 3292 | 152 |
| 63 | 193 | 1210 |
| 64 | 782 | 1462 |
| 65 | 2503 | 352 |
| 66 | 1697 | 1924 |
| 67 | 3821 | 147 |
| 68 | 3370 | 791 |
| 69 | 3162 | 367 |
| 70 | 3938 | 516 |
| 71 | 2741 | 1583 |
| 72 | 2330 | 741 |
| 73 | 3918 | 1088 |
| 74 | 1794 | 1589 |
| 75 | 2929 | 485 |
| 76 | 3453 | 1998 |
| 77 | 896 | 705 |
| 78 | 399 | 850 |
| 79 | 2614 | 195 |
| 80 | 2800 | 653 |
| 81 | 2630 | 20 |
| 82 | 563 | 1513 |
| 83 | 1090 | 1652 |
| 84 | 2009 | 1163 |
| 85 | 3876 | 1165 |
| 86 | 3084 | 774 |
| 87 | 1526 | 1612 |
| 88 | 1612 | 328 |
| 89 | 1423 | 1322 |

| | | |
|---|---|---|
| 133 | 1646 | 1817 |
| 134 | 2993 | 624 |
| 135 | 547 | 25 |
| 136 | 3373 | 1902 |
| 137 | 460 | 267 |
| 138 | 3060 | 781 |
| 139 | 1828 | 456 |
| 140 | 1021 | 962 |
| 141 | 2347 | 388 |
| 142 | 3535 | 1112 |
| 143 | 1529 | 581 |
| 144 | 1203 | 385 |
| 145 | 1787 | 1902 |
| 146 | 2740 | 1101 |
| 147 | 555 | 1753 |
| 148 | 47 | 363 |
| 149 | 3935 | 540 |
| 150 | 3062 | 329 |
| 151 | 387 | 199 |
| 152 | 2901 | 920 |
| 153 | 931 | 512 |
| 154 | 1766 | 692 |
| 155 | 401 | 980 |
| 156 | 149 | 1629 |
| 157 | 2214 | 1977 |
| 158 | 3805 | 1619 |
| 159 | 1179 | 969 |
| 160 | 1017 | 333 |
| 161 | 2834 | 1512 |
| 162 | 634 | 294 |
| 163 | 1819 | 814 |
| 164 | 1393 | 859 |
| 165 | 1768 | 1578 |
| 166 | 3023 | 871 |
| 167 | 3248 | 1906 |
| 168 | 1632 | 1742 |
| 169 | 2223 | 990 |
| 170 | 3868 | 697 |
| 171 | 1541 | 354 |
| 172 | 2374 | 1944 |
| 173 | 1962 | 389 |
| 174 | 3007 | 1524 |
| 175 | 3220 | 1945 |

| | | |
|---|---|---|
| 90 | 3058 | 1276 |
| 91 | 3782 | 1865 |
| 92 | 347 | 252 |
| 93 | 3904 | 1444 |
| 94 | 2191 | 1579 |
| 95 | 3220 | 1454 |
| 96 | 468 | 319 |
| 97 | 3611 | 1968 |
| 98 | 3114 | 1629 |
| 99 | 3515 | 1892 |
| 100 | 3060 | 155 |
| 101 | 2995 | 264 |
| 102 | 202 | 233 |
| 103 | 981 | 848 |
| 104 | 1346 | 408 |
| 105 | 781 | 670 |
| 106 | 1009 | 1001 |
| 107 | 2927 | 1777 |
| 108 | 2982 | 949 |
| 109 | 555 | 1121 |
| 110 | 464 | 1302 |
| 111 | 3452 | 637 |
| 112 | 571 | 1982 |
| 113 | 2656 | 128 |
| 114 | 1623 | 1723 |
| 115 | 2067 | 694 |
| 116 | 1725 | 927 |
| 117 | 3600 | 459 |
| 118 | 1109 | 1196 |
| 119 | 366 | 339 |
| 120 | 778 | 1282 |
| 121 | 386 | 1616 |
| 122 | 3918 | 1217 |
| 123 | 3332 | 1049 |
| 124 | 2597 | 349 |
| 125 | 811 | 1295 |
| 126 | 241 | 1069 |
| 127 | 2658 | 360 |
| 128 | 394 | 1944 |
| 129 | 3786 | 1862 |
| 130 | 264 | 36 |
| 131 | 2050 | 1833 |
| 132 | 3538 | 125 |

```
        176     2356    1568
        177     1604     706
        178     2028    1736
        179     2581     121
        180     2221    1578
        181     2944     632
        182     1082    1561
        183      997     942
        184     2334     523
        185     1264    1090
        186     1699    1294
        187      235    1059
        188     2592     248
        189     3642     699
        190     3599     514
        191     1766     678
        192      240     619
        193     1272     246
        194     3503     301
        195       80    1533
        196     1677    1238
        197     3766     154
        198     3946     459
        199     1994    1852
        200      278     165
EOF

KroC 100


           1     1357    1905
           2     2650     802
           3     1774     107
           4     1307     964
           5     3806     746
           6     2687    1353
           7       43    1957
           8     3092    1668
           9      185    1542
          10      834     629
          11       40     462
          12     1183    1391
          13     2048    1628
```

| | | |
|---|---|---|
| 14 | 1097 | 643 |
| 15 | 1838 | 1732 |
| 16 | 234 | 1118 |
| 17 | 3314 | 1881 |
| 18 | 737 | 1285 |
| 19 | 779 | 777 |
| 20 | 2312 | 1949 |
| 21 | 2576 | 189 |
| 22 | 3078 | 1541 |
| 23 | 2781 | 478 |
| 24 | 705 | 1812 |
| 25 | 3409 | 1917 |
| 26 | 323 | 1714 |
| 27 | 1660 | 1556 |
| 28 | 3729 | 1188 |
| 29 | 693 | 1383 |
| 30 | 2361 | 640 |
| 31 | 2433 | 1538 |
| 32 | 554 | 1825 |
| 33 | 913 | 317 |
| 34 | 3586 | 1909 |
| 35 | 2636 | 727 |
| 36 | 1000 | 457 |
| 37 | 482 | 1337 |
| 38 | 3704 | 1082 |
| 39 | 3635 | 1174 |
| 40 | 1362 | 1526 |
| 41 | 2049 | 417 |
| 42 | 2552 | 1909 |
| 43 | 3939 | 640 |
| 44 | 219 | 898 |
| 45 | 812 | 351 |
| 46 | 901 | 1552 |
| 47 | 2513 | 1572 |
| 48 | 242 | 584 |
| 49 | 826 | 1226 |
| 50 | 3278 | 799 |
| 51 | 86 | 1065 |
| 52 | 14 | 454 |
| 53 | 1327 | 1893 |
| 54 | 2773 | 1286 |
| 55 | 2469 | 1838 |
| 56 | 3835 | 963 |

| | | |
|---|---|---|
| 57 | 1031 | 428 |
| 58 | 3853 | 1712 |
| 59 | 1868 | 197 |
| 60 | 1544 | 863 |
| 61 | 457 | 1607 |
| 62 | 3174 | 1064 |
| 63 | 192 | 1004 |
| 64 | 2318 | 1925 |
| 65 | 2232 | 1374 |
| 66 | 396 | 828 |
| 67 | 2365 | 1649 |
| 68 | 2499 | 658 |
| 69 | 1410 | 307 |
| 70 | 2990 | 214 |
| 71 | 3646 | 1018 |
| 72 | 3394 | 1028 |
| 73 | 1779 | 90 |
| 74 | 1058 | 372 |
| 75 | 2933 | 1459 |
| 76 | 3099 | 173 |
| 77 | 2178 | 978 |
| 78 | 138 | 1610 |
| 79 | 2082 | 1753 |
| 80 | 2302 | 1127 |
| 81 | 805 | 272 |
| 82 | 22 | 1617 |
| 83 | 3213 | 1085 |
| 84 | 99 | 536 |
| 85 | 1533 | 1780 |
| 86 | 3564 | 676 |
| 87 | 29 | 6 |
| 88 | 3808 | 1375 |
| 89 | 2221 | 291 |
| 90 | 3499 | 1885 |
| 91 | 3124 | 408 |
| 92 | 781 | 671 |
| 93 | 1027 | 1041 |
| 94 | 3249 | 378 |
| 95 | 3297 | 491 |
| 96 | 213 | 220 |
| 97 | 721 | 186 |
| 98 | 3736 | 1542 |
| 99 | 868 | 731 |

```
      100      960      303
EOF

KroD100
        1     2995      264
        2      202      233
        3      981      848
        4     1346      408
        5      781      670
        6     1009     1001
        7     2927     1777
        8     2982      949
        9      555     1121
       10      464     1302
       11     3452      637
       12      571     1982
       13     2656      128
       14     1623     1723
       15     2067      694
       16     1725      927
       17     3600      459
       18     1109     1196
       19      366      339
       20      778     1282
       21      386     1616
       22     3918     1217
       23     3332     1049
       24     2597      349
       25      811     1295
       26      241     1069
       27     2658      360
       28      394     1944
       29     3786     1862
       30      264       36
       31     2050     1833
       32     3538      125
       33     1646     1817
       34     2993      624
       35      547       25
       36     3373     1902
       37      460      267
       38     3060      781
       39     1828      456
```

| | | |
|---|---|---|
| 40 | 1021 | 962 |
| 41 | 2347 | 388 |
| 42 | 3535 | 1112 |
| 43 | 1529 | 581 |
| 44 | 1203 | 385 |
| 45 | 1787 | 1902 |
| 46 | 2740 | 1101 |
| 47 | 555 | 1753 |
| 48 | 47 | 363 |
| 49 | 3935 | 540 |
| 50 | 3062 | 329 |
| 51 | 387 | 199 |
| 52 | 2901 | 920 |
| 53 | 931 | 512 |
| 54 | 1766 | 692 |
| 55 | 401 | 980 |
| 56 | 149 | 1629 |
| 57 | 2214 | 1977 |
| 58 | 3805 | 1619 |
| 59 | 1179 | 969 |
| 60 | 1017 | 333 |
| 61 | 2834 | 1512 |
| 62 | 634 | 294 |
| 63 | 1819 | 814 |
| 64 | 1393 | 859 |
| 65 | 1768 | 1578 |
| 66 | 3023 | 871 |
| 67 | 3248 | 1906 |
| 68 | 1632 | 1742 |
| 69 | 2223 | 990 |
| 70 | 3868 | 697 |
| 71 | 1541 | 354 |
| 72 | 2374 | 1944 |
| 73 | 1962 | 389 |
| 74 | 3007 | 1524 |
| 75 | 3220 | 1945 |
| 76 | 2356 | 1568 |
| 77 | 1604 | 706 |
| 78 | 2028 | 1736 |
| 79 | 2581 | 121 |
| 80 | 2221 | 1578 |
| 81 | 2944 | 632 |
| 82 | 1082 | 1561 |

```
        83      997      942
        84     2334      523
        85     1264     1090
        86     1699     1294
        87      235     1059
        88     2592      248
        89     3642      699
        90     3599      514
        91     1766      678
        92      240      619
        93     1272      246
        94     3503      301
        95       80     1533
        96     1677     1238
        97     3766      154
        98     3946      459
        99     1994     1852
       100      278      165
EOF


KroE100
         1     3477      949
         2       91     1732
         3     3972      329
         4      198     1632
         5     1806      733
         6      538     1023
         7     3430     1088
         8     2186      766
         9     1513     1646
        10     2143     1611
        11       53     1657
        12     3404     1307
        13     1034     1344
        14     2823      376
        15     3104     1931
        16     3232      324
        17     2790     1457
        18      374        9
        19      741      146
        20     3083     1938
```

| | | |
|---|---|---|
| 21 | 3502 | 1067 |
| 22 | 1280 | 237 |
| 23 | 3326 | 1846 |
| 24 | 217 | 38 |
| 25 | 2503 | 1172 |
| 26 | 3527 | 41 |
| 27 | 739 | 1850 |
| 28 | 3548 | 1999 |
| 29 | 48 | 154 |
| 30 | 1419 | 872 |
| 31 | 1689 | 1223 |
| 32 | 3468 | 1404 |
| 33 | 1628 | 253 |
| 34 | 382 | 872 |
| 35 | 3029 | 1242 |
| 36 | 3646 | 1758 |
| 37 | 285 | 1029 |
| 38 | 1782 | 93 |
| 39 | 1067 | 371 |
| 40 | 2849 | 1214 |
| 41 | 920 | 1835 |
| 42 | 1741 | 712 |
| 43 | 876 | 220 |
| 44 | 2753 | 283 |
| 45 | 2609 | 1286 |
| 46 | 3941 | 258 |
| 47 | 3613 | 523 |
| 48 | 1754 | 559 |
| 49 | 2916 | 1724 |
| 50 | 2445 | 1820 |
| 51 | 3825 | 1101 |
| 52 | 2779 | 435 |
| 53 | 201 | 693 |
| 54 | 2502 | 1274 |
| 55 | 765 | 833 |
| 56 | 3105 | 1823 |
| 57 | 1937 | 1400 |
| 58 | 3364 | 1498 |
| 59 | 3702 | 1624 |
| 60 | 2164 | 1874 |
| 61 | 3019 | 189 |
| 62 | 3098 | 1594 |
| 63 | 3239 | 1376 |

```
      64    3359    1693
      65    2081    1011
      66    1398    1100
      67     618    1953
      68    1878      59
      69    3803     886
      70     397    1217
      71    3035     152
      72    2502     146
      73    3230     380
      74    3479    1023
      75     958    1670
      76    3423    1241
      77      78    1066
      78      96     691
      79    3431      78
      80    2053    1461
      81    3048       1
      82     571    1711
      83    3393     782
      84    2835    1472
      85     144    1185
      86     923     108
      87     989    1997
      88    3061    1211
      89    2977      39
      90    1668     658
      91     878     715
      92     678    1599
      93    1086     868
      94     640     110
      95    3551    1673
      96     106    1267
      97    2243    1332
      98    3796    1401
      99    2643    1320
     100      48     267
EOF
```

# VITA

Jun Zhang was born in Jiujiang, Jiangxi, China, on January $3^{rd}$, 1972. He graduated from the Physics department at Nanchang University, China in 1994. He worked as an assistant engineer in China Petroleum & Chemical Corporation, Jiujiang Branch, China from 1994 to 2004. In 2007, he got his master's degree in Management Science and Engineering at Nanchang University, China. In the same year, he came to the United States to study for his Ph.D. in Engineering Management at Old Dominion University. He received his Ph.D. in Engineering Management at Old Dominion University on May, 2011.