A GENETIC ALGORITHM FOR TSP WITH BACKHAULS BASED ON CONVENTIONAL HEURISTICS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

İLTER ÖNDER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
INFORMATION SYSTEMS

SEPTEMBER 2007

Approval of the Graduate School of Informatics.

Prof. Dr. Nazife Baykal
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Yasemin Yardımcı
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science/Doctor of Philosophy.

Assoc. Prof. Dr. Haldun Süral
Co-Superivsor

Prof. Dr. Nur Evin Özdemirel
Supervisor

Examination Date : _____

Examining Committee Members (first name belongs to the chairperson of the jury and the second name belongs to supervisor)

Assoc. Prof. Dr. Yasemin Yardımcı	(METU, IS)	_____
Prof. Dr. Nur Evin Özdemirel	(METU, IE)	_____
Prof. Dr. Levent Kandiller	(Çankaya Univ., IE)	_____
Assoc. Prof. Dr. Haldun Süral	(METU, IE)	_____
Dr. Tuğba Temizel Taşkaya	(METU, IS)	_____

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this wok.

Name, Last Name : İlter Önder

Signature :

ABSTRACT

A GENETIC ALGORITHM FOR TSP WITH BACKHAULS BASED ON CONVENTIONAL HEURISTICS

Önder, İlter
M.S.c., Department of Information Systems
Supervisor : Prof. Dr. Nur Evin Özdemirel
Co-supervisor : Assoc. Prof. Dr. Haldun Süral

September 2007, 107 pages

A genetic algorithm using conventional heuristics as operators is considered in this study for the traveling salesman problem with backhauls (TSPB). Properties of a crossover operator (Nearest Neighbor Crossover, NNX) based on the nearest neighbor heuristic and the idea of using more than two parents are investigated in a series of experiments. Different parent selection and replacement strategies and generation of multiple children are tried as well. Conventional improvement heuristics are also used as mutation operators. It has been observed that 2-edge exchange and node insertion heuristics work well with NNX using only two parents. The best settings among different alternatives experimented are applied on traveling salesman problem with backhauls (TSPB). TSPB is a problem in which there are

two groups of customers. The aim is to minimize the distance traveled visiting all the cities, where the second group can be visited only after all cities in the first group are already visited. The approach we propose shows very good performance on randomly generated TSPB instances.

Keywords: Genetic Algorithms, Crossover operator, Mutation Operator, TSP with Backhauls, Conventional Heuristics

ÖZ

DAĞITIM VE TOPLAMALI GÜZERGÂHI BULMA PROBLEMİ İÇİN BİLİNEN SEZGİSELLERE DAYALI BİR GENETİK ALGORİTHMA

Önder, İlter

Yüksek Lisans, Bilişim Sistemleri Bölümü

Tez Yöneticisi : Prof. Dr. Nur Evin Özdemirel

Tez Ortak Yöneticisi : Doç Dr. Haldun Süral

Eylül 2007, 107 sayfa

Bu çalışmada toplamalı gezgin satıcı problemi için bilinen sezgisel yöntemleri operatör olarak kullanan bir genetik algoritma incelenmiştir. En yakın komşu sezgiseline dayalı bir çaprazlama yönteminin (En yakın komşu çaprazlaması, EYKÇ) özellikleri ve ikiden fazla ebeveyn kullanılması bir dizi deneyle incelenmiştir. Farklı ebeveyn seçilimi ve birden fazla çocuk yaratma stratejileri de kıyaslanmıştır. Bilinen sezgisel yöntemler mutasyon operatörü olarak kullanılmıştır. 2-kenar değişimi ve düğüm sokma yöntemlerinin EYKÇ ile iyi sonuçlar verdiği gözlemlenmiştir. Farklı alternatifler arasında en iyi sonuçları veren alternatifler Dağıtım ve Toplama Güzergâhı Bulma Problemine uygulanmıştır. DTGBP içinde iki grup şehir bulunan bir problemdir. Amacı, ikinci gruptakiler ancak birinci gruptakilerin tamamı gezildikten sonra gezilebilir şartını sağlayacak şekilde, tüm şehirleri gezen en kısa yolu bulmaktır. Kullandığımız yöntem rasgele üretilmiş DTGBP'de etkileyici sonuçlar vermiştir.

Anahtar Kelimeler: Genetik Algoritmalar, Çaprazlama Yöntemleri, Dağıtım ve Toplama Güzergâhı Bulma Problemi (DTGBP), Sezgisel Yöntemler

To all members of my loving family
and
caring extended family

ACKNOWLEDGMENTS

Firstly, I would like to express my gratitude to Prof. Dr. Nur Evin Özdemirel for her support, patient guidance, and understanding. I would like to thank to Assoc. Prof. Dr. Haldun Süral for his guidance and enthusiasm.

I would like to thank my chairman Prof. Dr. Levent Kandiller, for his tolerance and support for my graduate studies, and valuable comments in this thesis.

I have dedicated this work to all members of my extended family, who have always been there for me. This page is not enough to express all my thanks to them. I present my special thanks my dearest mom Atike Önder, my dearest dad Bünyamin Önder. I would thank my dearest sister Emel Önder for making my life easier, for her love, understanding, support, and care; she is the best roommate that any one can have. I would like to thank my brother İsmail Özkan for his support and encouraging curiosity.

I would also thank my colleagues and friends Engin Topan, and İpek Seyran Topan for their support, help, encouragement, guidance, and everything they have done to keep me focused on this work. I would like to thank my colleagues and friends Miray Hanım Arslan, Ender Yıldırım and Serdar Soysal for their support and help.

I would like to thank all my friends, especially Ahmet Göktaş and Rafet Ilğın for their wonderful company. Special thanks to Aytunç Göy and Aslı Erdoğ, for their existence, understanding, support, and care.

I am grateful to Çankaya University for making the Simulation and Modelling Laboratory available for my studies, and good working environment provided.

I would like to thank my friends Ömer Ünal, Özgül Sökmen, my students Kubilay Volkan Kaygısız and Kıvanç Uçar and everyone I could not mention here for sharing their CPU's with me. I would like to thank İrfan Nuri Karaca and Koray Kadığolu for their answers and help in all my problems.

Thanks to everyone whoever I have shared my time, and who have helped me become who I am now.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

In today's world of globalization, logistics has become one of the areas on which we have to focus on for achieving better living standards. Nearly all the products we use in our daily life are brought to our homes from far distances, and the transportation / logistics costs constitute an important portion of the costs of goods we purchase. The efficient use of transportation systems can decrease the cost of movement and improve the delivery timing of the goods to be transferred. In this study we focus on a variant of the single vehicle routing problem with pick up and delivery, where the pick up operations can be accomplished only after the deliveries are finished. The problem arises when side loading is not possible and there are goods to be picked up after delivery. There are cases reported in bottled goods or grocery industries, where empty bottles must be collected back, after full ones are delivered.

This study is limited to the case where only one vehicle or carrier exists in the planning region. It is realistic since in regional distribution environments the service region of each vehicle can be decided prior to the routing. Then, the decision is to select the best possible route to be traversed in order to deliver linehauls and then collect backhauls (pickups). The problem with a single vehicle dealing with backhauls is called Traveling Salesman Problem with Backhauls (TSPB). This problem is a constrained version of the well-known Traveling Salesman Problem (TSP).

TSP forms a general class of problems where a salesman has a list of cities to be visited exactly once and the salesman completes the tour back home where it has started. TSP is

an NP-hard problem (Sipser, 1997). The number of possible tours for a problem with n cities and symmetric distances is $\frac{(n-2)!}{2}$ (Reinelt, 1996), when the initial city (depot) is fixed. Gathering the solution of the problem gets harder as various side constraints are added. There are numerous variants that arise from real life applications and can be formulated as TSP. For instance, TSP with pickup and delivery (TSPPD) arises from logistics of brewery or bottled product industries, where linehauls and backhauls are served in a mixed order. TSPB is the precedence constrained case of TSPPD. TSP with time-windows, another example, is based on applications with time limitations, like collection or delivery of personnel according to a schedule.

The applications of TSP are not limited to transportation of goods or people. The drilling problem of PCB, deciding of positioning for x-ray crystallography (Michalewicz and Fogel, 2000), the sequencing problem on a single machine with order dependent set-up times (Lenstra and Rinnooy Kan, 1975), the frequency assignment problem in communication networks (Punnen, 2002) can all be formulated and solved as TSP instances.

TSP is an easy to formulate but hard to solve problem. A large amount of time is required to solve even a moderate sized TSP (Michalewicz and Fogel, 2000). The operations research society has been working on producing good enough solutions in reasonable amount of time since TSP was first solved by Dantzig, Fulkerson and Johnson (1954). TSP heuristic methods that use problem specific knowledge have been in use since 1965 (Lin, 1965). These heuristics are grouped in two broad categories: construction heuristics trying to construct a tour from stratch, and improvement heuristics trying to improve a given tour. Sönmez (2003) states that the heuristics are capable of generating tours roughly 10-15% longer than the optimal tour.

Metaheuristic approaches based on natural improvement mechanisms are proposed for solving difficult problems in the recent years. Genetic Algorithms (GAs) (Holland, 1975), which are specialized cases of Evolutionary Algorithms, are used to solve TSPs in this study. GA is based on “survival of the fittest” idea of Charles Darwin and tries to improve a population of solutions by the help of a set of operators. GAs exploit the search space by generating new solutions that use solutions in the current population (selection and crossover), and explore the space by making random changes in these solutions (mutation).

Sönmez (2003) and Demir (2004) used conventional TSP heuristics in developing GAs based on the idea of Jog, Suh and Gucht (1989). The well known TSP heuristics are demonstrated to giveyield better results compared to some of the GA operators used in the literature that aim to preserve solution features.

Gendreau, Hertz, and Laporte (1996) state that "the current state of knowledge on TSPB is still unsatisfactory and more powerful algorithms must be designed" (Ghaziri and Osman, 2003). In this study, a solution to TSPB is sought by transforming TSPB to a TSP. A GA that uses conventional TSP heuristics as operators is developed to solve the resulting TSP. The conventional heuristics used in our GA implementation are nearest neighbor, 2-edge exchange, and cheapest insertion heuristics. The basic idea of these heuristics is preserved in developing the crossover and mutation operators. Detailed explanation of these conventional heuristics can be found in the comprehensive book by Reinelt (1996).

The general idea of GAs is to generate one child or two children using the edges of two parents. However, our GA allows the preservation of good edges available in more than two parents and generates multiple children. GA developed in this study is therefore experimented with using multiple parents in crossover. Different numbers of parents is used to find a point that can provide a balance between preserving good edges present in the parents and including new edges from the complete graph. Moreover, generating more than two children using the same parent combination is also experimented with in order to capture the best properties of the parents.

The rest of the thesis is organized as follows. The second chapter summarizes different solution techniques applied on TSP and TSPB. The main structure of GAs and well-known genetic operators are also summarized in Chapter 2. More specifically, a summary of the major GA applications for TSP since 1995 is provided and the results of different operator combinations are compared. Chapter 3 includes the experiments conducted to develop our operators and to calibrate GA to give the best results. The best configuration of the algorithm obtained in Chapter 3 is used to solve TSPB test instances in Chapter 4. Chapter 5 concludes the thesis with a discussion of the results and our remarks about the operators used to solve TSP and TSPB.

CHAPTER 2

REVIEW OF TSP(B) SOLUTION TECHNIQUES WITH AN EMPHESIS ON GENETIC ALGORITHMS

The objective of TSP is to find the minimum weighted Hamiltonian tour over all vertices (cities) on an undirected weighted graph G = (V, E), where V represents a finite set of vertices and E represents weighted edges connecting these vertices. This definition of symmetric TSP can be extended to a broad class of TSP variants. According to Reinelt (1996), there are many variants such as multi-salesman problem, shortest Hamiltonian path problem, rural postman problem, prize collecting TSP and generalized TSP. Some of them impose side constraints on TSP. In TSP with pickup and delivery (TSPPD), there are two different types of cities, called pickup and delivery cities, to be visited in an order such that capacity constraints are not exceeded. A more strictly constrained version of TSPPD is TSP with Backhauls (TSPB), in which the pickup cities cannot be visited before all the delivery cities are visited. More details about TSPB are given in the following section.

Sönmez (2003) presents a detailed history of TSPs. Sönmez (2003) mentions that TSP in modern sense "was introduced by RAND corporation in 1948 and then the problem became popular and well-known in operations research. In 1954, Dantzig Fulkerson and Johnson solved a symmetric TSP instance of 49 United States cities" Today, problems of size with 85,900 cities are solved to optimality (Reinelt, 2007). According to Reinelt (1996), the progress in the ability to solve the problems with large sizes "is only partly due to the increase in hardware power of computers. Above all, it was made possible by

the development of mathematical theory (in particular combinatorics) and of efficient algorithms". However, TSP cannot be considered as an easy to solve problem, as the complexity of the problem increases exponentially with the number of cities. TSP is a member of NP-hard problems. Therefore, efficient and effective solution procedures are required for the solution of practical TSPs. Among these procedures CONCORDE (Cook, 2007) is a powerful tool for generating exact solutions for small and medium sized problems and good lower bounds for larger problems including up to 1,904,711 cities (Applegate, 2007). More details of the solution approaches are presented in the following sections.

TSP has a wide area of applications; Reinelt (1996) summarizes the following application of TSP:

- Drilling of printed circuit boards: The cities are initial position of the drill and set of holes to be drilled, and the distance corresponds to the time to move of the head from position to position.
- X-Ray Crystallography: The cities correspond to the different positions of a diffractometer that is used for crystallography, and the distances are the positioning times between these positions.
- Gas turbine engines: The positioning of different gas valves in a turbine in the best possible way is modeled as TSP.
- Order-Picking Problem: The collecting and shipping of orders in a warehouse is modeled as TSP.
- Computer Wiring: Location of modules on a computer board is modeled as TSP.
- Clustering Data Arrays: The task of identifying highly related elements in data is modeled as TSP.
- Seriation in Archeology: The classification of gravesites according to the distance in between to find the chronological order is solved as TSP.
- Vehicle Routing: The route each vehicle will follow is solved as TSP when the cluster first- route second approach is followed.
- Scheduling: Sequence of jobs with sequence dependent setups in a single machine is modeled as TSP.

- Mask Plotting in PCB Production: The moving of a mechanical plotting device on photosensitive plates is modeled as TSP.
- Control of Robots: The control of a robot cannot be formulated exactly as TSP yet the solution method applied for TSP gives good solution in robot control.

TSP with Backhauls (TSPB)

TSPB mainly arises from three different application areas. As mentioned before it is a strictly constrained version of TSPPD. On the other hand, TSPB is a special case of the vehicle routing problem with backhauls. Moreover, TSPB is also formulated as a special case of the Clustered TSP (CTSP) (Gendreau et al., 1996). In CTSP, the cities to be visited are partitioned into clusters and all the clusters are to be visited contiguously (Chrisman, 1975). In this sense, TSPB is a three-cluster version of CTSP, one cluster containing only the depot, and others containing the linehaul and backhaul customers, separately.

Chrisman (1975) has solved CTSP by transforming the problem into a TSP, subtracting large numbers from the inter-cluster distances. TSP is then solved without changing the intra-cluster distances. Chrisman (1975) reports that the problems with modified distance matrixes are solved to optimality without exceptions.

Gendreau et al. (1996) used GENI-US heuristic proposed by Gendreau et al. (1992) to solve TSPB. The heuristic basically consists of two parts. GENI (Generalized Insertion) tries to insert the cities to the positions by evaluating elimination of three new edges for each neighbor, within a p-neighborhood on a given tour; US tries to improve the tour by using reverse GENI operations. Gendreau et al. (1996) have experimented with six different GENI-US variants to solve TSPB. H1 is the GENIUS with the modified cost matrix where large numbers (instead of subtraction) are added to the inter-cluster distances. H2 is the GENIUS that first solves the linehaul and backhaul tours separately and then connects these subtours. H3 is similar to H2, yet the depot is not included in calculations. H4 is basically the cheapest insertion heuristic plus US for post optimization, and H5 is a GENI with Or-opt improvement heuristic. H6 is the cheapest insertion heuristic plus Or-opt improvement heuristic. Gendreau et al. (1996) reported that the best results were found by H1.

Gendreau et al. (1997) prove that the worst-case performance ratio of 3/2 of the Christofides algorithm is applicable to TSPB.

Mladenović and Hansen (1997) have improved the performance of GENIUS for TSPB, incorporating the variable neighborhood search (VNS). VNS is a random search mechanism in which an incremental length of neighborhood is processed until an improving move has been found. Ghaziri and Osman (2003) report that GENIUS combined with VNS is better than the original GENIUS by an average of 0.4% with an increase of 30% in running time.

Ghaziri and Osman (2003) is the first study to develop TSPB solution techniques that are not based on the conventional heuristics. They use an artificial neural network to solve TSPB and demonstrate that 2-opt can improve the performance of the artificial neural network. Ghaziri and Osman (2003) report better results compared to GENIUS + VNS on a set of randomly generated test problems.

Demir (2004) is the first to solve TSPB using EAs. EA developed by Demir (2004) is based on the nearest neighbor heuristic. He reports that the best results were obtained when infeasible tours were repaired after generation, instead of rejecting infeasible tours or constructing only feasible tours.

The reported studies on TSPB are limited to the ones mentioned in this section, there are no well-known benchmark problems, and each author generated the problems randomly by a method proposed by Gendreau et al. (1992). The solution quality is measured in terms of relative quality be comparing averages present in the literature.

The rest of this chapter concentrates on the solution of TSP as we solve TSPB by converting it to a TSP based on the method proposed by Chrisman (1975) and later improved by Gendreau et al. (1996). Gendreau et al. (1997) state that TSP that represents TSPB preserves the symmetry and the triangle inequality.

2.1 Solution Methods for TSP

The solution methods for TSP can be grouped in three categories. The exact solution methods aim to find a provably optimal solution to a TSP instance on hand. These techniques are commonly based on implicit enumeration of solutions and therefore require large amount of computation times. The NP-hardness of TSP forced operations research scholars to develop new ideas to find not the optimal but good enough solutions in a short enough time. Heuristic methods are the general name for such faster methods for solving the problems. The last category is the meta-heuristics, which are

optimization methods mostly based on the natural processes to solve complex problems in various domains.

2.1.1 Exact Methods

The exact methods are usually associated with the mathematical formulations (mainly integer programming formulations of the problem). The methods are not very effective in solving very large problems with single processor PCs, yet they are very useful in calculating lower bounds for TSPs. The lower bounds are useful in assessing the quality of solutions for the problems without known optimal solutions.

Sönmez (2003) reports the most widely used mathematical model to be the Dantzig-Fulkerson-Johnson formulation, using zero-one binary variables to represent the edges in the tour. The formulation has n(n-1) binary variables and 2^n - 2n - 2 constraints for an instance with n cities. The Miller-Tucker-Zemlin formulation further improves the formulation with additional continuous variables limiting the number of sub-tour elimination constraints to n^2. The power of the integer programming is limited, as the number of decision variables and constraints becomes very large with an increase in the number of cities.

Branch and bound is perhaps the most popular approach in solving these models / formulations. The previous and following nodes of a starting node of a TSP tour over a network are represented as branches in a search tree. The branching is limited when there are infeasible tours or the higher lower bounds of bad tours are reached. The success of the branching is dependent on branching rules and lower bounds, and only small instances of TSP can be solved using standard branch and bounds (Sönmez, 2003).

According to Michalewicz and Fogel (2000), dynamic programming can also be used to solve TSPs. Dynamic programming "is a recursive procedure, in that each next intermediate point is a function of the point already visited" (Michalewicz, Fogel 2000). It is impractical with the current computing technology to solve more than 50 city TSP instances to optimality using dynamic programming.

Another exact solution method is the A* Algorithm that resembles the branch and bound. Instead of branching all possible nodes, A* uses a heuristic to calculate the possible length of the un-branched nodes, and "tries to order the available cities to be

visited according to the value of the heuristic." The cities that offer best chance of finding a good solution are selected first for branching (Michalewicz, Fogel 2000). The results are similar to the results of branch and bound, yet the A* is capable of generating good intermediate solutions if the heuristic function used can capture the characteristic of the real objective function.

2.1.2 Heuristic Methods

The heuristic methods used to solve TSP are grouped in two categories: construction heuristics and the improvement heuristics. The heuristics described in this section are based on the comprehensive book by Reinelt (1996). The construction heuristics form a tour gradually, starting from a city and adding cities to a partial tour constructed. The improvement heuristics try to improve a given tour by making changes on the tour.

Construction Heuristics

Nearest Neighbor (NN) heuristic is the simplest construction heuristic. The city nearest to the current city is selected to be added to the tour. There are different versions of NN, and the best variant gives an average deviation of 21.5% (Reinelt, 1996). Another important construction heuristic is the Insertion Heuristic (IH). IH inserts nodes to a partial tour according to some predefined criterion. A popular version is the cheapest insertion, where the node whose insertion causes the lowest increase in tour length is inserted to the partial tour. The best configuration of IH results in 17.2% deviation on average (Reinelt, 1996). Christofides develops a construction heuristic based on the spanning tree and reports an average deviation of 19.5% (Reinelt, 1996). The savings heuristic proposed by Clarke and Wright merges the subtours considering the savings in tour length. All the subtours are finally merged based on the savings to give a complete TSP tour. The savings heuristic has an average deviation of 11.1% from the optimal (Reinelt, 1996). Reinelt concludes that the savings heuristic usually gives the best results among all construction heuristics (1996).

Improvement Heuristics

Node and edge insertions are commonly used as improvement operators. A node or an edge is removed from the tour and is inserted in a point that reduces the total tour length. There are different criteria to choose the node or edge to insert, leading to various versions of insertion heuristic. A second class of improvement heuristics is k-

opt (or edge exchange). In 2-opt, the two edges are deleted and the tour is reconnected the other way around. 3-opt eliminates at most three edges and reconnects the subtours to improve the tour length. Lin-Kernighan proposes a recursive search for k-opt moves.

The node and edge insertions heuristics result in 16.6% and 17.4% deviations respectively when used with the NN. The best results with node end edge insertions are obtained as 8.2% and 9.7%, respectively, when the initial tours are found by the savings heuristic (Reinelt, 1996). A recent implementation of the Lin-Kernighan heuristic report an average deviation of 1.4% with slight modifications on the algorithm (Gamboa, Rego and Glover, 2006).

2.1.3 Metaheuristic Methods

Metaheuristics are optimization methods trying to mimic the natural improvement mechanisms. "A metaheuristic is an iterative generation process which guides a subordinate heuristic by combining intelligently different concepts for exploring and exploiting the search space, and various learning strategies are used to structure information in order to find efficiently near optimal solutions" (Osman and Laporte, 1996). Metaheuristics are not problem dependent and they can be applied to various problem domains by changing the subordinate heuristic.

Simulated annealing (SA) proposed by Metropolis, Rosenbluth, Rosenbloth, Teller and Teller (1953), is a metaheuristic based on statistical physics. The annealing process in physics seeks a good molecular structure by allowing formation of different molecular structures depending on the rate of change in the cooling temperature. SA is a search process controlled by a parameter, the temperature (Kirkpatrick, Gelatt Jr. and Vecchi, 1983 and Ćerny, 1985). The process is based on small changes in the current solution and the good moves are always accepted. When a move in an undesirable direction is encountered, the move is still accepted based on a probability depending on the temperature. At the initial phases of the algorithm, when the temperature is high, the algorithm accepts more non-improving moves. At the final stages when the temperature is gradually decreased, only improving moves are accepted. The algorithm explores the search space when the temperature is high, and exploits the current solution when the temperature is low. Sönmez (2003) reports results that are 4% above the optimal when SA is used for TSP.

According to Larrañaga, Kuijipers, Murga, Inza, Dizdarevic (1999), Evolutionary Algorithms (EA) were proposed for solving probabilistic search problems by Bremermann et al. (1965) and Rechenberg (1975). Holland (1975) introduced the Genetic Algorithms (GAs) to optimization problems. GAs are based on "survival of the fittest" idea of Charles Darwin and genetic theory of Mendel. In GAs every solution is coded as a chromosome and the algorithm deals with a population of solutions instead of a single solution. These parent solutions are used to generate new children solutions that preserve chromosomes of previous ones. According to the schemata theorem (Holland, 1975) and building block hypothesis (Goldberg, 1989), the newly generated solutions preserve good characteristics of their ancestors, and the algorithm eventually converges to give good results. The computation time is not longer than the time required to solve the problems to optimality. Larrañaga at al. (1999) report 28 different studies that deal with developing good GA operators for TSP. Section 2.2 describes in detail GAs tailored for TSP.

Tabu Search is a deterministic search mechanism (with a limited memory) proposed by Glover (1986). The search is based on a hill climbing mechanism where memory is used to escape from the local optima. Hill climbing eventually gets stuck at the local optima. Tabu search keeps in memory the points previously visited during the hill climbing mechanism as tabu points. Revisiting the tabu points is avoided to enforce the algorithm to explore the search space. Sönmez (2003) mentions examples of tabu search for TSP with solution quality of 3% above the lower bound.

Artificial neural networks are based on neural activity model of Warren McCulloch and Walter Pitts, mimicking central neural networks of animals (Michalewicz, Fogel, 2000). The web of natural neurons does the reasoning in all animals. A neuron is a simple entity that gets inputs as a step function and reacts accordingly, multiplying the input signal and adding a preset weight. The web of artificial neurons arranged in two layers is used to solve TSP according to Michalewicz and Fogel (2000). The coordinates of cities in TSP are input to the neural network and the network produces the tour with unsupervised learning. Michalewicz and Fogel (2000) state that "many neural network methods for addressing TSP are not very competitive with other heuristics."

Another well known metaheuristic is the ant colony algorithm proposed by Colorni, Dorigo and Maniezo in 1991 (Gendreau et al, 2001). This algorithm represents the ant behavior to find the shortest route. When solving a TSP, a number of artificial ants

move on a complete graph to find a route disposing pheromone, similar to the real ants. The pheromone evaporates with time and the routes with highest pheromone levels are connected to give a tour. The main idea behind the scheme is that the short edges will have a higher level of pheromone as the ants will travel those edges in a shorter time. The approach is relatively new and lacks well-established rules. The applications are not very competitive with other heuristics in terms of the ability to solve large problems.

2.2 Genetic Algorithms for TSP

Holland (1975) was the first to introduce the genetic algorithms. The genetic algorithm is a search methodology based on biological phenomenon of evolution. The algorithm starts with a group of solutions, named as population of individuals. These solutions represent different points in the search space. The solution an individual represents is encoded using a representation scheme. The encoded solution is named as the genotype, and the actual solution to which a genotype corresponds to is named as the phenotype of that individual.

There is no single representation for TSP that keeps all the information about the edges in a solution, and can be used with any crossover operator. There are alternative representation schemes used. The most common representation is the path representation. The tour is represented with a string of the numbers assigned to cities in the order of visit.

The search in the genetic algorithm is done by two means. Individuals of the current population (parents) are used to generate new individuals (children) that preserve the genotype and/or phenotype of the individuals in the current population, combining good properties of different individuals. This process is called the reproduction, which consists of selection and crossover. The second search method is based on small perturbations in the current solution to find points in the neighborhood of the current solutions with a better value of the objective. The algorithms are designed to converge to a fitness value by replacing the better individuals with worse ones. A sample genetic algorithm adapted from Larrañaga (1999) can be seen in Figure 2.1.

Initial populations can be generated randomly or using some construction heuristics. Authors such as Tsai, Yang, Tsai and Kao (2004a, 2004b), Maekawa, Mori, Tamaki, Kita and Nishikawa (1996), Baraglia, Hidalgo and Perego. (2001), and Nagata and Kobayashi (1997) report impressive results when the initial population is generated

randomly. On the other hand, Mertz and Freisleben (1997), Yang (1997), Tsai et al. (2003), and Freisleben and Mertz (1996) are some of the authors who came up with very good solutions when the initial population is generated using a heuristic. Sönmez (2003) experimented with different initial population settings, where the entire or a portion of initial population is generated using conventional TSP heuristics. Sönmez (2003) concludes that the use of heuristics does not always improve the solution quality of GA, and the structure of the initial population has an effect depending on the crossover operator used.

```
Begin GA
        Generate initial population
        WHILE NOT stop DO
                BEGIN
                        Select parents from population
                        Produce children from selected parents
                        Mutate the children
                        Extend the population adding children to it
                        Reduce the extended population
                END
        Output the best individual found
END
```

Figure 2.1 Pseudo code for a simple genetic algorithm

The first step of the algorithm is parent selection. Beasley, Bull and Martin (1993) state that "the behavior of GA very much depends on how individuals are chosen to go into the mating pool." There are two different approaches when the mating pool is considered. The whole population can be used as the mating pool. This approach is named as the generational GA. In the second approach, only a pair (or a portion) of individuals are selected as parents and they are used to generate children. In the extreme case, the size of the mating pool is two, and this approach is named as the steady-state reproduction. Both approaches are widely used in the literature. For instance, Tsai et al. (2003) and Burkowski (2003) report good results using the generational GA, whereas authors like Chen and Smith (1999) and Takenaka and Funabiki (1998) report good results using steady state GA. Although, Goldberg and Deb (1991) found no evidence that steady-state approach is superior to the generational approach, Demir (2004) reports that a steady state GA can give better results than the generational GA with some crossover operators.

In both mating pool approaches, the individuals that have a better fitness value need to be able to transfer the information encoded in their genotypes to the future generations. Thus, the fitness or the ability to meet the pre-specified objective of an individual needs to be assigned to differentiate between individuals that perform well and poorly. The fitness in GAs for TSP is usually the length of the tour. The selection mechanism is generally designed to favor the highly fit individuals, based on the biological phenomenon that the individuals with better phenotype have a higher chance of survival and reproduction. The selection in the generational reproduction is designed to decide on the number of copies of an individual in the mating pool. The fitness value or adjusted fitness values are used to apply selection pressure in choosing individuals for reproduction in generational GA. According to Beasley at al. (1993), Grenfenstettes's GENESIS is an example where adjusted fitness is used for selection. Another selection method for the generational GA is the tournament selection, where a couple of individuals are compared with each other and the one that has better fitness value is chosen for reproduction. The tournament selection can be modified to make the selection probabilistic. Goldberg and Deb (1990) conclude that no selection pressure is absolute best, and the selection schemes can be made to give similar performances.

When steady-state reproduction is implemented, the selection is simply used to select the parents that generate the new children. The parents can be selected at random, according to their fitness values or according to their rank in the population. Nagata and Kobayashi (1997) and Katayama, Sakamoto and Narihisa (2000) have used random selection, and Julstrom (1995) and Nguyen, Yoshihara and Yasunaga (2000) are two examples where the selection based on ranking gives good results. The selection of the individual with best fitness can cause the algorithm to converge in a short time to local optima, spreading properties of relatively good individual throughout the population. The algorithm will eventually converge to the point where the relatively good individual is located, instead of the global optimal. Random selection slows the algorithm to a degree, as non-promising individual will also be used for reproduction. Assigning selection probabilities based on the raking used by Whitley's (1989) GENITOR can be used to select parents effectively.

The children are generated by crossover operators and modified by mutation operators. A crossover basically tries to preserve good characteristics of parents, while a mutation operator tries to find different solutions with small perturbations on a given individual.

The crossover operator in general tries to improve the exploitation characteristic of GA. Two or more individuals are taken into consideration and an intermediate solution that preserves good characteristics of these individuals is exploited. The mutation operator tries to explore the search space with the help of slight changes in newly generated children. Crossover and mutation operators for TSP are described in detail in Section 2.3.

The newly generated children are added to the population on hand, and then some individuals in the extended population are deleted to keep the population size constant. The biological population behavior analysis states that the population in a given habitat is limited, and keeping the population size constant is based on this fact.

2.3 Crossover and Mutation Operators for TSP

This section briefly reviews various different crossover and mutation operators that have been used in the literature. The crossover and mutation operators are named and classified according to the review paper of Larrañaga et al. (1999) and the study conducted by Sönmez (2003). For detailed explanations and historical references of the well-known operators, the reader may refer to these references and the references therein for the operators that are not mentioned in Larrañaga et al. (1999) and Sönmez (2003). More detailed information is given here only.

The crossover operators are grouped in two categories. The first category includes the ones that aim to preserve the position or the order of cities in the solution. Unfortunately, there is no single representation for TSP that encapsulates all the edge information of the individuals and that can preserve edges when a simple crossover operator is applied. The second category is the crossover operators that preserve the edges in solutions. Mutation operators are presented in rough groups to include similar operators used by different authors.

2.3.1 Crossover Operators Preserving Position or Order of Cities

The crossover operators described in this section aim to preserve sub-strings or relative order of the values in the genotypes of an individual. The edge information is usually not considered during the crossover procedure. The success of the operators mentioned here is limited when compared to the crossover operators that use edge information.

Partially Mapped Crossover (PMX) preserves part of a string from one parent and the relative order of visits of the other parent. This is done by randomly selecting a substring from the first parent and filling the remaining cities according to a mapping created based on their absolute positions on the parents. The order in the second parent is preserved according to the mapping. Cycle Crossover (CX) keeps the absolute position of the cities visited in the order they are presented in the parents, selecting cities in cycle from the two parents in a cyclic manner. In Position Based Crossover (POX), absolute positions of randomly selected cities of one parent are inherited to the child, where the remaining cities are inserted in the absolute order they appear on the other parent. Alternating Position Crossover (APX) selects cities one by one from each parent and places them in the child in the order they appear, keeping the relative order unchanged, but loosing a great deal of edges. Alternating Edge Crossover (AEX) selects every other edge from the parents and inserts them in the child similar to the APX. Sub-tour Chunks Crossover (SCX) preserves subtours of random length from parents; a subtour from a parent is followed by a subtour from the second parent. Order Preserving Crossover (OX1) is similar to the PMX, where a substring from the first parent is copied to the child, and the remaining cities are positioned according to their relative position in the other parent. Starkweather, McDaniel, Whitley, Mathias and Whitley (1991) showed that OX1 performs better than PMX, and PMX performs better than CX. Maximal Preservative Crossover (MPX) and Order Based Crossover (OX2) use the same basic idea of OX1, with a slight modification in copying of the string from the first parent and filling the remaining cities in the order they appear. Rocha, Vilela and Neves (2000) proposes a new crossover UOPX (uniform OX), and demonstrates (using eil51) that UOPX gives better results compared to OX1, OX2, PMX, CX, ERX, MPX (to be discussed later) (and modified MPX) when used with some of the mutations (to be discussed later).

Wang, Maciejewski, Siegel and Roychowdhury (2006) use the gene therapy to improve the performance of PMX with mutation operators. The good edges (eugenic genes) are inserted to the places of bad edges (morbid genes) for the individuals generated. The eugenic and morbid genes are found investigating the superior and poor individuals. The method improves the performance of the crossover and mutation operators.

EMX or (Inver-over operator in Michalewicz, Fogel (2000)) is a crossover based on only one parent, the cities on an individual are exchange in an iterative manner as long

as there is an improvement in the tour length. The crossover is based on only one individual thus can be named as a mutation operator.

Complete Edge-exchange Crossover (CSX), proposed by Katayama et al. (2000), preserves the substrings containing the same cities regardless of the order of the cities. All possible combinations of the substrings are listed and the remaining cities are preserved in their relative order. The good children are selected using stochastic hill climbing method.

Voting Edge Recombination Crossover (VEX) is proposed by Mühlenbein (1989). VEX selects the edge positions that will be inherited to the children by voting. More than two parents are selected and the position that is most popular among these parents is preserved for a city. Larrañaga et al. (1999) point out that VEX is used in an evolutionary algorithm for the quadratic assignment problem.

Črepinšek, Mernik and Žummer (2000) suggest a meta-evolutionary approach where more than one crossover operator can be used in the GA. The authors conclude that using PMX, OX, CX and ERX (to be discussed later) together gives results better than any single operator. Affenzeller (2002) suggest using different crossover operators together while selective pressure is also adapting itself when the algorithm proceeds.

Scmitt and Amini (1998) conduct a very detailed experiment with statistical analysis (of approximately 5,000 TSP solutions) for OX1, OX2, POX, CX, PMX, and SM (to be discussed later) as crossover operators. The authors also investigate the effects of initial population, population size and replacement strategy. They conclude that a configuration containing a hybrid population at the initialization (i.e. 50% of the population is generated with a construction heuristic), a large population (over 200) in size, a steady state evolution strategy, elitists replacement strategy, and SM or OX1 is the best configuration. The authors also suggest the use of small population (fewer than sixties) and CX, all other characteristic remaining the same.

The idea of preserving the position or the order of cities is not very promising. According to the results of Larrañaga et al. (1999), Edge Recombination Crossover (ERX) that will be discussed in the following section, performs better than APX, CX, OX1, OX2, PMX, POX, and VEX. Moreover, none of these crossover operators is faster than ERX according to the results of the same study.

2.3.2 Crossover Operators Preserving Edges

The operators described in this section use the edge information and try to preserve the good edges in the parents. The operators preserving edge information are seen after 1987 when Suh and Van Gucht (1987) first used edge information in the Heuristic Crossover, according to Sönmez (2003). Heuristic Crossover (HX) is based on selecting one of the four edges, adjacent to a city in two parents according to their lengths. A probability distribution is defined according to the edge lengths and the edge is selected according to this distribution. If all neighbors of a city are already visited and none of these four edges can be used, an edge is selected randomly from the complete graph. Larrañaga et al. (1999) report that 30% of the edges in the parents are preserved if the probability distribution is uniform.

ERX aims to increase the ratio of the edges preserved. This crossover selects edges based on the number of feasible neighbors that each city will hold if that city is visited next. The algorithm visits the cities with fewer feasible neighbors, in order to avoid getting stuck. If the current city has no unvisited neighbors, an unvisited edge is selected randomly, to introduce a new edge from the complete graph. The edge length is not considered at all in ERX, yet the edges in parents are tried to be preserved as long as it is possible. Sönmez (2003) reports that about 95% of the edges are transferred to the children. Whitley, Starkweather and D'Ann Fuquay (1989) demonstrate that ERX performs better than PMX, CX and OX1. Six different versions of edge recombination are developed by Nguyen et al. (2000). Nguyen et al. (2000) point out that using both ends of a partially formed string improves the performance of ERX. Moreover breaking the current sub-tour into parts to avoid getting an edge from the complete graph is reported to produce better results than pure ERX.

Ting (2004) improves the performance of ERX by incorporating tabu search into ERX, some edges becoming tabu edges and selecting edges alternating between parents.

Sorted Match Crossover (SMX) tries to find a substring that includes the same cities, starting and ending with the same city in both parents. The order within the substring in a parent with shorter substring is copied to the other parent. Larrañaga et al. (1999) reports that SMX reduces the computation time, but it is a weak scheme for crossover.

Freisleben and Mertz (1996) suggest the use of Distance Preserving Crossover (DPX), based on the observation that the two locally optimal solutions are equally distant to the

optimal solution. The crossover they design is similar to ERX, as the common edges in both parents are kept in the children, but the remaining edges are selected such that the child is equally different from both parents. Freisleben and Mertz (1996) report a deviation of 0.5% for instances as large as 3745 in size. White and Yen (2004) propose use of ant colony systems to generate the connections for the non-common edges in DPX. According to their study, the ant colony is capable of generating of good solutions that can improve the performance of DPX.

Soak and Ahn (2004) propose a new crossover operator (SPX) that preserves the subtours in parents, and calculate the alternative connection methods. Their operator chooses the best connection similar to DPX. The results presented suggest that, their operator is superior to MPX, HX, VGX, DPX, ERX, and CSX in terms of percent deviation. However, the CPU time is higher compared to DPX.

Tagawa, Kanzaki, Okada, Inoue and Haneda (1998) generalize the idea of generating the children equally distant from two parents and propose a crossover technique called harmonic crossover (H-), which uses a metric function to find the distances between individuals. H-PMX and H-CX are some of the operators proposed by the authors. The results seem to improve in terms of quality and CPU time on a problem instance of size 53.

Katayama et al. (1999) compare the performance of three different crossover operators when H-GA is used: CSX, MPX and ERX. According to the results presented, CSX gives the best results on 25 instances from TSPLIB (Reinelt, 2007). A deviation of 5.0% is observed for a problem instance of size 2392 when CSX is employed.

The crossover designed by Yang (1997) is very similar to ERX and DPX. It is called as Very Greedy Crossover (VGX). The common edges in both parents are always selected. When a common edge cannot be found, the shortest of the parental edges is selected. VGX uses a k-nearest neighbor candidate graph for selection when there is no feasible parental edge. If VGX fails to find a feasible edge in the k-nearest neighbor list, a feasible edge is selected randomly. Julstrom (1995) uses a similar crossover but not on the k-nearest neighbor candidate graph.

CST/NN proposed by Chen and Smith (1999) keeps the common edges in the parents and uses the NN heuristic to select the edges that are not common in the parents. Chen and Smith (1999) report an average deviation of 1.8% for the instances up to 574 cities.

Pullan (2003) proposes Heuristic Edge Recombination (HEX) that firstly divides the parent tours into arcs and reconnects these arcs by using edge information. The reconnection of these arcs also creates a degree of mutation, preserving the parental edges.

Edge Assembly Crossover (EAX) uses the edges present in the parents to construct AB cycles, which consist of parental edges selected alternating by between the first and the second parents. Then, these AB cycles are merged to obtain a grated set (E-set), which is applied to each parent to obtain subtours that contain edges of both parents. These sub-tours are connected calculating the minimum spanning tree. Nagata and Kobayashi (1997) report optimal solutions for problem instances with sizes up to 3038 cities. Moreover, Nagata and Kobayashi (1999) show that EAX handles the tradeoff between the number of edges inherited from parents and the newly added edges better, compared to EXX that is similar to PMX. EAX creates better children by replacing some parental edges with the minimum spanning tree.

Jung and Moon (2002) devise a NX crossover where the tours are plotted on a graph, that is partitioned randomly, and then the partitions from different parents are merged to give partial tours. These partial tours are merged using the shortest edges. They argue that the results they present are better than EAX and faster than DPX. The authors report that EAX "showed poorer performance than the original paper (Nagata, Kobayashi, 1997)". LK is used to improve the results of NX, and a deviation of 0.085% is obtained for a problem instance with 11849 cities.

Merz (2002) proposes a new edge recombination (GX) operator where the probabilities of selecting an edge inherited from the parents, and an edge to be selected from the complete graph can be adjusted. Merz (2002) shows that GX is superior to DPX and MPX. The results are comparable with the results of EAX for small problems.

Nearest Neighbor Crossover (NNX), which was proposed by Sönmez (2003) and improved by Demir (2004), is based on the NN. The edges adjacent to a city in the parents are ordered in the increasing length and the shortest feasible edge is selected. The algorithm is totally deterministic as the shortest feasible edge from the complete graph is selected when there is no feasible edge remaining in the parents.

Ray, Bandyopadhyay and Pal (2005) use an operator similar to NNX, but this operator is devised to improve the individuals. They propose fragmentation of tours generated

with NN heuristic and connecting the tours using the shortest edge to connect from the cost matrix.

2.3.3 Mutation Operators

The mutation operators are generally based on conventional improvement heuristics. This section briefly describes mutation operators designed for TSP. The classification is based on Larrañaga et al. (1999).

Displacement Mutation (DM) operator removes a substring from the individual and replaces the substring in another position in the individual. Exchange Mutation (EM) randomly selects the two cities and exchanges them. Insertion Mutation (ISM) randomly selects a city, removes it from the current individual, and places it at a random point on the individual. Inversion Mutation (IVM) randomly selects a sub-string on the individual and inverts it. Scramble Mutation (SM) randomly selects a sub-string and scrambles the order of the cities in the substring. Larrañaga et al. (1999) point out that SM is designed for use in scheduling applications.

Xiaoming, Runmin, Rong, Rui and Shao (2002) prove that GA converges to global optima when only mutation operators are applied (e.g. EM). They argue that crossovers that preserve the order or positions of cities are redundant in optimization. They demonstrate that EM can find the optimal solutions for problem instances with as large as 1002 cities. Moreover, Fox and McMahon (1991) report that PMX and ERX give better results compared to IVM and SM and other single parent operators they have devised, on a set of test instances.

Tsai et al. (2003) propose a mutation operator named as Neighbor-Join (NJ) Mutation. NJ generates four different children and best child is selected. The operator randomly selects a city, and than either selects another individual and tries to insert an edge neighboring to a current city from the other individual, or inserts an edge among the nearest three cities to the current city from the complete graph. If the current city cannot connect to the city from the other individual, the shortest city inversion is applied until one of the cities is connected.

Conventional improvement heuristics such as 2-opt, 3-opt, Or-opt and Lin-Kernighan (LK) are often used as a form of mutation (Jog et al., 1989). k-opt heuristics try to eliminate k edges and reconnect the resulting subtours by adding new k-edges to create a shorter tour. The LK iteratively tries to eliminate these edges resulting in k-opt moves

where k is decided by LK. Johnson (2004) implemented the method proposed by Johnson and McGeoch (1997) where only longest 40 edges are tried as k-opt moves. Johnson (2004) reports that 2-opt and 3-opt results in 5.9% and 4.3 % deviations respectively on TSPLIB instances with 1000 cities. The deviations become 9.3% and 3.5% when the problems size is 85900. Chained LK is reported to end with 0.96% deviation with problem instances of size 1000.

2.4 GA Applications on TSPLIB Instances

TSPLIB (Reinelt, 2007) is the main source of TSP benchmark instances that are commonly used for validation of new algorithms in the literature. TSPLIB consists of 111 problem instances with sizes varying between 14 and 85900 cities with provably optimal results. As we have demonstrated in the previous sections, the literature on GAs for TSP consists of a large number of studies.

The studies prior to 1995 are discussed in Larrañaga et al. (1995), the major work after 1995 reported in electronically available publications is consolidated in Table 2.1. The work of authors who use TSPLIB instances and their results or percentage deviations for problem with more than 50 cities are given in that Table. Table 2.1 demonstrates the year and source information for the listed work, with the crossover and mutation operators used. Note that various authors have made slight changes on the main operators presented in the Table. The sixth column in the Table gives the size of the smallest and the largest problem instances solved in a study. When an author solves only one problem from TSPLIB, only that problem size is reported. The seventh column shows the corresponding solution quality reported by the author. Table 2.1 contains the results of 36 different studies. The number of studies is limited as the authors usually concentrate on the properties of the operators they propose and demonstrate the convergence using figures instead of reporting numerical results. The results in the table can be used to evaluate the performance of the operators. Johnson (2004) states that only problems sizes with over 1000 are used to asses the quality of algorithms in the webpage of "8th DIMACS Implementation Challenge: The Traveling Salesman Problem". Details of various solution algorithms, different from GAs, can be found in the related reference.

When the studies that solve instances with more than 1000 cities are considered, DPX, EAX, MPX, CGA and HEX seem superior (with deviation less than 0.01%) to the other crossovers.

Both DPX and EAX give good results to the problem instances with sizes larger than 3000 cities. The deviation with EAX is hardly above zero for large instances (like size of 13509 cities). LK, NJ, and 3-opt give best results compared to the other mutation operators.

Table 2.2 summarizes the problem solved by each author and the percentage deviations. The last column in the Table presents remarks related to the results reported in the studies. We can say that use of different problem instances can be regarded as random choices as different scholars from different field such as electrical and electronics engineering, computer science, operations research etc., are working in the same domain. Although there are TSPLIB test problems like lin318 that was solved in 13 different studies, most problems are solved 2.2 times on average.

A more detailed version of tables is presented in Appendix A.

Table 2.1 Comparison of GAs with promising results conducted after 1995

Authors	**Year**	**Source**	**Crossover**	**Mutation**	**Problem Size**	**Deviation (%)**
Julstrom	1995	Appl. Comp.	VGX	IVM	105	0.03
Fresleben, Merz	1996	Evol. Comp.	DPX	LK	51 - 1577	0.00 - 0.46
Maekawa et al.	1996	Evol. Comp.	PXM	2-opt	51 - 575	0.00 - 0.08
Gorges-Schkeuter	1997	Evol. Comp.	MPX	LK	198 - 3795	0.00 - 0.34
Julstrom	1997	Appl. Comp.	CX	EM	200 - 318	0.61 -1.33
Merz, Freisleben	1997	Evol. Comp.	DPX	LK	198 - 3795	0.00 - 0.51
Nagata, Kobayashi	1997	ICGA	EAX	-	101 - 3038	0.00 - 0.03
Yang	1997	GA in Eng. Sys.	VGX	2-opt, 3-opt	51 - 2392	0.00 - 2.33
Chen, Smith	1999	Found. of GA	CST/NN	2-opt	198 - 574	0.87 - 2.65
Katayama et al.	1999	Sys. Comp.Jap.	CSX	SM	51 - 2392	0.23 - 5.00
Ho, Chen	2000	High Per. Comp.	OAX (NN)†	IVM	105 - 2392	5.16 - 41.69
Katayama et al.	2000	Math. And Comp.	CSX	2-opt	51 - 532	0.28 -1.99
Stütze et al.	2000	IPPS	DPX	3 -opt	1000 - 2392	0. 08 - 0.08
Baraglia et al.	2001	Evol. Comp.	CGA‡	LK	198 - 14461	0.00 - 0.01
Yang, Stancey	2001	AI, Canada	NNX	-	50 - 75	0.00 - 0.00
Jung, Moon	2002	Evol. Comp.	NX	LK	318 - 11849	0.00 - 0.06
Merz	2002	PGEC	GX	LK	318 - 1002	0.00 - 0.00
Xiaoming et al.	2002	WCICA	-	EM	130 - 2352	0.00 -0.00
Hui et al.**	2003	Evol. Comp.	EMX	-	100 - 532	0.00 - 1.18
Pullan	2003	Evol. Comp.	HEX	2 - opt	70 - 1060	0.04 - 0.50
Tsai, Tsai	2003	Eng. Opt.	EAX	NJM	101 - 3038	0.00 - 0.00
Burkowski	2004	Parallel Comp.	PMX	-	51 - 225	0.00 - 0.01
Nagata	2004	Evol. Comp.	EAX	2-opt	532 - 1173	0.04 - 0.07
Ray et al.	2004	ICPR	OX2	SM	70 - 100	1.48 - 1.04
Soak, Ahn	2004	AI and Soft Com.	SPX	IVM	51 - 195	0.70 - 2.4

Table 2.1 (Continued)

Authors	Year	Source	Crossover	Mutation	Problem Size	Deviation (%)
Tsai et al.	2004a	IEEE Cyber.	EAX	LK	318 - 15112	0.00 - 0.14
Tsai et al.	2004b	Inf. Bio-Med.	EAX	NJM	101 - 13509	0.00 - 0.00
Tsai et al.	2004c	Soft Comp.	EAX	NJM	318 - 13509	0.00 - 0.00
Wang et al.	2004	ICMLC	NGA	RIM	105 - 1000	0.01 - 0.00
White, Yen	2004	Evol. Comp.	ANT-DPX*	2 - opt	51 - 318	0.00 - 0.00
Zou et al.	2004	Evol. Comp.	EAX	NJM, LK	318 - 11889	0.00 - 0.00
Chan et al.	2005	IGEC	EAX	2 - opt	575 - 1173	0.03 - 0.02
Ray et al.	2005	LNCS	OX2	IVM	100 - 783	0.31 - 7.22
Xuan, Li,	2005	ICGC	OX1	IVM, LK	51 - 442	1.40 - 0.89
Yan et al.	2005	ICMLC	EMX	IVM	70 - 280	0.00 - 0.00
Wang, Cui, Wan, Wang	2006	Ins. Meas. Control	PMX	DM, SM	51 -318	0.00 - 0.00
Wang, Han, Li, Zhao	2006	Eng. Opt.	NGA	RIM	105 - 1000	0.00 - 0.00

* DPX uses Ant Colony optimization to introduce new edges.

† OAX (NN) uses orthogonal array representation and NN heuristic to construct the decoded tours.

‡ CGA uses probability matrixes for representation and the algorithm proceeds by updating these matrixes.

** Hui et al. (2003) chooses edges from the gene library in their implementation of GA with immunology principle.

‡‡ Wang et al. (2004) use their own crossover (NGA) and mutation operators (RIM) designed for the representation scheme based on the permutations of positions encoded.

Table 2.2 Problem instances used by authors, and percent deviations

Authors	Year	Problem Instances (Deviation %)	Remarks
Julstrom	1995	Lin105 (0.03)	
Friesleben, Merz	1996	eil51 (0.00), kroA100 (0.00), d198 (0.00), att532 (0.05), rat783 (0.04), fl1577 (0.46)	
Maekawa et al.	1996	eil51 (0.00), kroA100 (0.00), rat575 (0.08)	
Gorges-Schkeuter	1997	d198 (0.00), lin318 (0.00), pcb442 (0.27), att532 (0.03), rat783 (0.03), fl1577 (0.23), fl3795 (0.34)	
Julstrom	1997	kroA200 (0.61), pr226 (0.00), pr299 (0.44), lin318 (1.33)	best of two replications reported
Merz, Freisleben	1997	d198 (0.00), lin318 (0.00), pcb442 (0.27), att532 (0.04), rat783 (0.00), fl1577 (0.21), fl3795 (0.51)	
Nagata, Kobayashi	1997	eil101 (0.00), kroA200 (0.00), lin318 (0.01), fl417 (0.00), pr439 (0.03), pcb442(0.00), att532 (0.03), u574 (0.00), rat575 (0.00), p654 (0.00), d657 (0.00), u724 (0.01), rat783 (0.00), u1060 (0.00), vm1086 (0.01), pcb1173 (0.01), nrw1379 (0.01), u1432 (0.01), vm 1748 (0.00), pr2392 (0.01), pcb3038 (0.03)	
Yang	1997	eil51 (0.00), kroA100 (0.00), pa561 (1.34), pr1002 (2.33), pr2392 (2.33),	
Wang et al.*	1998	kroA100 (0.0), rd100 (0.15), lin101 (0.00), pr107 (0.16), pr124 (0.00),	
Chen, Smith	1999	d198 (0.87), lin318 (0.31), fl417 (1.16), pcb442 (1.22), u574 (2.68),	
Katayama et al.	1999	eil51 (0.22), pr76 (0.12), kroa100 (0.06), kroB100 (0.34), rd100 (0.33), lin105 (0.01), eil101 (0.35), pr124 (0.00), pr144 (0.00), kroA150 (0.54), kroB150 (0.41), pr152 (0.10), pr226 (0.02), pr264 (0.20), pr299 (0.69), lin318 (1.62), pr439 (1.17), att532 (2.14), u547 (3.10), rat783 (3.49), pr1002 (2.91), u1060 (3.78), pr2392 (5.00)	
Ho, Chen	2000	Lin105 (5.16), lin318 (16.51), pr349 (18.20), pr1002 (29.33), pr2392 (41.69)	
Katayama et al.	2000	eil51 (0.28), pr76 (0.00), kroa100 (0.00), rd100 (0.15), lin105 (0.00), pr107 (0.16), pr124 (0.00), pr226 (0.03), pr299 (0.64), lin318 (1.62), pr439 (0.69), att532 (1.99)	
Stütze et al.	2000	dsj100 (0.08), pr1002 (0.00), u1060 (0.03), vm1086 (0.20), pcb1173 (0.02), d1291 (0.04), rl1304 (0.00), rl1323 (0.01), nrw1379 (0.07), fl1400 (0.12), u1432 (0.45), fl1577 (0.16), d1655 (0.00), vm1748 (0.05), u1718 (14), rl1889 (0.04), d2103 (0.02), u2152 (0.18), u2319 (1.24), pr2392 (0.08)	
Baraglia et al.	2001	d198 (0.00), lin318 (0.00), pcb442 (0.00), att532 (0.00), gr666 (0.00), rat738 (0.00), pr1002 (0.00), u2152 (0.00), fl3795 (0.00), frl5915 (0.00), fnl14461 (0.01)	
Yang, Stancey	2001	eil50 (0.00), eil75 (0.00),	
Jung, Moon	2002	Lin318 (0.00), att532 (0.01), dsj1000 (0.00), d2103 (0.04), pcb3038 (0.03), fnl44461 (0.07), rl11849 (0.09),	
Merz	2002	Lin318 (0.00), pcb442 (0.00), att532 (0.00), rat783 (0.00), pr1002 (0.00)	
Xiaoming et al.	2002	Ch130 (0.00), tsp225 (0.00), pr1002 (0.00), pr2392 (0.00),	
Hui et al.	2003	kroA100 (0.00), kroB100 (0.00), lin105 (0.00), pr124 (0.00), chc144 (0.00), pr152 (0.00), rat195 (0.00), att532 (1.18)	uses good common edges to generate new individuals, (immunology)

Table 2.2 (Continued)

Authors	Year	Problem Instances (Deviation %)	Remarks
Pullan	2003	eil51 (0.00), st70 (0.00), eil76 (0.00), pr76 (0.00), rat99 (0.00), kroA100 (0.00), kroB100 (0.00), rd100 (0.00), kroc100(0.00), kroD100 (0.00), kroE100 (0.00), eil101 (0.00), pr124 (0.00), bier127 (0.00), ch130 (0.00), pr136 (0.00), pr144 (0.00), kroA150 (0.00), kroB150 (0.00), ch150 (0.00), kroB152 (0.00), pr152 (0.00), u159 (0.00), d198 (0.00), kroA200 (0.00), kroB200 (0.00), tsp225 (0.00), pr226 (0.00), gil262 (0.00), pr264 (0.00), a280 (0.00), pr299 (0.00), lin318 (0.00), pr439 (0.00), att532 (0.00), p654 (0.01), d657 (0.09), u724 (0.12), pr1002 (0.33), u1060 (0.5)	
Tsai et al.	2003	eil101 (0.00), kroA200 (0.00), lin318 (0.01), pcb442(0.00), att532 (0.00), u574 (0.00), rat575 (0.00), u724 (0.00), rat783 (0.00), vm1086 (0.02), pcb1173 (0.01), u1432 (0.00), vm1748 (0.01), pr2392 (0.00), pcb3038 (0.00)	
Burkowski	2004	eil51 (0.00), berlin52 (0.00), kroA100 (0.03), lin105 (0.00), eil101 (0.02), ch130 (0.02), ch150 (0.03), ts225 (0.00),	
Nagata	2004	att532 (0.04), rat575 (0.05), u724 (0.04), rat783 (0.05), p1060 (0.09), vm1084 (0.13), pcb1173 (0.07)	
Ray	2004	st70 (1.48), kroA100 (1.04)	best of 30 replications reported
Soak, Ahn	2004	eil51 (0.70), rat99 (1.4), kroA100 (0.8), rat195 (2.4),	
Tsai et al.	2004a	eil101 (0.00), kroA200 (0.00), lin318 (0.00), pcb442 (0.00), att532 (0.00), u574 (0.00), rat783 (0.00), vm1084 (0.00), pcb1173 (0.00), u1432 (0.00), vm1748 (0.00), fnl4461 (0.00), usa13509 (0.00),	
Tsai et al.	2004b	lin318 (0.00), pcb442 (0.00), att532 (0.00), rat783 (0.00), pr1002 (0.00), vm1086 (0.00), pcb 1173 (0.00), u1432 (0.00), vm1748 (0.00), u2151 (0.00), pr2392 (0.00), pcb3038 (0.00), frl5915 (0.00), usa13509 (0.01), fnl4461 (0.00), d15112 (0.01)	
Tsai et al.	2004c	eil101 (0.00), kroA200 (0.00), lin318 (0.00), att532 (0.00), rat783 (0.00), vm1084 (0.00), pcb1173 (0.00), u1432 (0.00), pr2392 (0.00), pcb3038 (0.00), fnl5915 (0.00), u13509 (0.00),	
Wang et al.	2004	lin105 (0.01), lin318 (0.00), att532 (0.00), dsj1000 (0.00),	
White, Yen	2004	eil51 (0.58), kroA100 (0.01), d198 (0.04), lin318 (1.3),	
Zou et al.	2004	lin318 (0.00), att532 (0.00), si535 (0.00), gr666 (0.00), rat783 (0.00), u1060 (0.00), fl1400 (0.00), r11889 (0.00),	only best values reported
Chan et al.	2005	rat575 (0.03), u724 (0.00), rat783 (0.00), vm1086 (0.07), pcb1173 (0.02),	
Ray et al.	2005	kroA100 (0.31), d198 (0.31), ts225 (0.19), pcb442 (2.26), rat783 (7.22)	
Xuan, Li,	2005	eil51 (1.47), eil76 (1.89), eil101 (2.26), chn150 (1.21), gr202 (1,20), pcb442 (0.89)	
Yan et al.	2005	st70 (0.00), eil76 (0.00), kroA200 (0.00), rd100 (0.00), pr136 (0.00), pr144 (0.00), a280 (0.00),	
Wang, Cui, Wan, Wang	2006	eil51 (0.00), st70 (0.00), kroA100 (0.00), ch130 (0.00), lin318 (0.00)	uses gene therapy to improve the performance
Wang, Han, Li, Zhao	2006	lin105 (0.00), lin 318 (0.00), att532 (0.00), dsj1000 (0.00)	only bests values are reported

CHAPTER 3

DEVELOPMENT OF THE EVOLUTIONARY ALGORITHM

This chapter describes development of a GA using operators based on conventional TSP heuristics. Specifically, we focus on the Nearest Neighbor Crossover (NNX) with multiple parents. As the name implies, NNX is derived from the well-known TSP heuristic called as Nearest Neighbor heuristic. We also use edge exchange and node insertion ideas in developing mutation operators.

Throughout the chapter, alternative settings for the design of GA are described in detail, and then the experiments conducted using these alternatives setting are reported. The results of these experiments are used in generation of new alternatives that may improve the performance of the NNX operator. In the initial experiments, a set of small problem instances from TSPLIB (Reinelt, 2007) are used for testing alternative settings. Later on, larger instances are also included, and the final form of the algorithm is reached. The chapter concludes with reporting the best settings that are determined by the convergence analysis conducted on a larger problem instances, and the computational results of these settings for large problems.

3.1. Nearest Neighbor Crossover (NNX) with Multiple Parents

NNX is based on the well-known Nearest Neighbor heuristic (NN), which is based on visiting the nearest city to the current city every time. Sönmez (2003) mentions that NN usually generates tours which are around 25% longer than the optimal tour. The main reason of this deviation is the few but severe mistakes that are present in the solution. The number of feasible edges short starts to decrease during the construction of a tour, thus the algorithm is forced to include the forgotten cities at the expense of some long

edges. However, NN tours contain paths of short edges. NNX tries to eliminate these few severe mistakes by combining good edges that are present in the parents. In NNX, the edges that are present in the parents are used to create a union graph of the parents. Then, NN is applied on this union graph instead of the complete instance graph only if there are no feasible edges left in the union graph. NNX selects the shortest of all feasible edges from the complete graph.

NNX has flexibility on the number of parents as there is no limitation on the number of edges that are included in the union graph. In the extreme case, all the edges can be present in the union graph, and the solution is thus identical to the NN solution. Mühlenbein (1989) was the first to introduce more than a pair of parents to the GAs, with his voting recombination operator. This operator does not resemble the NNX, yet the idea of p-sexual crossover is applicable to the NNX. This novice approach is different from the general stream of crossover operators, as they tend to preserve the good properties of either one parent or the other. Using more than two parents can bring improvement to the solution quality, as good properties of more than two individuals can be preserved. Yet, the effect of more than two parents cannot be easily judged, as the performance of NNX is affected by the number of good edges that are acquired from the complete graph when the algorithm gets stuck, as well as the edges that are present in the parents.

The initial (prototype) evolutionary algorithm used for evaluating and improving the performance of NNX is given in Figure 3.1. We use path representation of our solutions, and the fitness value of a solution is the tour length.

```
Generate initial population randomly
Repeat
        Select k parents
        Apply NNX to obtain child(ren)
        Replace some population members with child(ren)
Until Population Converges
```

Figure 3.1 Evolutionary algorithm for NNX

Sönmez (2003) has experimented with different initial populations and concluded that NNX gives best results when the initial population is generated randomly. Demir (2004) also suggests use of NNX with random initial population and reports significant solutions on constrained TSPs. Moreover, Larrañaga et al. (1999) reports that ERX operator, which is similar to the NNX, gives the best results among all the operators

when random initialization is used. Therefore, the initial population used in our initial experiments is generated randomly.

The size of the population is another important factor that affects the performance of an EA. There are applications where different population sizes are reported to give promising results. Sönmez (2003) has experimented with two different population sizes, 50 and 100, for small problems (i.e. $n \leq 250$). The population size becomes an important factor as the problems size increases, because different members of the population represent different points in the solution space. In the initial experiments of our study, the population size is set to the problem size for small problems. For large problems, the population size is fixed as 200 due to computational constraints similar to Larrañaga et al. (1999).

There are two popular strategies for reproduction in the literature. The first one is the generational approach, where the whole population reproduces and newly generated solutions replace in the previous population. The second reproduction strategy, which employed in this study, is the steady-state evolution, where only a small group of parents is used to generate a group of children, and the children generated are introduced to the population.

Parent selection for reproduction is done using the method of GENITOR proposed by Whitley (1989). This method gives the possibility to experiment with different selection pressures without a major difference in the algorithm. The method is a rank based selection method where the selection pressure is adjusted using a parameter η, which varies between 1 and 2. The probability of selection of an individual for crossover is calculated using equation 3.1, where i denotes the rank of the i^{th} individual, in a population with N individuals.

$$p_i = \frac{1}{N}\left[\eta - 2(\eta - 1)\left(\frac{i-1}{N-1}\right)\right] \tag{3.1}$$

The selection is totally random when $\eta = 1$, and the best individual is favored most when $\eta = 2$. Actually the selection probability assigned to the best individual is twice as large as the average when $\eta = 2$ (Larrañaga et al., 1999). Different η values are utilized in the initial experiments of our study, as none of the different selection probability assessment methods are proven to be the best for all types of settings. Assigning probabilities according to ranks have been practiced successfully with impressing

results, but there are cases when the random selection has been used and has produced the best result. In general, different replacement, crossover, and mutation methods affect the quality of the solutions interacting with the selection pressure.

The edges present in the selected parents are used to generate the union graph. The child generated using the union graph is dependent on the starting point of the tour construction. Thus, NNX makes it possible to generate different children using the same parents. In this study, the starting point of each child is selected randomly, due to the promising results reported by Sönmez (2003) and Demir (2004). The number of children generated can play an important role in exploration of the search space. Moreover, the algorithm of EAX (Nagata, Kobayashi, 1997) also includes generation of more than one child.

The replacement strategy is another important factor as the members that are replaced may be holding very important edges that cannot be reintroduced to the population easily. Replacement also affects the convergence behavior of the algorithm. The population can easily converge to local optima if the replacement strategy does not favor differences among members. There are two common replacement strategies that are employed in the literature. One is the family competition or Replacement of the Worst Parent (RWP), and the other is Replacement of the Worst Population Member (RWM). Demir (2004) uses RWP, with steady-state evolution. In RWM, the best child generated replaces the worst population member if its fitness value is better than that individual. Similarly, in RWP, the child replaces the worst parent only if its fitness value is better. We experiment with both RWP and RWM in this study.

The aim of the initial experiments with NNX is to improve its performance using suitable parameter settings. Therefore, there is no mutation operator in the EA given in Figure 3.1. The mutation operator will be taken into consideration after the parameters that give good solutions with NNX are identified.

The number of generations is chosen as 10,000 for small problems as a convergence criterion. According to the results of the experiments, the algorithm has converged for all small problems by the end of 10,000 generations.

3.2. Initial Experiments with NNX

The performance of NNX is investigated using different parameter settings to improve the solution quality. Slight changes in the structure of initial population and NNX are

also implemented. Eight small instances from TSPLIB are used in these experiments. These instances, which are also used by Sönmez (2003), are berlin52, eil101, bier127, ch130, ch150, u159, kroA200, and pr226. This section concludes describing the best configuration for the algorithm given in Figure 3.1.

3.2.1 Parameter Settings

We have experimented with four parameters of the algorithm. The values for each of these parameters and their representations throughout the chapter are given below:

- The Number of Parents for NNX (P): The numbers of parents tried are 2, 3 and 6.
- Etha (η): Values tried are 1 (random selection), 1.5 and 2 (higher selection pressure).
- The Number of Children (C): The number of children is set as 1 and 10.
- Replacement (R): Replacement with the worst parent (RWP) and replacement with the worst population member (RWM) are tried.

All possible combinations of the above parameter settings are replicated 30 times for each problem. The solution quality is assessed using two measures. The first measure is "the best solution of 30 replications (Best)" is an indicator of the exploratory power of the algorithm. The second measure is "average of solutions of 30 replications (Avg)" and is a good indicator of the consistency of the solutions. The solutions obtained with all parameter combinations are ranked based on both quality measures. The best three configurations for each problem are given in Table 3.1.

In Table 3.1, the three columns following the first column give the best three parameter combinations when solutions are ranked according to the best solution (Best). The last three columns give the best three parameter settings when the solutions are ranked according to the average solution (Avg).

Observing that the best parameter settings in Table 3.1 using either quality measure never predict the use of a single child, the best parameter settings when only one child is generated are given in Table 3.2. We consider the single child case because generally 10 children take significantly longer computational time. In Table 3.2 the second column gives the best parameter setting when the average deviation of 30 replications is

considered; the third column gives the best parameter setting when the best of 30 replications is considered.

Table 3.1 Best parameter settings for small problems

		Settings for Best			Settings for Avg		
		1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
P	berlin52	3	2	2	3	2	2
Etha		1	1	1	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWM
Best Deviation (%)		0.00	0.00	0.00	0.00	0.00	0.00
Avg Deviation (%)		1.18	1.24	2.12	1.18	1.24	2.12
P	eil101	2	2	2	2	2	6
Etha		1.5	1	1.5	1.5	1	1.5
C		10	10	10	10	10	10
R		RWM	RWP	RWP	RWP	RWP	RWM
Best Deviation (%)		0.64	0.79	1.43	1.59	0.79	2.86
Avg Deviation (%)		5.66	4.32	4.53	4.26	4.32	4.35
P	bier127	2	2	2	2	2	2
Etha		2	1	1	1.5	2	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.47	0.61	0.64	0.65	0.47	0.61
Avg Deviation (%)		1.7	1.74	1.86	1.56	1.7	1.74
P	ch130	6	2	6	2	3	3
Etha		1	1	2	1	1.5	1
C		10	10	10	10	10	10
R		RWM	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		1.29	1.64	1.8	1.64	2.08	2.03
Avg Deviation (%)		3.8	3.12	4	3.12	3.28	3.41
P	ch150	2	2	2	2	3	2
Etha		1.5	1	2	1	1	1.5
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		1.18	1.23	1.24	1.23	1.42	1.18
Avg Deviation (%)		2.12	1.95	2.32	1.95	2.04	2.12
P	u159	2	2	6	2	2	3
Etha		1	1.5	1	1	1.5	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.62	0.79	0.85	0.62	0.79	0.98
Avg Deviation (%)		1.66	1.71	2.37	1.66	1.71	1.87
P	kroA200	6	6	3	3	6	2
Etha		1.5	1	1	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.96	1.14	1.21	1.21	1.14	1.25
Avg Deviation (%)		2.05	1.75	1.74	1.74	1.75	1.83
P	pr226	3	2	2	2	6	3
Etha		1	2	1.5	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.72	0.8	0.83	0.86	1.01	0.72
Avg Deviation (%)		1.57	1.77	1.6	1.44	1.51	1.57

The best configuration for one child according to Table 3.2 is using 2 parents, selecting randomly with small problems and selecting the good individuals when the problems size is larger. Replacement with parents is the best replacement strategy, but the replacement with population members is also the best strategy in some problems. Figure 3.3 (explained in detail below) can also be used to evaluate the interaction of the parameter settings when only one child is generated. Generating only one child is discarded from further consideration in as the results are relatively poor according to the results in Table 3.2 and Figure 3.3.

Table 3.2 Best parameter settings when only one child is generated

		Avg	Best			Avg	Best
P	berlin52	2	2	**P**	ch150	6	2
Etha		1	1	**Etha**		1	1
R		RWP	RWP	**R**		RWM	RWP
Best Deviation (%)		0	0	**Best Deviation (%)**		1.95	1.46
Avg Deviation (%)		2.28	2.28	**Avg Deviation (%)**		2.69	2.7
P	eil101	2	2	**P**	kroA200	3	2
Etha		1	2	**Etha**		1.5	1
R		RWP	RWP	**R**		RWP	RWP
Best Deviation (%)		2.7	2.23	**Best Deviation (%)**		1.49	1.1
Avg Deviation (%)		5.53	5.8	**Avg Deviation (%)**		3.81	3.89
P	bier127	2	3	**P**	u159	3	3
Etha		1.5	1.5	**Etha**		1.5	1.5
R		RWP	RWP	**R**		RWM	RWM
Best Deviation (%)		1.07	0.89	**Best Deviation (%)**		1.82	1.82
Avg Deviation (%)		2.33	2.69	**Avg Deviation (%)**		3.12	3.12
P	ch130	2	3	**P**	pr226	2	2
Etha		1.5	1.5	**Etha**		2	2
R		RWP	RWM	**R**		RWP	RWP
Best Deviation (%)		2.77	2.37	**Best Deviation (%)**		1.27	1.27
Avg Deviation (%)		4.51	4.98	**Avg Deviation (%)**		3.72	3.72

The results of individual runs that are summarized in Table 3.1 are analyzed using ANOVA, in order to find the optimal configuration with statistical significance. Appendix B summarizes the results of the ANOVA and the residual plots required to verify the assumptions of ANOVA. Main effect and interaction plots of the parameters are given in Figures 3.2 and 3.3. The number 1 stands for RWM, 2 for RWP.

According to the results of ANOVA, all four parameters have significant effects. Moreover, interactions between the pairs P and η, P and R, C and η, C and R, η and R are also statistically significant. Thus, the decisions are mostly based on the interaction plot. Although 3 parents seem to give slightly the better result in the main effects plot (Figure 3.2), using the best configurations in terms of the children (10), η (1.5) and RWP as replacement strategy, the best choice is 2 parents considering all the two-way

interactions given in Figure 3.3. Generating 10 children gives the best results regardless of the interactions. Similarly, smaller values of η give better results, although not very obvious when interaction with the number of parents is considered. The most unexpected result is observed with the replacement strategy; p-sexual crossover dominates 2-parent crossover when the child replaces the worst population member, whereas two parents give the best result then RWP is used.

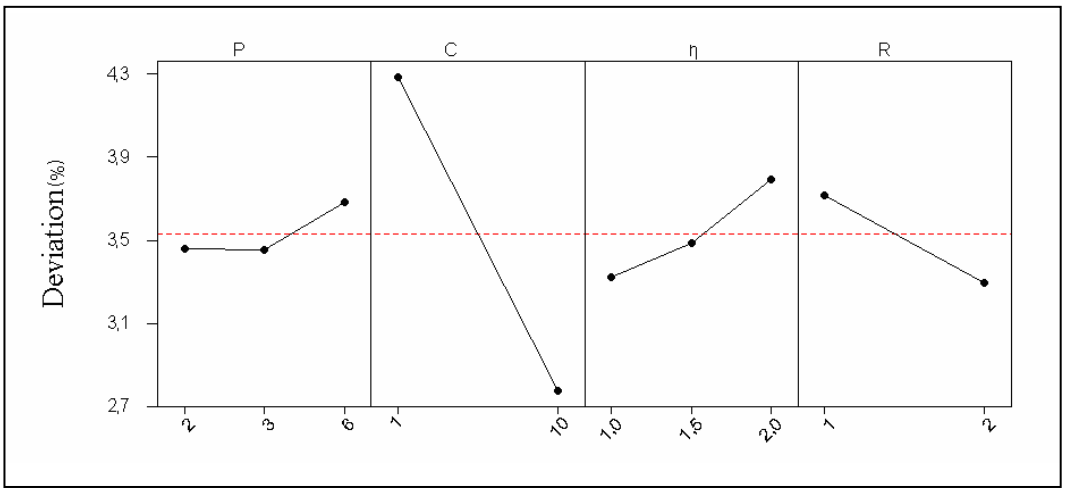


Figure 3.2 Main interaction plots of the experiment parameters

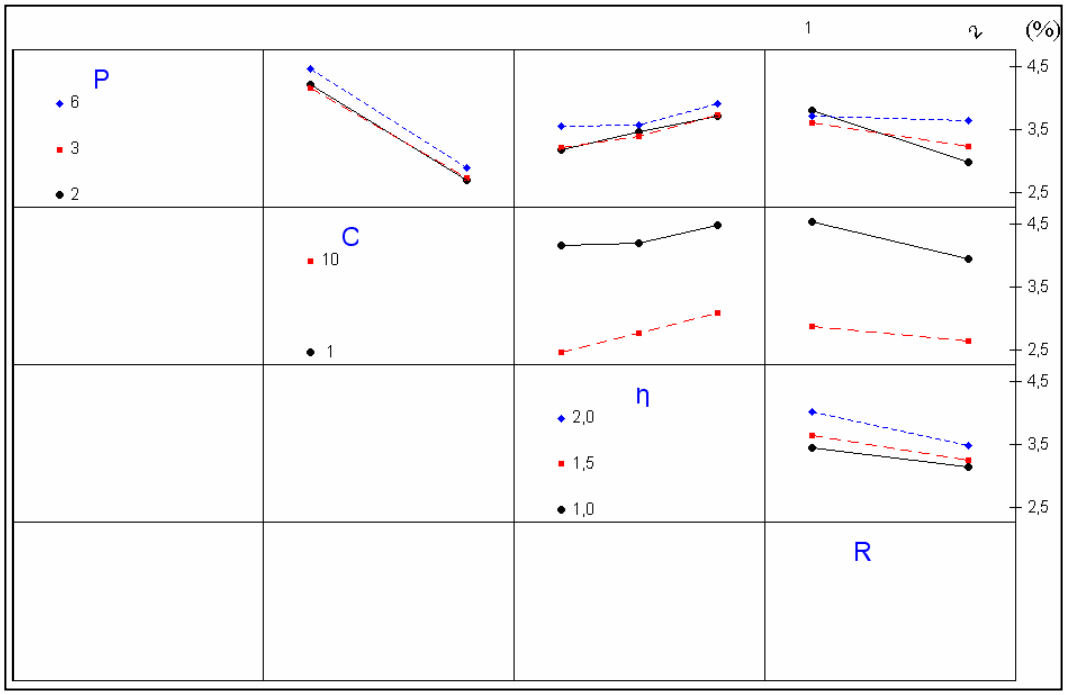


Figure 3.3 Interaction plot of the parameters under consideration

A promising configuration seems to be selecting two parents at random (η=1), and generating 10 children from these parents using NNX, the best of which will replace the worst of the parents.

3.2.2 Alternative Initial Population Generation

Another setting that has effect on the performance of NNX is the initial population, according to Sönmez (2003). She states that the best results are formed when the population is generated randomly. It can be seen in Table 3.3 that the randomly generated initial population has a very poor solution quality. A very simple alternative to this is to generate the initial population using NN. The initial solution with NN is generated starting the tour with each of the cities. The percent deviation of the initial population using NN is much better than the random initial population.

Table 3.3 Deviations of the initial populations when created randomly and using NN

		Random			Nearest Neighbor Heuristic		
Problem instance	**Optimal**	**Minimum Deviation (%)**	**Average Deviation (%)**	**Maximum Deviation (%)**	**Minimum Deviation (%)**	**Average Deviation (%)**	**Maximum Deviation (%)**
berlin52	7542	242.40	296.04	338.18	8.47	24.31	36.54
eil101	629	389.59	444.3	497.74	18.6	32.35	45.15
bier127	118282	384.75	431.59	473.65	13.25	23.57	32.36
ch130	6110	658.54	726.85	826.85	16.68	26.61	44.63
ch150	6528	652.05	725.38	793.24	8.96	17.53	27.90
u159	42080	859.55	969.01	1071.37	15.47	28.68	35.67
kroA200	29368	946.46	1058.96	1170.18	17.62	28.02	43.16
pr226	80396	1816.76	2009.19	2188.64	15.12	24.16	41.73

The algorithm starting with the alternative population generation method is tried for smallest two problem instances and a summary can be found in Table 3.4. The results suggest that NNX is not capable of escaping the local optima created by NN heuristic (especially see the second problem's results in the Table). Thus, further investigation using NN to generate the initial population is abandoned.

In Table 3.4, six parents can be seen in both of the problems instances even for the smallest two problems. NNX is forced to merge the good edges from different parents when used with individuals constructed using NN heuristic.

3.2.3 Alternative Crossover (NNX-a)

Recombination of edges from different parents is the premise of NNX, but this cannot be done when one of the parents dominates the other. There are cases where all edges in the children come from the same parent.

Table 3.4 Results for small problems when initial population is generated using NN Heuristic

		Settings for Best			Settings for Avg		
		1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
P	berlin52	2	3	6	2	3	6
Etha		1	1	1	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.07	0.07	0.07	0.07	0.07	0.07
Avg Deviation (%)		1.04	1.06	1.11	1.04	1.06	1.11
P	eil101	6	6	3	3	2	2
Etha		1	2	1.5	1	1	1.5
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		2.86	3.34	3.5	4.61	4.61	3.82
Avg Deviation (%)		6.72	7.42	6.73	6.45	6.52	6.58

An alternative worth investigation is the selection of edges from the union graph using a method similar to the one employed in EAX Crossover (Nagata, Kobayashi, 1997). EAX constructs the AB cycles by selecting edges alternating between the two parents then uses these AB cycles in creating new solutions. In our NNX-a, if two successive nearest neighbors are from the same parent, we take the second neighbor from the other parent, even if it is farther. This way we make sure that edges are selected by alternating between the parents and children are different from their parents.

Table 3.5 summarizes the results for NNX-a. When we compare Tables 3.1 and 3.5, we observe that enforcing edges from different parents does not bring any improvement in terms of average percent deviation when more than one child is generated, and does not have a very large impact when only one child is generated. Thus, alternating between parents when selecting the edges is discarded from further consideration.

Table 3.5 Best parameter settings for small problems using alternating parents

		Settings for Best			Settings for Avg		
		1st	**2nd**	**3rd**	**1st**	**2nd**	**3rd**
P	berlin52	2	2	2	2	2	2
Etha		1	1.5	1.5	1	1.5	1.5
C		10	10	1	10	10	1
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.07	0.07	0.07	0.07	0.07	0.07
Avg Deviation (%)		1.24	1.37	1.56	1.24	1.37	1.56
P	eil101	2	2	2	2	2	2
Etha		1.5	1	1	1	1.5	2
C		10	10	1	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		1.27	2.07	2.23	2.07	1.27	2.54
Avg Deviation (%)		4.28	4.26	6.42	4.26	4.28	4.86
P	bier127	3	2	2	3	2	2
Etha		1	2	1.5	1.5	1	1.5
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.61	0.77	0.76	0.78	0.95	0.89
Avg Deviation (%)		1.82	1.88	2.33	1.64	1.73	1.78
P	ch130	2	3	3	2	2	3
Etha		1.5	2	1.5	1	1.5	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		2.14	2.5	2.5	2.86	2.14	3
Avg Deviation (%)		3.8	4.22	4.86	3.77	3.8	4.06
P	ch150	2	2	2	2	2	2
Etha		1	1.5	2	1	1.5	2
C		10	10	10	10	10	10
R		RWM	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.98	1.35	1.36	1.44	1.35	1.61
Avg Deviation (%)		2.57	2.25	3.5	2.16	2.25	2.46
P	u159	2	2	3	2	2	2
Etha		1.5	2	1	1	1.5	2
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.76	0.93	0.94	1.06	0.76	0.93
Avg Deviation (%)		1.93	2.27	2.2	1.66	1.93	2.27
P	kroA200	3	3	2	2	3	2
Etha		1.5	1	1	1	1	1.5
C		1	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		1.43	1.66	1.78	1.78	1.69	2.03
Avg Deviation (%)		4.57	3.56	2.72	2.72	2.96	3.41
P	pr226	3	3	2	2	2	2
Etha		1	2	2	1	1.5	2
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.82	0.94	0.95	1.01	1.04	0.95
Avg Deviation (%)		1.86	2.03	1.62	1.32	1.33	1.62

3.2.4 Results and Discussion of the Initial Experiments

The best settings among the possible alternatives tried to improve the performance of NNX can be summarized as follows. NNX operator gives the best results, when the initial population is generated randomly instead of using the NN heuristic. Using 2 parents in generating the union graph and selecting them randomly outperforms the other alternatives. The best settings so far are given in Figure 3.4.

Initial Population: Random
P = 2
$\eta = 1$
C = 10
R = RWP

Figure 3.4 Best parameter configurations after the initial experiments

3.3. Mutation Operators

The new edges NNX operator introduces into the population are limited, as NNX concentrates on the shortest edges in the union graph. There are cases observed that the child generated using the shortest edges of both parents results in children that are not better than their parents. An example where the child generated using only parental edges and is not shorter than both of the parents is demonstrated in Appendix C.

Children get new edges from the complete graph, only when tour construction on the union graph gets stuck. Three mutation operators are used in this study to increase the power of NNX to explore new edges, as the edges in the optimal solution are not always the shortest edges.

Jog et al. (1989) suggest using improvement heuristics such as 2-opt and 3-opt as mutation operators, which makes it possible to incorporate problem specific information into the GA (Sönmez, 2003). The first mutation that we use is based on 2-opt, concentrating on the longest edges. The second mutation is similar to the first one, but randomly selected edges are considered for deletion instead of the longest ones. The third mutation is based on the cheapest insertion, another well-known TSP heuristic. We use edge exchange and insertion for mutation because these two are fundamental improvement moves for TSP.

3.3.1 Longest Edge Mutation (LEM)

This mutation aims to eliminate the long edges in a child, which cause large increases in the tour length. All the edges in an individual are ranked according to their lengths and the longest edges are tried to be eliminated by restricted 2-opt moves. This mutation is inspired from the fact that, NN tours contain paths of short edges and few severe mistakes (long edges). Thus concentrating only on the longest edges in an individual can bring significant improvement to the solution. The algorithm of LEM is given in Figure 3.5.

```
for i = 1 to 15
        Choose the longest edge
        for j = 1 to n-3
                Delete the j^th non-adjacent edges
                Calculate the length of connecting the subtours
        end for
        Apply the best improving (if any) of these n-3 possible exchanges
end for
```

Figure 3.5 Algorithm of longest edge mutation

A limitation arises as a 2-opt move takes $O(n^2)$ time, and examining all the edges in relatively larger problems would take a very long time. Therefore, the number of 2-exchange moves is limited to 15 at most.

2-exchange moves can be applied until the first improving move is found within the limit of 15 trials (LEM 1), or can continue to examine all 15 longest edges in decreasing order of length regardless of the improvements (LEM 2). As the mutation is based on an improvement heuristic, the trade-off between computation time and solution quality needs to be considered.

3.3.2 Random Edge Mutation (REM)

This mutation is similar edges with LEM, however it tries to eliminate randomly selected edges by 2-opt moves instead of the longest ones. This saves the time to sort the edges and still introduces some new edges to the tour. The algorithm of REM can be seen in Figure 3.6.

```
for i = 1 to 15
        Choose an edge at random
        for j = 1 to n-3
                Delete the j^th non-adjacent edges
                Calculate the length of connecting the subtours
        end for
        Apply the best improving (if any) of these n-3 possible exchanges
end for
```

Figure 3.6 Algorithm of random edge mutation

The number 2-opt moves is again limited to 15, without considering improvement in the previous moves.

3.3.3 Cheapest Insertion Mutation (CIM)

The third mutation has the advantage of bringing in new edges randomly, whereas both the NNX operator and LEM use edge lengths to find good edges. CIM is used as an alternative mutation, where a node is selected randomly and removed from the individual. All possibilities are checked for insertion of removed node to find the insertion point that causes the minimum increase in the tour length. The algorithm runs in $O(n^2 \log n)$ time, therefore we limit the number of insertions in order to have an efficient mutation operator. The algorithm of CIM can be seen in Figure 3.7

```
for i = 1 to 15
        Choose one node at random
        Try inserting it in every possible place in the tour
        Apply best improving (if any) of these n-2 insertions
end for
```

Figure 3.7 Algorithm of cheapest insertion mutation

The number of nodes that are selected for CIM is limited to 15. Another decision is whether to stop the algorithm if an improvement is found (CIM 1) or continue for 15 different nodes regardless of the improvements found (CIM 2).

3.4. Further Experiments with Mutation

The behavior of NNX combined with the mutation operators is to be investigated, yet the idea of p-sexual crossover is still worth further consideration. A slightly modified replacement strategy is also implemented in further experiments.

LEM is used in the further experiments as it is promising with for eliminating the longest edges. On the other hand, CIM is included, as the edges CIM brings in are not easily incorporated with NNX or LEM. For each child generated either LEM or CIM is used for mutation with equal chances. REM is considered at a later stage as an alternative to LEM.

The idea of replacing the parents with their children only if the children are better than their parents results in waste of children. An alternative scheme is to replace worst half of the parents without considering if the children are better than them (RWH).

It is possible that more than half of the children are better than the parents that are used to generate them. Another alternative replacement strategy can be continuation of the replacements as long as the children are better than the parents, and replacing the parents that fall behind the children in fitness.

In this part of the study, every new child generated starts from the last node visited in the previous child as implemented by Demir (2004), to ensure that children are different from each other.

3.4.1 Parameter Settings for Further Experiments

The parameters that we consider in these experiments are the number of parents and children, mutation and the new replacement strategy (RWH). The number of parents and the number of children are selected to be equal, as the new replacement strategy aims to replace at least half of the parents with half of the newly generated children. The experimental values for each of these parameters are:

- The Number of Parents (P) and Children (C): The values tried are 2, 4, 6 and 12.
- Longest Edge Mutation (LEM): Stopping after the first improvement (1) and trying all 15 longest edges (2).
- Cheapest Insertion Mutation (CIM): Stopping after the first improvement (1) and trying to insert 15 different random nodes (2)
- Replacement (R): Replacing worst half of the parents with the best half of children (1) and replacing more than half of the parents with children as long as the children have better fitness values (2).

All possible combinations of the above parameter settings are replicated 30 times. The termination condition is again set as 10,000 generations.

The parent selection is done randomly in these experiments as increases in η cause larger deviation from the optimal and, according to the results of the ANOVA, η has a statistically significant effect on the deviation.

Table 3.6 is designed similar to Table 3.1. The decision parameters are the number of children (equal to the number of parents), the type of LEM operator, the type of CIM operator and the type of replacement. The Table suggests that better results are observed when the number of children is small (2 or 4), LEM and CIM are continued 15 times regardless of the improvement and replacement is continued as long as the children are better than their parents after half of the parents are replaced.

3.4.2 Results of Further Experiments

The results obtained with the above parameter combinations are analyzed using ANOVA, in order to find the optimal configuration with statistical significance. Appendix D summarizes the results of the ANOVA and the residual plots required to verify the assumptions of ANOVA. According to ANOVA, parameters C and LEM have significant effect as well as the interaction between this pair. Main effect plots of the parameters for percent deviation are given in Figure 3.9, and interaction plots are given in Figure 3.10.

The previous observations in Section 3.2.4 support that generating more children improves the solution quality. However, we do not see this in Figure 3.9 because of the deterioration of the solution due to the increase in the number of parents. The best solutions are obtained when only two parents are used to generate two children. A reason for this observation may the fact that the number of edges borrowed from the complete graph starts to decrease when the number of parents is larger. The edges that are brought in by the children with poor fitness values may also contribute this.

Trying to improve all the 15 longest edges using LEM is better than quitting at the first improvement. Inserting 15 random nodes with CIM instead of ending trials when an improvement is found gives better results, although the difference is not statistically significant. Replacement does not have any effect on the deviation from the optimal.

Table 3.6 Best parameter settings for small problems using further experiments

		Settings for Best			Settings for Avg		
		1st	2nd	3rd	1st	2nd	3rd
P	berlin52	3	2	2	3	2	2
Etha		1	1	1	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWM
Best Deviation (%)		0	0	0	0	0	0
Avg Deviation (%)		1.18	1.24	2.12	1.18	1.24	2.12
P	eil101	2	2	2	2	2	6
Etha		1.5	1	1.5	1.5	1	1.5
C		10	10	10	10	10	10
R		RWM	RWP	RWP	RWP	RWP	RWM
Best Deviation (%)		0.64	0.79	1.43	1.59	0.79	2.86
Avg Deviation (%)		5.66	4.32	4.53	4.26	4.32	4.35
P	bier127	2	2	2	2	2	2
Etha		2	1	1	1.5	2	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.47	0.61	0.64	0.65	0.47	0.61
Avg Deviation (%)		1.7	1.74	1.86	1.56	1.7	1.74
P	ch130	6	2	6	2	3	3
Etha		1	1	2	1	1.5	1
C		10	10	10	10	10	10
R		RWM	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		1.29	1.64	1.8	1.64	2.08	2.03
Avg Deviation (%)		3.8	3.12	4	3.12	3.28	3.41
P	ch150	2	2	2	2	3	2
Etha		1.5	1	2	1	1	1.5
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		1.18	1.23	1.24	1.23	1.42	1.18
Avg Deviation (%)		2.12	1.95	2.32	1.95	2.04	2.12
P	u159	2	2	6	2	2	3
Etha		1	1.5	1	1	1.5	1
C		10	10	10	10	10	10
R		RWP	RWP	RWM	RWP	RWP	RWP
Best Deviation (%)		0.62	0.79	0.85	0.62	0.79	0.98
Avg Deviation (%)		1.66	1.71	2.37	1.66	1.71	1.87
P	kroA200	6	6	3	3	6	2
Etha		1.5	1	1	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.96	1.14	1.21	1.21	1.14	1.25
Avg Deviation (%)		2.05	1.75	1.74	1.74	1.75	1.83
P	pr226	3	2	2	2	6	3
Etha		1	2	1.5	1	1	1
C		10	10	10	10	10	10
R		RWP	RWP	RWP	RWP	RWP	RWP
Best Deviation (%)		0.72	0.8	0.83	0.86	1.01	0.72
Avg Deviation (%)		1.57	1.77	1.6	1.44	1.51	1.57

The interaction between the parameter C and LEM is statistically significant. According to the interaction plot in Figure 3.10 LEM gives better results as the number of parents

and children decreases. According to the Figures 3.9 and 3.10, the best strategy is to generate 2 children from 2 parents and to apply LEM and CIM for 15 trials regardless of the improvement.

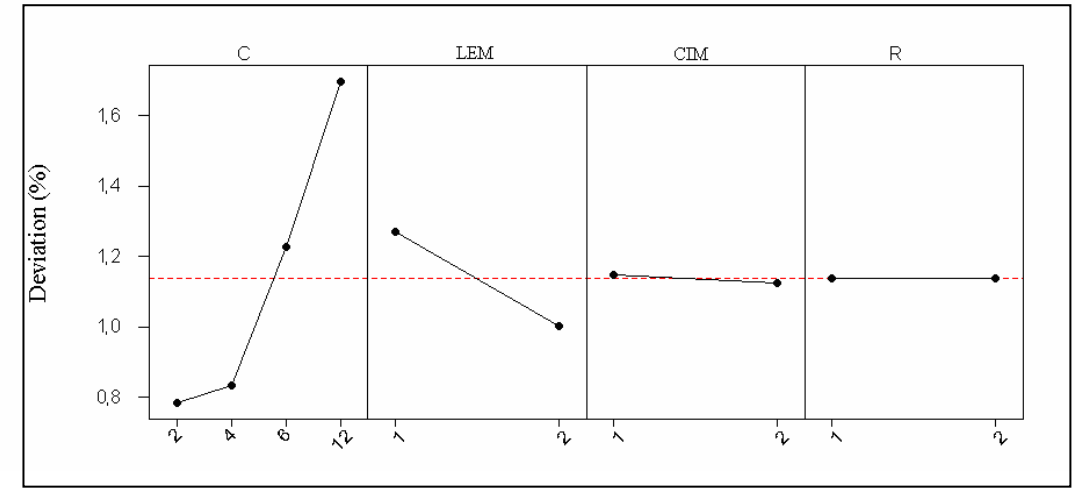


Figure 3.8 Main effects plots of the experiment parameters

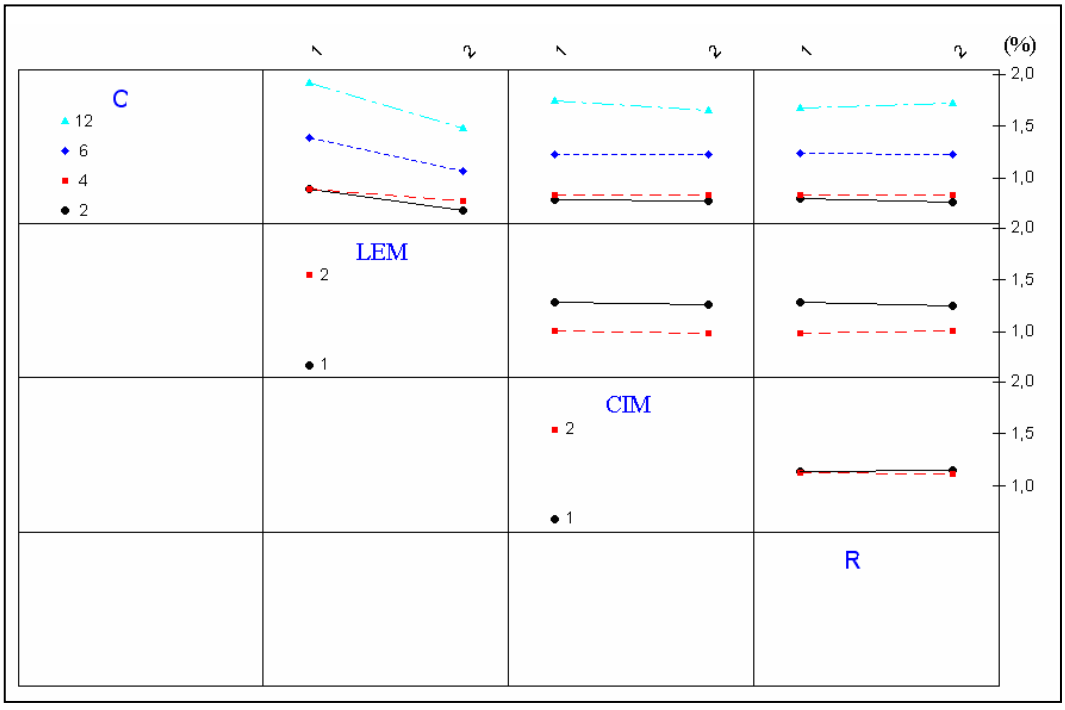


Figure 3.9 Interaction plots of the parameters under consideration

The best settings among the alternatives tried in further experiments are given in Figure 3.8.

Initial Population: Random
P = C= 2
η = 1
LEM = CIM=15
R = RWH

Figure 3.10 Best parameter configurations after the initial experiments

Table 3.7 compares the results of the initial and further experiments. The best deviation and average deviation gives the best and average of 30 replications. Note that the RWH with P = 2 is equivalent to RWP with P = 2 in Section 3.2. Both performance measures support that the mutation operators bring in significant improvement.

Table 3.7 Comparison of initial and further experiments with small problems

	Initial Experiments P = 2, η = 1, C = 10, R = RWP		**Further Experiments** P = C =2, η = 1, LEM = CIM = 15, R = RHW	
Problem	**Best Deviation (%)**	**Avg Deviation (%)**	**Best Deviation (%)**	**Avg Deviation (%)**
berlin52	0.00	1.24	0.00	0.10
eil101	0.79	4.32	0.00	1.60
bier127	0.61	1.74	0.28	0.43
ch130	1.64	3.12	0.49	1.08
ch150	1.23	1.95	0.32	0.46
u159	1.30	5.09	0.00	0.15
kroA200	1.25	1.83	0.14	0.63
pr266	0.86	1.44	0.35	0.79
Average	0.96	2.59	0.20	0.65

According to the results although C = 10 in initial experiments is reduced to C = 2 in further experiments, the solution quality is improved. This is mainly due to the mutation operators used.

The computation times vary between 6 and 38.97 seconds in the initial experiments, and between 3.7 and 16.8 in the further experiments on a PC with AMP Turion 64 x2 1.6 GHz processor having 512 MB or RAM. This suggests that improvements on the solution quality do not require extra computational times. They are even fast runner for this experiment.

3.5. Discussion and Convergence Analysis

The results of the both experimental investigations are consolidated in Figure 3.11. However, the sizes of the test problems used so far are relatively small, and the convergence of this best configuration when the increase in problems size needs to be investigated. Thus, the NNX configuration by which the best results are obtained is tested with larger problems to find out the effect of the increase in problem size on the convergence behavior of the algorithm. The instances are again selected form TSPLIB (pcb442, rat575, pr1002).

The number of parents is fixed as 2, the best value when all the experiments are considered. However, according to Figures 3.2 and 3.3, the best results are obtained when the number of children is large. A method that uses a limited number of parents yet generates numerous children is necessary; therefore, we adopt the child generation scheme proposed by Nagata and Kobayashi (1997). In our modified scheme, at most 10 children are generated using the same parents (union graph) as done in the initial experiments, but the first improving child is accepted and generation is terminated. The reason for generating more than one child is that the number of children with fitness values that are better than their parents' decreases as the population starts to converge. When a pair of parents fails to generate a child that is better than at least one of the parents in these 10 trials, a new pair is selected. At most 10 randomly chosen parent pairs are used until an improving child is found.

LEM and CIM are tested on the first improving child with equal probabilities. LEM tries to find a 2-opt move that results in improvement by deleting 15 longest edges in the child. CIM tries to improve the place of 15 randomly selected nodes in the child. Then, the child replaces worse of the two parents.

Initial Population: Random
P = 2
$\eta = 1$
C = 10
LEM = 15 trials
CIM = 15 trials
R = RWP

Figure 3.11 Best parameter configurations with mutation

The results for the larger problems with the configuration in Figure 3.11 can be seen in Table 3.8. The population size is fixed as 200 for these instances. It can be seen that the deviation for problem pr1002 (with more than 1000 cities) is high. Further analysis of these runs is required.

Table 3.8 Results of larger problems

Problem	CPU Time	Best Deviation (%)	Avg Deviation (%)	LEM / R	CIM / R	C / Gen	P / Gen
pcb442	313.5	2.96	3.09	0.83	0.7	56.5496	6.1
rat575	453.8	2.24	2.21	0.95	0.8	57.0934	6.1
pr1002*	906.0	7.92	9.51	0.99	1.0	61.5173	6.5

* Results of 10 replications of pr1002 are reported due to large time requirements.

In Table 3.8, when the generation limit of 10,000 is reached, in every generation six pairs of parents (see P / Gen) are used to generate the union graph, which would be at most 10, and 60 children (see C / Gen) are generated on the average. Hence, it can be concluded that 10 parent pairs and 100 child generation trials are adequate, there is no need to increase the number of parents tried and the number of children generated from a union graph.

If an improving child is found in every generation, then the number of 2-opt trials by LEM and the number of node insertion trials by CIM should be 15 / 2 = 7.5 per generation, on average. These numbers however decrease to 4.03 for LEM and 4.04 for CIM, as finding an improving child becomes more difficult towards the end. The number of successful 2-opt moves is 0.92 (see LEM / R) and the number of successful insertions is 0.83 (see CIM /R) per replacement.

3.5.1 Convergence Analysis of pr1002

A detailed convergence analysis of pr1002 is conducted to investigate the factors that cause large deviation from the optimal solution, and low utilization of the mutation operators. Two different replications are investigated in detail. The convergence plots, the plots of percentage of individuals that contain edges taken from complete graph, plots of the replacement percentages, and a plot showing the percentage of optimal edges included in the solutions are given in Appendix E.

The algorithm brings substantial improvement at the initial phase when no mutation operator is used. Remember that the solution quality of the randomly generated initial population was very poor. Therefore, the deviation from the optimal solution drops from

22000% to 15% in the initial 5000 generations. However, the improvement after this initial phase is not very significant.

The plots in Appendix E suggest that the poor results are due to the slowing effect of mutation operators. The percentage of individuals that contain edges borrowed from the complete graph in the pure NNX algorithm has a major peak after the algorithm starts to converge, during which the newly borrowed edges are spread around the population. However, when LEM and CIM operators are employed, the peak where the population converges is spread in time. In other words, LEM and CIM slow the convergence making it possible to observe improvements at even after 30,000th generation.

The success of mutation operators can be measured with average number or exchanges per generation. The average number of exchanges per generation is expected to be around 7.5, as 15 moves are to be tried with probability of 0.5. The number of successive exchange moves of LEM drops to 0.05 per replacement after 15,000 generations. This mutation is quite time consuming. When there are 1002 edges present, it is possible to observe up to 300,000 sorting operations in the worst case, at least 40,000 sorting operations in the best case. Moreover, there is an exhaustive search for a second edge to replace the current one. Thus, the limitation of number of exchanges to 15 does not guarantee shorter CPU times.

The convergence behavior of REM and CIM combination is also investigated in Appendix E. REM and CIM combination preserves the peak observed in the pure NNX, avoiding the delay of convergence of the population. Moreover, the algorithm with REM and CIM converges to a better value.

A simple summary of the convergence analysis based on the performance of mutations on the problem instance pr1002 is summarized throughout Tables 3.9 – 3.11. Detailed results of two different replications are reported in these tables. The results of the first replication (S1) were the worst results during the replications Table 3.8, and the results of the second replication (S2) were average results.

The mutation operators do not affect the relative solution quality between two initial conditions, as S1 results in poorer results compared to S2 in both the presence and absence of mutation operators.

Table 3.9 CPU times and deviation results of pr1002 with different mutation operators' combinations

		S1		S2	
	Gen	**CPU Time**	**Deviation (%)**	**CPU Time**	**Deviation (%)**
No Mut.	5000	252	14.92	260	15.45
	10000	575	8.72	657	7.28
	20000	1881	7.80	1780	6.26
	30000	3411	7.80	3270	6.26
	40000	5094	7.80	4760	6.26
LEM and CIM	5000	539	13.04	556	13.76
	10000	1143	5.90	1422	9.01
	20000	1830	3.51	2563	3.72
	30000	3253	3.42	3815	3.07
	40000	4824	3.39	5332	2.92
REM and CIM	5000	208	15.82	196	14.39
	10000	597	10.29	514	8.58
	20000	1128	3.81	860	2.67
	30000	1565	2.73	1281	2.26
	40000	2086	2.71	1769	2.24

In Table 3.9, REM and CIM combination gives better results; moreover, it is also better in term of CPU times. This combination dominates the other alternative combinations, including the case without mutation. The high CPU time in the case without mutation is

Table 3.10 CPU times and results for pr1002 when non-improving moves are allowed with a given probability after 5000th generation

			S1		S2	
	Acceptance Probability	**Gen**	**CPU Time**	**Deviation (%)**	**CPU Time**	**Deviation (%)**
LEM and CIM	Pr = 0.05	5000	200	14.13	209	15.14
		10000	391	11.44	382	10.69
		20000	670	6.5	655	5.61
		30000	894	3.27	906	4.31
		40000	1147	2.88	1164	3.58
REM and CIM	Pr = 0.05	5000	203	15.82	187	14.4
		10000	407	12.31	375	12.04
		20000	766	9.06	693	8.03
		30000	1038	5.42	932	3.96
		40000	1268	3.75	1168	3.58
REM and CIM	Pr = 0.20	5000	200	15.82	186	14.4
		10000	320	14.06	302	12.46
		20000	539	11.58	518	11.29
		30000	742	9.88	719	9.01
		40000	944	8.09	904	6.81

not surprising, as the solution without mutation does not improve after 20,000 generations, enforcing 100 children at each generation. This child generation increases the CPU time to the values much larger than the time used by the mutation operator. Compared to REM, the sorting of solutions to find the longest edges in LEM consumes a great deal of time, while bringing in a limited improvement.

The results of convergence analysis (in Appendix E) suggest that the average number of replacements per generation decreases rapidly at the 5000th generation, regardless of the presence of a mutation operator. Furthermore, the percentage of individuals that contain edges taken from the complete graph per replacement has a steady decrease as the algorithm proceeds. The decrease in the average number individuals that contain of edges taken from complete graph may be compensated by accepting some non-improving children with a given probability. The results for two different acceptance probabilities (Pr) are compared in Table 3.10.

Table 3.11 CPU times and results for pr1002, when non-improving moves are allowed with a probability of 0.5

		S1		S2	
Generations and Accept. Pr.*	**Gen**	**CPU Time**	**Deviation (%)**	**CPU Time**	**Deviation (%)**
Gen ≥ 10000 Pr = 0.05	5000	197	15.82	203	14.4
	10000	580	10.7	662	8.59
	20000	810	10.03	994	5.74
	30000	1029	9.72	1294	4.82
	40000	1237	8.74	1470	3.89
Gen ≥ 20000 Pr = 0.05	5000	204	15.82	189	14.4
	10000	587	10.3	498	8.59
	20000	1231	5.29	844	2.89
	30000	1439	4.89	1031	2.6
	40000	1631	4.45	1203	2.58
Gen ≥ 30000 Pr = 0.05	5000	262	15.82	191	14.4
	10000	884	10.3	501	8.59
	20000	1530	5.29	847	2.89
	30000	2195	2.77	1736	2.56
	40000	2388	2.68	1923	2.56

* Number of Generations after which non-imptoving children are acepted, with given acceptance probability

The results of LEM and CIM improve if non-improving children are accepted after 5000th generation (Gen), yet the results are not as good as REM and CIM, accepting only improving children. The best value obtained is 2.88, while REM and CIM resulted in 2.41 (See Table 3.9). The limited improvement possibility and high computa tion time requirements made us to discard LEM and CIM from further consideration.

REM and CIM combination does not yield any improvement when non-improving moves are accepted after 5,000th generation. In fact, REM and CIM combination seems to have a peak in the percentage of edges borrowed from complete graph later than 5000 generations, thus accepting non-improving children later can bring an improvement for REM and CIM combination. REM and CIM are still in a phase that is capable of finding good children at 5000 generations. Some improvement can be observed if non-improving children are accepted when the ability of REM and CIM combinations to generate improving children is decreased. In Table 3.11 different three alternative generation levels for accepting non-improving moves are experimented and the results are given.

According to Table 3.11, increasing the number of generations after which the non-improving children are accepted improves the results with REM and CIM. Yet, the results when non-improving children are accepted are not as good as the result when only children that are better than their parents replace their parents. The non-improving moves are taken out of consideration.

3.6. The Final Algorithm

The detailed algorithm containing the best parameter settings resulting from the experiments conducted is given in Figure 3.12. The algorithm is designed starting with the algorithm in Figure 3.1. The population size is equal to the problem size for small problems ($n \geq 250$), and 200 for larger problems.

```
Generate initial population randomly
        Size = n for small problems (n < 250)
        Size = 200 for problems with n ≥ 250
for g = 1 to G
        for i = 1 to I
                Select k parents at random
                for j = 0 to J
                        Generate a child using NNX on union graph of parents
                        If the child is better than the worse parent go to mut
                end for
        end for

mut     Apply REM or CIM (with equal chances) to the child
        Replace the worse parent with the mutated child
end for
```

Figure 3.12 The final algorithm for NNX

The termination condition (G) is the number of generations, which is set as 10,000 for small problems, and 40,000 for problems with size larger than 250. The termination condition is to be modified parallel to the problems size for larger problems. The number of parent pairs (I) tried if no improving child is generated is limited to 10. The number of children (J) generated from the same union graph if an improving child is not generated is limited to 10.

The mutation operator is applied to children only if they are better than their worst parent. REM and CIM are applied with equal probability. The child generated after the mutation operators replaces the worst parent.

This algorithm is further experimented on problems with larger size.

3.7. Experimental Results for Large Problems

The final algorithm is experimented with five larger problems from TSPLIB (Reinelt, 2007). The problems selected are pr1002, nrw1379, u2152, u2392 and pcb3038. The percent deviations, the CPU time, and the number of generations the algorithm runs given as the averages of 15 replications in Table 3.12. The percent deviations are high compared to the results in experiments with small problems, and no optimal solutions are found.

Table 3.12 Results for larger problems

Problem	Gen	CPU Time	Best Deviation (%)	Avg Deviation (%)
pr1002	40000	1698.60	2.82	3.47
Nrw1379	40000	4535.27	6.19	7.95
u2152	60000	7052.27	2.04	2.21
pr2392	60000	12588.60	6.36	7.99
Pcb3038	80000	31703.50	13.16	15.12

The percent deviation of the problem nrw1379 is higher than the problems with larger number of cities. The structure of the problem is different compared to the other problems. All the problems except nrw1379 are clustered and there are relatively long edges between the clusters, on the other hand, the nrw1379 is relatively uniform. The plots of the coordinates of all problems are given in Appendix A. The ability of NNX with REM and CIM to find good solutions to clustered problems makes it attractive to be used in TSPB, where there are three main clusters when the distance matrix is converted to a TSP. The increase in deviation for larger problems is not very significant as the largest TSPB problems in literature consist of 1000 cities.

CHAPTER 4

TSP WITH BACKHAULS

TSP with Backhauls (TSPB) arises in three different applications. It is a constrained version of TSPPD (Lenstra and Rinnoy Kan, 1975), the single vehicle case of VRPB (Goeschalckx and Jacobs-Blecha, 1989), and the three-cluster version of the CTSP (Chrisman, 1975). The solution approaches for these problems can be modified to solve TSPB.

The number of studies concentrating on TSPB is limited and they can be grouped in two categories. The first group aims to solve TSPB imposing the backhaul constraints on TSP where as the second group transforms an instance of TSPB to an instance of TSP and solves the corresponding the TSP instance.

Experiments with heuristics that take into account service for backhauls as the constraints to solve TSPB are very limited. Gendreau et al. (1997) work on the Christofides TSP heuristic and prove that the algorithm results in 3/2 of the optimal solution in the worst case. Ghaziri and Osman (2003) used an artificial neural network to solve TSPB. They report results comparable to those of solution methods that transform TSPB to TSP.

Gendreau et al (1996) use the GENIUS heuristic, which is designed for TSP, to solve TSPB. The best configuration experimented by Gendreau et al. (1996) deviates on average 3-4% from the lower bound. This configuration is based on the idea of modifying TSPB cost matrix to convert it to TSP (Chrisman, 1975). Chrisman (1975) subtracted a large number (10 times the longest distance in the cost matrix) from the inter-cluster distances, leaving the intra-cluster distances the same. The triangle

inequality was still valid and the symmetry of the cost matrix was not disturbed. Chrisman (1975) reported optimal results with the modified cost matrix, compared to the constrained problem with side constraints.

In this study, we modify TSPB cost matrix to solve TSPB as a TSP using the GA developed in Chapter 3. The only study that uses a GA to solve CTSP is conducted by Potvin and Guertin (1996). Potvin and Guertin (1996) use ERX as the crossover operator and 2-opt as the mutation operator. They use ERX in two phases, the inter-cluster edges are preserved in the first phase, and the intra-cluster edges are preserved in the second phase. 2-opt mutation is applied within clusters. The results are reported to be better than GENIUS. To the best of our knowledge, the first study solving TSPB with a GA is the study conducted by Demir (2004). Our algorithm will initially be compared to that of Demir (2004), based on his problem instances.

Demir (2004) also use NNX crossover to solve TSPB without changing the distance matrix. His first approach is repairing infeasible solutions after they are generated by NNX. Secondly, he keeps generating children until a feasible child is obtained. His third strategy is to generate two paths (one for linehaul and one for backhaul customers) and connect them to obtain a feasible child. The fourth and fifth strategies are based on penalizing the infeasible solutions. The penalty in the first case is based on the cost that will be incurred to repair the infeasible tour. The second is based on increasing the penalty over the iterations, forcing the algorithm to generate feasible solutions. The algorithm that penalizes the cost of repair gives better results compared to the results of the algorithm that increases the penalty incrementally. The best results in terms of percent deviation are observed using the repairing strategy.

We have added a very large number (one million) to the inter-cluster distances in the distance matrix to solve TSPB with our GA. A sample modification of the matrix is demonstrated in Demir (2004). Other than this, we have not imposed any explicit constraints to make sure that all linehaul customers are visited before backhaul customers. The results of the GA can be seen in Table 4.2 for small test problems. The first column of the table gives the problem values number assigned by Demir (2004). The second and the third columns list the problem size and the optimal solution found by Demir (2004) calculated using the modified distance matrix in CONCORDE (Cook, 2007). The fourth column gives the average number of generations, which can be 10,000 at most. The maximum number of generations hardly reaches 10,000 since the

problems sizes are small and the algorithm is fully converged (i.e. the population best is equal to the population average) in the most of replications for 18 problems. “Best” represents the best solution value among the results of 30 replications. “Avg” represents the average of the best solution values. The population average (Pop Avg) is also reported in the Table, as it is a good indicator of the convergence of population. Table 4.2 shows that 10 out of 20 problems are solved to optimality in all 30 runs, and three problems are never solved to optimality by the algorithm. The average deviation for all 20 problems is 0.18%.

Table 4.1 compares our results with the results reported by Demir (2004). The results of the best strategy of Demir (2004) and the corresponding CPU times are reported in Table 4.1. Demir use no mutation operators and use NNX by imposing side constraints to solve the problem. Our results are reported in two columns, the first one is the pure NNX (indicated by “No Mut”), and the second column stands for the final algorithm where REM and CIM are applied with equal probabilities (indicated by “Mut”). Both pure NNX and NNX with mutations are reported to make clear the contribution of mutations. Note that we modified the distance matrix instead of imposing side constraints.

Demir (2004) did not report the results at 10,000th iteration, but when his algorithm is fully converged. Our results are 90% better compared to those of Demir (2004). Moreover, the CPU times are improved 30% on average. CPU times are expected to be equivalent as Demir (2004) used a Pentium 4 1.6 GHz processor and we employed an AMD Turion with dual 1.6 GHz processors where only one of the processors was fully utilized.

Table 4.1 Results for small TSPB compared with the results of Demir (2004)

Gen	Best Deviation (%)		Avg Deviation (%)		Pop Avg Deviation (%)		Worst* Deviation (%)		CPU Time		Demir's Avg Dev (%)	Demir's CPU Time
	No Mut	Mut	No Mut	Mut	No Mut	Mut	No Mut	Mut	No Mut	Mut		
2000	1.24	0.09	3.51	0.21	4.02	0.38	4.93	1.26	0.49	1.39	2.16	2.52
4000	0.79	0.07	3.03	0.18	3.10	0.28	3.37	1.11	2.34	3.87		
5000	0.79	0.07	3.06	0.19	3.08	0.28	3.31	1.09	3.48	5.07	1.83	6.3
80000	0.79	0.07	3.01	0.18	3.03	0.27	3.24	1.02	6.79	8.74	1.81	10.09
10000	0.79	0.07	3.01	0.18	3.03	0.27	3.24	1.00	9.06	11.17	1.79†	17.3†

* Average of the worst member in the population in 30 replications

† The results are averages at full convergence (17,133.26 generations)

Table 4.2 Results on of the GA for small TSPB problems

	Problem Size	Optimal	# of Gen[1]	Best[2]	Avg[3]	Pop Avg[4]	Best Deviation (%)	Avg Deviation (%)	Pop Avg Deviation (%)	CPU Time
p00	22	385	69.37	385	385	385	0.00	0.00	0.00	0.00
p01	33	580	492.50	580	580	580	0.00	0.00	0.00	0.20
p02	51	589	9678.13	589	589	590	0.00	0.00	0.14	8.93
p03	101	808	4635.20	808	810	811	0.00	0.26	0.32	7.43
p04	45	904	668.50	904	904	904	0.00	0.01	0.02	0.27
p05	121	870	8702.43	870	870	870	0.00	0.00	0.02	20.80
p06	30	123	111.63	124	124	124	0.81	0.81	0.81	0.00
p07	20	68	49.50	68	68	68	0.00	0.00	0.00	0.00
p08	30	128	111.14	128	128	128	0.00	0.00	0.00	0.00
p09	25	6244	84.65	6244	6247	6247	0.00	0.05	0.05	0.00
p10	23	652	2385.33	652	652	652	0.00	0.00	0.03	1.07
p11	30	530	110.00	530	530	530	0.00	0.00	0.00	0.00
p12	151	1067	10000.00	1071	1081	1085	0.37	1.28	1.70	35.23
p13	76	779	9477.50	779	780	781	0.00	0.09	0.26	14.87
p14	72	321	1829.07	321	321	321	0.00	0.00	0.00	1.63
p15	135	1151	10000.00	1154	1154	1155	0.26	0.29	0.32	25.47
p16	101	859	5398.27	859	860	860	0.00	0.14	0.15	8.23
p17	48	45697	9361.70	45697	45902	45963	0.00	0.45	0.58	8.20
p18	34	659	9343.63	659	660	664	0.00	0.11	0.69	5.70
p19	36	477	6422.53	477	477	478	0.00	0.07	0.21	3.93
Average		3144.55	4446.55	3144.95	3156.13	3156.13	0.07	0.18	0.27	7.10

[1] Average of 30 replications

[2] Best of 30 replications

[3] Average of the bests of 30 replications

[4] Average of the population averages in 30 replications

Table 4.1 also includes the population averages and the averages of the worst solution in the population in 30 replications. These figures are included to demonstrate that even the averages of worst solutions in the populations are better compared to Demir's results when mutation operators are employed.

Using pure NNX with modified cost matrix does not give better results compared to the strategies proposed by Demir (2004). The mutation operators reduce the deviation by 95% with a 23% increase in the CPU Time.

TSPB does not have a well-defined benchmark problem set. The second set of TSPB instances that have been solved with our GA is generated randomly based on the method proposed by Gendreau et al. (1996). The same method is used by Mladenović and Hansen (1997) and Ghaziri and Osman (2003), thus the results of these studies are comparable in terms of the averages over problem instances. 750 test instances are generated as follows. The coordinates of customers and depots are generated in the interval [0,100] assuming continuous uniform distribution. The problems size (n) is the first decision parameter and the ratio of backhaul customers to the number of linehaul customers (p) is the second decision parameter. 30 instances are generated for each pair (n, p) where n = 100, 200, 300, 500 and 1000 and p = 0.1, 0.2, 0.3, 0.4 and 0.5. When p = 0.1, 10% of the total customers are assumed to be backhaul customers.

Table 4.3 presents the averages of 30 replications reported by different authors for randomly generated instances. GENI gives the results of the heuristic developed by Gendreau et al. (1992), while GENIUS gives the results when an improvement method is applied after GENI, and GENIUS-VNS is the variable neighborhood search procedure applied on GENIUS (Mladenovic and Hansen, 1997). VNS systematically tries to improve the results of GENIUS by searching the immediate neighborhood of the solution point. The neighborhood is formed by node exchange moves, a node is deleted from a tour and inserted at a point that improves the tour length. The next two columns are the results of a self-organizing feature map type neural network, SOFM. SOFM* corresponds to the solutions when the results of SOFM are improved with 2-opt (Ghaziri and Osman, 2003).The last two columns represent the results of NNX with LEM and CIM mutations. The GA is run for 10,000 generations for problems n= 100, 200, 300; for 20,000 generations for problems with n = 500 and 30,000 generations for problems with n =1000. The Best column in Table 4.3 represents the averages over 30 different problems, of the best replication among five different replications. The Avg

column represents the mean of the average results of 30 different problems and five different replications for each problem.

Table 4.3 Average solution values for randomly generated problems

n	p	GENI	GENIUS	GEN- VNS	SOFM	SOFM*	Best	Avg
100	0.1	1012.50	994.12	987.11	1043.56	996.13	994.18	994.57
	0.2	1068.70	1047.01	1044.66	1072.30	1052.71	1052.06	1052.30
	0.3	1109.66	1088.09	1085.34	1108.54	1092.07	1089.04	1089.85
	0.4	1125.63	1106.69	1102.29	1131.83	1106.97	1100.51	1101.02
	0.5	1133.87	1114.34	1108.68	1123.29	1112.37	1103.16	1104.31
Average		1090.07	1070.05	1065.62	1095.90	1072.05	1067.79	1068.41
200	0.1	1418.63	1387.22	1378.80	1436.12	1381.15	1381.17	1381.85
	0.2	1498.83	1470.95	1464.88	1489.91	1462.32	1467.57	1467.85
	0.3	1550.52	1525.26	1519.93	1545.00	1523.71	1502.41	1502.99
	0.4	1585.76	1555.26	1548.73	1554.69	1551.48	1521.97	1522.78
	0.5	1586.93	1554.13	1546.97	1553.90	1549.61	1534.64	1535.90
Average		1528.13	1498.56	1491.86	1515.92	1493.65	1481.55	1482.27
300	0.1	1720.82	1683.76	1675.82	1702.60	1680.93	1668.96	1671.28
	0.2	1824.62	1784.80	1782.62	1787.14	1784.90	1773.81	1774.43
	0.3	1886.48	1854.86	1849.05	1877.15	1854.30	1830.29	1832.32
	0.4	1903.29	1874.43	1865.75	1876.49	1866.84	1868.45	1869.54
	0.5	1927.34	1892.20	1887.35	1891.39	1888.92	1873.38	1874.08
Average		1852.51	1818.01	1812.12	1826.95	1815.18	1802.98	1804.33
500	0.1	2197.16	2158.79	2156.61	2168.59	2161.07	2139.06	2142.44
	0.2	2342.99	2297.11	2292.04	2310.70	2297.35	2279.95	2285.57
	0.3	2409.80	2370.45	2363.16	2398.49	2376.73	2342.80	2347.55
	0.4	2443.12	2399.35	2388.07	2441.94	2397.06	2392.52	2398.11
	0.5	2464.11	2418.20	2405.55	2428.72	2410.81	2405.93	2411.29
Average		2371.44	2328.78	2321.09	2349.69	2328.60	2312.05	2316.99
1000	0.1	3099.17	3042.60	3029.76	3083.59	3048.69	3035.92	3044.569
	0.2	3281.34	3232.65	3213.61	3279.02	3228.44	3212.79	3216.77
	0.3	3366.07	3314.80	3302.93	3327.04	3312.38	3323.03	3328.28
	0.4	3451.02	3387.43	3366.23	3392.17	3371.28	3358.36	3366.75
	0.5	3455.69	3388.16	3379.67	3408.41	3386.50	3398.27	3403.74
Average		3330.66	3273.13	3258.44	3298.05	3269.46	3265.68	3272.02
Overall		2034.56	1997.71	1989.82	2017.30	1995.79	1986.01	1987.54

Table 4.4 Average CPU times of TSPB solution methods

n	p	GENIUS	GEN - VNS	SOFM	SOFM*	Actual CPU Time
100	0.1	4.7	5.4	23.5	23.7	20.5
	0.2	4.8	5.8	22.1	22.8	18.5
	0.3	4.8	5.5	21.2	21.6	17.9
	0.4	5.1	5.4	27.6	28.0	17.3
	0.5	4.4	5.6	23.1	23.8	17.2
Average		4.8	5.5	23.5	24.0	18.3
200	0.1	36.3	31.6	61.1	61.7	38.0
	0.2	32.4	35.9	63.6	64.3	38.7
	0.3	31.7	30.7	71.3	71.9	35.8
	0.4	31.2	38.8	72.9	73.8	35.6
	0.5	39.6	43.1	62.1	62.6	35.2
Average		34.2	36.0	66.2	66.9	36.6
300	0.1	106.4	109.3	237.9	239.2	68.5
	0.2	105.9	87.2	278.3	283.6	65.4
	0.3	70.9	100.1	286.3	287.8	68.8
	0.4	69.6	105.6	365.0	371.0	68.7
	0.5	72.3	101.1	354.0	360.3	66.2
Average		85.0	100.7	304.3	308.4	67.5
500	0.1	325.6	343.6	732.0	749.7	317.4
	0.2	289.5	248.1	729.0	751.9	300.5
	0.3	317.7	383.3	798.0	821.5	321.4
	0.4	374.0	326.1	802.0	834.2	343.1
	0.5	405.9	472.2	852.0	872.0	295.4
Average		342.5	354.7	782.6	805.9	315.6
1000	0.1	1130.3	1417.9	1398	1428.1	1933.7
	0.2	1211.1	1637.2	1423	1495.3	1921.4
	0.3	1019.8	1643.1	1412	1432.1	1829.1
	0.4	1302.8	1898.3	1435	1470.5	1776.9
	0.5	1324.6	1762.6	1402	1440	1832.9
Average		1197.7	1671.8	1414.0	1453.2	1858.8
Overall		332.9	433.7	518.1	531.7	459.4

The algorithms are coded in different languages and are run on computers with different processors. According to Ghaziri and Osman (2003), GENI, GENIUS, and GENIUS-VNS are coded in C++ and run on a SUN SPARC 10 computer by Mladenovic and Hansen (1997). SOFM and SOFM* are coded in FORTRAN and run on Intel Pentium MMX 233 MHz PC. Ghaziri and Osman (2003) used the benchmark results of Dongarra (2007) to calculate the relative computing time if they had utilized SUN SPARC 10. Our GA was coded in C and run on a PC AMD Turion 64 1.6 Ghz PC, and the actual CPU times are reported in the last column of Table 4.4. Choi and Tcha (2005) state that performance comparison in terms of CPU time is not always recommended, owing to other influencing factors such as cache, main memory and compilers have been subject to vast changes with the changes in the CPU architectures.

A paired t-test on the difference of the overall averages between Avg versus GENI, GENIUS, SOFM, SOFM* and GENIUS-VNS. The GA is significantly better than GENI (p-value = 0.000), GENIUS (p-value = 0.001), SOFM (p-value = 0.000) and SOFM* (p-value = 0.006). Results of the paired t-tests are given in Appendix F. The average results are better than GENIUS -VNS, but the difference is relatively small to derive a statistically significant result (p-value = 0.692). When the Best results and GENIUS-VNS's results are compared, p-value becomes 0.122.

When the problems with size $n \leq 500$ are considered, the test for the difference of overall averages between Avg (our GA) and GENI, GENIUS, SOFM, SOFM* and GENIUS-VNS indicates that our GA is statistically the best algorithm among all alternatives. The p-value for GENIUS-VNS and Avg (Best) comparison is 0.056 (0.011).

The results of the random TSPB instances indicate that our GA with LEM and CIM mutations gives better results compared to the competitors, except GENIUS-VNS. The application of GA is simple and the constraint handling is eliminated as the algorithm can effectively find good solutions just by making the necessary modifications in the distance matrix.

CHAPTER 5

CONCLUSION

In this research, a genetic algorithm (GA) that uses well-known TSP heuristics is developed and an application of this algorithm for solving TSPB is presented. The proposed GA uses Nearest Neighbor Crossover (NNX), which is based on the well known nearest neighbor heuristic. The mutation operators use 2-edge exchange and node insertion. Our aim is to investigate the performance of the mentioned crossover mechanisms when running with more than two parents and different child generation schemes, in order to develop a GA that can be used to solve TSP and its variants like TSPB in reasonable times.

NNX is a crossover operator available in the literature, which uses the nearest parental edge as long as it is feasible. The shortest edge in the complete graph is selected if no parental edge is feasible. NNX is deterministic as it concentrates only on the shortest edges. On the other hand, NNX is powerful as it is capable of adding new edges to the gene pool, as the algorithm is usually stacked when generating a child. About 70% of the offspring generated by NNX contain edges from the complete graph for a problem instance with 1000 cities until the 20,000th generation.

NNX is experimented with in a steady state fashion, where only two parents are considered for reproduction. The results of this study suggest that NNX gives better results compared to Sönmez (2003), when steady state evolution is used.

Different parent numbers are also experimented as NNX makes it possible to generate children that preserve the characteristics of more than two parents. Although more parents as large as twelve parents are experimented, our experiments show that using more than two parents brings in some improvement occasionally, but the impact is not very significant. Using two parents with NNX better balances the edges that are inherited from the parents and the ones taken from the complete graph.

The selection scheme that gives the best results with NNX is random selection when the child is generated from two parents and replaces their worst parents. Favoring the best individual does not improve the results, and replacement with worst population member causes a premature convergence of GA.

Random initial population gives the best results with NNX, and NNX cannot generate better solutions when NN is used as to initialize the population.

Enforcing NNX to take edges from alternating parents is experimented to avoid only one parent being very dominant during the generation of an offspring. The edge that follows an edge that is present only in one parent is enforced to be selected from the other parent or the complete graph to increase the exploratory power of NNX. The results are not as good as the unrestricted NNX, while good edges are usually observed to be in chains and enforcing false moves in every other edge deteriorates the solution quality.

NNX is highly dependent on the initial starting point when a child is to be generated, similar to the NN heuristic. Therefore, NNX gives best results when more than one child is generated from a pair of parents and using different starting points. This is done by generating a number of children until there is an improvement, and changing the parent pair if no improvement is observed for a number of trials.

Three different mutation operators are used in this research. The first one (Longest Edge Mutation) concentrates on the longest edges and tries to eliminate 15 longest edges by 2-edge exchange moves. The success of this mutation is limited becuse the mutation consumes a great deal of computing time; it is highly deterministic in edge deletion. Random Edge Mutation, that deletes 15 randomly selected edges and reconnects the subtours by 2-edge exchange moves, shows a better performance.

Another mutation operator that is not limited to a single edge is the Cheapest Insertion Mutation where the insertion of a node that has been removed from the current tour is considered. This results in deletion of three edges and addition of three new edges.

Convergence of NNX is analyzed in detail on two replications of a problem instance with 1002 cities. It has been observed that NNX is capable of generating good alternative solutions in problem instances with clusters. The results of NNX for such problem instances are around 3% close to the optimal solution, but NNX is capable of finding only 75% of the edges that are present in the optimal solution. The ability of NNX to generate solutions 3% above optimum, although it misses 25% of the "optimal edges" implies the power to discover alternative solutions for clustered problems.

We have also demonstrated that it is possible to generate a child that is worse than both of its parents when it is entirely generated using the shortest edges in the parents. The myopic strategy that concentrates only on the length of the next edge makes NNX to pick longer edges at the final stages of the edge search. It has been observed that in some cases it is not possible to generate children better than their parents as long as no edge from the complete graph is introduced.

Our GA has been implemented on TSPB instances after the algorithm is finalized based on the above observations. The TSPB instances are converted to the symmetric TSP instances by adding a very large number to the inter-cluster distances. The GA can then solve the intra and inter cluster problems to obtain a feasible solution. TSPB is experimented on two different benchmark sets. To the best of our knowledge, there is no published work with results on TSPB solved using a GA; except the study is conducted by Demir (2004) in his master's thesis.

TSPB instances of Demir (2004) have been used, and it has been demonstrated that NNX with REM and CIM mutation operators gives better results compared to the results when side constraints are imposed. Demir (2004) actually introduced different penalty or generation schemes that imposed on the backhaul constraints instead of modifying the distances. Our GA with modified distances is demonstrated to give better results in a shorter time when the mutation operators are used.

TSPB instances that are generated as suggested by Gendreau et al. (1996) has been used as the second set. According to the results on these randomly generated problems, our GA is significantly better than four out of the five algorithms published in the literature. Moreover, it gives the best results on problem instances with less than 1000 cities, among all.

For future research, NNX can be further improved to become faster in solving large problems in shorter time. The exhaustive search for the shortest edge when the algorithm gets stuck during the child generation process can be improved. A k-nearest neighbor subgraph as used by Yang (1997) can be employed to improve the computational time.

Another research topic can be implementation of NNX with the mentioned mutation operators to vehicle routing problems (VRP), as the GA applications in the VRP domain usually use crossover operators that reserve position or order of the cities in the parents. It has been demonstrated that the crossover operators that preserve edges in the parents give significantly better results.

Another interesting research area is solving and comparing the results of TSPB problems with using different crossover operators such as EAX (Nagata, Kobayashi, 1997) or DPX (Freisleben, Merz, 1996).

Other conventional TSP heuristics can also be used to come up with different GA operators.

REFERENCES

Affenzeller, M. (2002). A generic evolutionary computation approach based upon genetic algorithms and evolution strategies. *Systems Science*, 28(2), 59–71.

Applegate, D. (2007). World traveling salesman problem. Georgia Institute of Technology. Retrieved August 3, 2007 from http://www.tsp.gatech.edu/world/.

Baraglia, R., Hidalgo, J.I. & Perego, R. (2001). A hybrid heuristic for the travelling salesman problem. *IEEE Transactions on Evolutionary Computation*, 5(6), 613 – 622.

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

Burkowski, F.J., (2003). Proximity and priority: applying a gene expression algorithm to the Traveling Salesperson Problem. *Parallel Computing*, 30 (5-6), 803-816.

Černy, V., (1985). Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. *J. Opt. Theory Appl*, 45, 41-51.

Chan, C., Lee, S.A., Kao, C.Y. & Tsai, H.K., (2005). Improving EAX with restricted 2-opt. *Proceedings of the 2005 conference on Genetic and Evolutionary Computation*, 1471 – 1476.

Chen, S. & Smith, S. (1999). Putting the "genetics" back into genetic algorithms (reconsidering the role of crossover in hybrid operators). *Foundations of Genetic Algorithms 5*, Morgan Kaufmann, 1999.

Chisman, J. A. (1975). The clustered traveling salesman problem. *Computers and Operations Research,* 2, 115–119.

Choi, E. & Tcha, D.W. (2005). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers and Operations Research*, 34 (7), 2080 – 2095.

Cohoon, J.P., Karro, J.E., Martin, W. N., Niebel, W.D. & Nagel, K., (1998). Perturbation method for probabilistic search for the traveling salesperson problem. *Applications and Science of Neural Networks, Fuzzy Systems, and Evolutionary Computation*, Proc. SPIE (3455) p. 118-127.

Colorni, A., Dorigo, M. & Maniezzo, V., (1991) Distributed optimization by ant colonies. Varela, F., Bourgine, P., (Ed.) *Proceedings of European Conference on Artificial Life*, Elsevier, Amsterdam.

Cook, W. (2007) *Concorde home*. Georgia Institute of Technology. Retrieved August 3, 2007 from http://www.tsp.gatech.edu/concorde.html.

Črepinšek, M., Mernik, M. & Žummer, V., (2000). A metaevolutionaty approach for the travelling salesman problem. *22nd International Conference on Information Technology Interfaces* (ITI).

Dantzig, G., Fulkerson, D., Johnson, S. (1954). Solution of large-scale traveling-salesman problem. *Operations Research*, 2, 93 – 410.

Demir, E. (2004). *Analysis of Evolutionary Algorithms for Constrained Routings Problems*., MSc Thesis, Ankara, METU.

Dongarra, J. (2007) *Performance of Various Computers Using Standard Linear Equations Software*, University of Tennessee, Knoxville TN, 37996, Computer Science Technical Report Number CS - 89 – 85, Retrieved on August 3rd, 2007 from: http://www.netlib.org/benchmark/performance.ps.

Freisleben, B. & Merz, P. (1996). A genetic local search algorithm for solving symmetric and asymmetric traveling salesman problems. *Proceedings of IEEE International Conference on Evolutionary Computation*.

Fox, B. R. & McMahon, M.B. (1991), Genetic operators for sequencing problems, *Foundations of Genetic Algorithms*.

Gamboa, D., Rego, C. & Glover, F., (2006) Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. *Computers and Operations Research*, 33 (4).

Gendreau, G., Hertz, A. & Laporte, G. (1992). New insertion and post-optimization procedures for the traveling salesman problem. *Operations Research*, 40: 1086–1094.

Gendreau, G., Hertz, A. & Laporte, G. (1996). The traveling salesman problem with backhauls. *Computers and OperationsResearch*, 23, 501–508.

Gendreau, G., Laporte, G. & Hertz, A., (1997). An approximation algorithm for the traveling salesman problem with backhauls. *Operations Research*, 45 (4), 639 – 641.

Gendreau, M., Laporte, G. & Potvin, J. Y. (2001). Metaheuristics for the capacitated vrp. The Vehicle Routing Problem, Ed. Toth, P., Vigo, D., SIAM monographs on discrete mathematics. *Society of Industrial and Applied Mathematics. Philedelphia.* 129 – 154.

Ghaziri, H. & Osman, I.H. (2003). A neural network algorithm for the traveling salesman problem with backhauls. *Computers and Industrial Engineering*, 44(2), 267 - 281.

Glover, F., (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13, 533 – 549.

Goetschalckx M. & Jacobs-Blecha C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research,* 42, 39–51.

Goldberg, D. E. (1989). *Genetic Algorithms in Search*, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, MA.

Goldberg, D. E. & Deb, K (1990). A comparative analysis of selection schemes used in genetic algorithms. *Foundations of Genetic Algorithms*, 69 – 93.

Gorges-Schleuter, M. (1997). Asparagos96 and the travelling salesman problem, *Proceedings of IEEE International Conference on Evolutionary Computation*.

Gorges-Schleuter, M. (1997) On the power of evolutionary optimization at the example of ATSP and large TSP Problems. *European Conference on Artificial Life*.

Ho, S.Y. & Chen J.H., (2000). A GA-based systematic reasoning approach for solving travelingsalesman problems using an orthogonal array crossover. *Proceedings The Fourth International Conference/Exhibition on High Performance Computing in the Asia-Pacific Region*, 2 (2), 659 - 663.

Holland, J. (1975) *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Harbor, MI.

Homaifar, A. Guan, S. & Liepins, G.E. (1993). A new approach on the travelling salesman problem by genetic algorithms. *Proceedings of the 5th International Conference on Genetic Algorithms*, 460 – 466.

Hui, Y. Kang, L. Yan, Z. & Zou, X. (2003). Guiding genetic operators with immunology principle: a case study in TSP. *The 2003 Congress on Evolutionary Computation.*

Jog P., Suh J.Y. & Gucht D.V. (1989). The effects of population size, heuristic crossover and local improvement on genetic algorithms for traveling salesman problem. Schaffer D.J. San Mateo (Ed.), *Proceedings 36 of the Third International Conference of Genetic Algorithms.* Ca, Morgan Kaufmann Publishers, 110 – 115.

Johnson, D. S. & McGeoch, L. A. (1997). The traveling salesman problem: a case study in local optimization. *Local Search in Combinatorial Optimization*, E. H. L. Aarts and J. K. Lenstra (editors), John Wiley and Sons, Ltd. pp. 215 – 310.

Johnson, D.S. (2004). *8th DIMACS Implementation Challenge: The Traveling Salesman Problem.* Retrieved 10 August, 2007 from: http://www.research.att.com/~dsj/chtsp/index.html

Julstrom, B.A. (1995). Very greedy crossover in a genetic algorithm for the traveling salesman problem. Proceedings of the 1995 ACM Symposium on Applied Computing.

Jultsrom, B. A. (1998). Comparing decoding algorithms in a weigh-coded GA for TSP. *Proceedings of the 1998 ACM Sumposium on Applied Computing*, 313 – 317.

Julstrom, B.A.(1999) Coding TSP tours as permutations via an insertion heuristic. *Proceedings of the 1999 ACM symposium on Applied computing*. 297-301

Jung, S. & Moon, B.R., (2002) Toward minimal restriction of genetic encoding and crossovers for the two-dimensional Euclidean TSP. *IEEE transactions on evolutionary computation*, 6 (6).

Katayama, K., Hirabayashi, H. & Narihisa, H., (1999). Performance analysis for crossover operators of genetic algortihm. *Systems and Computer in Japan*, 30 (2), 20 – 30.

Katayama, K., Sakamoto, H. & Narihisa, H. (2000). The efficient irbid mutation genetic algorithm for the travelling salesman problem. *Mathematical and Computer Modelling*, (31), 197 – 203.

Kirkpatrick, S., Gelatt Jr., C. D. & Vecchi, M. P.,(1983) Optimization by Simulated Annealing. *Science*. 220 (4598), 671 – 680.

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

Lenstra, J. K. & Rinooy Kan, A. H. G. (1975). Some simple applications of the travelling salesman problem. *Operations Research Quarterly*, (26), 717-733. Pergamon Press.

Lin, S. (1965) Computer Solutions of the Traveling Salesman Problem, *Bell systems Technologies Journal*. 44,2245 – 2269.

Maekawa, K., Mori, N., Tamaki, H., Kita, H. & Nishikawa, Y. (1996). A genetic solution for the traveling salesman problem by means of a thermodynamical

selection rule. *Proceedings of IEEE International Conference on Evolutionary Computation*, 529 – 534.

Merz, P. & Freisleben, B. (1997). Genetic local search for tsp: new results. *Proceedings of IEEE International Conference on Evolutionary Computation*, 159 – 164.

Merz, P., (2002) A comparison of memetic recombination operators for the traveling salesman problem. *Proceedings of the Genetic and Evolutionary Computation Conference*, 472 - 479.

Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N. , Teller, A.H. & Teller E., (1953). Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6):1087-1092.

Michalewicz, Z. & Fogel, D.B. (2000). Michalewicz, Z. & Fogel, D.B (Eds.) *How to Solve It: Modern Heuristics*. Germany Springer-Verlag.

Mladenovic, N. & Hansen, P. (1997). Variable neighborhood search. *Computers and Operations Research*, 24, 1097–1100.

Mühlenbein, H. (1989). Parallel genetic algorithms, population genetics and combinatorial optimization. In Schaffer, J. (Ed.) *Proceedings on the Third International Conference on Genetic Algorithms*. pp. 416-421. Los Altos, CA: Morgan Kaufmann Publishers.

Nagata, Y. & Kobayashi, S. (1997) Edge assembly crossover: a high-power genetic algorithm for the traveling salesman problem. *Proceedings of the 7th International Conference on Genetic Algorithms*, 450-457.

Nagata, Y. & Kobayashi, S., (1999). An analysis of edge assembly crossover for the traveling salesman problem. *IEEE International Conference on Systems, Man, and Cybernetics*, 628 – 633.

Nagata, Y., (2004). Criteria for designing crossovers for TSP. *Congress on Evolutionary Computation*, 1465 – 1472

Nguyen, H.D., Yoshihara, I. & Yasunaga, M. (2000). Modified edge recombination operators of genetic algorithms for the traveling salesman problem. *IECON 2000 26th Annual Conference of the IEEE Industrial Electronics Society*. 2815-2820.

Osman, I. H. & Laporte, G., (1996). Metaheuristics: A bibliography. *Annals of Operations Research*, 63: 511-623.

Potvin, J. Y., & Guertin, F. (1996). The clustered traveling salesman problem: A genetic approach. (Eds.) Osman, I. H., Kelly, J. P., *Meta-heuristics theory, and applications* (pp. 619-631). Boston: Kluwers Academic Publishers.

Pullan, W. (2003). Adapting the genetic algorithm to the travelling salesman problem. *The 2003 Congress on Evolutionary Computation*, 1029 – 1035.

Punnen, A.P. (2002). The traveling Salesman Problem: Applications, Formulations and Variations. *The Traveling Salesman Problem and Its Variations* (Ed. Gutin, G., Punnen, A.P.) Dordrecht, Kluwer Academic Publishers.

Ray, S. S., Bandyopadhyay, S. & Pal, S. K., (2004). New operators of genetic algorithms for traveling salesman problem. *Proceedings of the 17th International Conference on Pattern Recognition*.

Ray, S. S., Bandyopadhyay, S. & Pal, S.K., (2005). New genetic operators for solving tsp: application to microarray gene ordering. *Lecture notes in computer science*, 3776, 605 – 610.

Reinelt, G (1996). *The trevelling salesman problem: computational solutions for tsp applications.* Goos, G., Hartmanis, J.. (Eds.) Lecture Notes in Computer Science Springer Verlag.

Reinelt, G. (2007). *TSPLIB*. Retrieved August 10th, 2007 from http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/

Rocha, M., Vilela C. & Neves, J. (2000) A study of order based genetic and evolutionary algorithms in combinatorial optimization problems. proceedings: intelligent problem solving. *Methodologies and Approaches: 13th International*

Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems. Lecture Notes in Computer Science. 601 – 611.

Ronald, S., Kirkby, S. & Eklund, P., (1997) Multi-chromosome mixed encodings for heterogeneous problems. *IEEE International Conference on Evolutionary Computation*.

Sipser, M. (1997). *Introduction to the Theory of Computation*. PWS-Kent, Belmont, California.

Smitt, L.J. & Amini, M.M. (1998). Performance characteristics of alternative genetic algorithmic approaches to the traveling salesman problem using path representation: An empirical study. *European Journal of Operational Research*, 108 (3), 551-570.

Soak, S.M. & Ahn, B.H. (2004). New genetic crossover operator for the tsp. *Artificial Intelligence and Soft Computing* - Lecture Notes in Computer Science, 481 – 485.

Sönmez, M. (2003). *An Evolutionary Approach to TSP: Crossover with Conventional Heuristics*, MSc Thesis, Ankara, METU.

Starkweather, T., McDaniel, S., Whitley, C., Mathias, K. & Whitley, D., (1991). A Comparison of Genetic Sequencing Operators, *Proceedings of the Fourth International Conference on Genetic Algorithms*, 69 – 76.

Stützle,T., Grün,A ., Linke, S. & Rüttger,M., (2000) A comparison of nature inspired heuristic on the traveling salesman problem, In Deb et al. (eds) *In Proceeding of PPSNVI, Sixth International Conference on Parallel Problem Solving from Nature*, 1917: 661-670

Tagawa, K., Kanzaki, Y., Okada, D., Inoue, K. & Haneda, H., (1998). A new metric function and harmonic crossover for symmetric and asymmetric traveling salesman problem. *IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on Evolutionary Computation Proceedings*, 822- 827.

Takenaka, Y. & Funabiki, N., (1998). An improved genetic algorithm using the convex hull for traveling salesman problem. *IEEE Transactions on Systems, Man, and Cybernetics.*

Ting, C.K., (2004). Improving edge recombination through alternate inheritance and greedy manner. *Lecture Notes in Computer Science*, 210 – 219.

Tsai, H.K., Yang, J.M., Tsai, Y.F. & Kao, C.Y., (2003). Heterogeneous selection genetic algorithm for traveling salesman problems. *Engineering Optimization*, 35 (3), 297 – 311.

Tsai, H.K., Yang, J.M., Tsai, Y.F. & Kao, C.Y., (2004a). An evolutionary approach for large traveling salesman problems. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 34 (4), 1718 – 1727.

Tsai, H.K., Yang, J.M., Tsai, Y.F. & Kao, C.Y. (2004b). Some issues of designing genetic algorithms for traveling salesman problems. *Soft Computing*, 8, 689–697.

Tsai, H.K., Yang, J.M., Tsai, Y.F. & Kao, C.Y. (2004c). An evolutionary approach for gene expression patterns. *IEEE Transactions on Information Technology in Biomedicine,* 8(2), 1089-7771.

Wang, L., Maciejewski, A.A., Siegel, H.J. & Roychowdhury, V.P. (1998). A comparative study of five parallel genetic algorithms using the traveling salesman problem. *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing*, IEEE.

Wang, Y.P., Li, Y. H. & Dang, C.Y.,(2004). A novel globally convergent hybrid evolutionary algorithm for traveling salesman problems. *Proceedings of the Third International Conference on Machine Learning and Cybernetics*, 2485-2489.

Wang, X.,C, Cui, D.W., Wan, D. S. & Wang, L. (2006). A novel genetic algorithm based on gene therapy theory. *Transactions of the Institute of Measurement and Control*, 28 (3), 253- 262.

Wang, Y., Han, L., Li, Y. & Zhao, S. (2006). A new encoding based genetic algorithm for the traveling salesman problem. *Engineering Optimization*, 38 (1), 1–13.

Waslaw, C., (2002). A multilevel approach to the travelling salesman problem. *Operations Research*, INFORMS, 862 – 877.

Watabe, H. & Kawaoka, T.,(2000). Application of multi-step GA to the traveling salesman problem. *Proceedings of Fourth International Conference on Knowledge-Based Intelligent Engineering Systems and Allied Technologies*.

White, C.M. & Yen, G. G. (2004), A hybrid evolutionary algorithm for traveling salesman problem. *CEC2004 Congress on Evolutionary Computation*. 1473 - 1378.

Whiteley, D. (1989). The GENITOR algorithm and selection pressure: why rank--based allocation of reproductive trials is best. *Proceedings of the Third International Conference on Genetic Algorithms*, San Mateo, California, USA, 116-121.

Whitley, D. Starkweather, T. & D'Ann Fuquay (1989). Scheduling problems and travelling salesman: the genetic edge recombination operator. Schaffer, J., (Ed.) *Proceedings on the Third International Conference on Genetic Algorithms.* 133-140. Los Altos, CA: Morgan Kaufmann Publishers.

Xiaoming, D., Runmin, Z., Rong, S., Rui, F. & Shao, H. (2002).Convergence properties of non-crossover genetic algorithm. *Proceedings of the 4~ World Congress on Intelligent Control and Automation*. 1822-1826.

Xuan, W., & Li, Y. (2005) Solving traveling salesman problem by using a local evolutionary algorithm. *IEEE International Conference on Granular Computing,* 318 – 321.

Yan, X. S., Li, H., Cai, Z. H. & Kang, L. S., (2005). Fast evolutionary algorithm for combinatorial optimization problems. *Proceedings of the Fourth International Conference On Machine Learning And Cybernetic,* 3288 -3292.

Yang, R. (1997). Solving large travelling salesman problems with small population. *Genetic Algorithms in Engineering Systems: Innovations and Applications*, Conference Publication No. 466. IEEE.

Yang, L. & Stacey, D. A. (2001) Solving the traveling salesman problem using the enhanced genetic algorithm. *Proceedings: Advances in Artificial Intelligence: 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, 307 – 316.

Zou, P., Zhou, Z., Chen, G, Yao, X. & Hefei, C.,(2004). A novel memetic algorithm with random multi-local-search: a case study of TSP. *Congress on Evolutionary Computation*, 2335 – 2340.

APPENDICES

A. DETAILED TABLES FOR GENETIC ALGORITHMS FOR TSP AND EXPERIMENT RESULTS

The tables and graphics of this appendix are provided in the CD appended at the back cover. The CD includes folders containing data of all the runs for initial experiments, initial experiments with alternating parents, further experiments, and the prots of the most common edges using in convergence analysis. A fine containing details of the GA applications on TSP electronically available in the literature is also given.

B. ANOVA RESULTS AND RESIDUAL PLOTS FOR INITIAL EXPERIMENTS

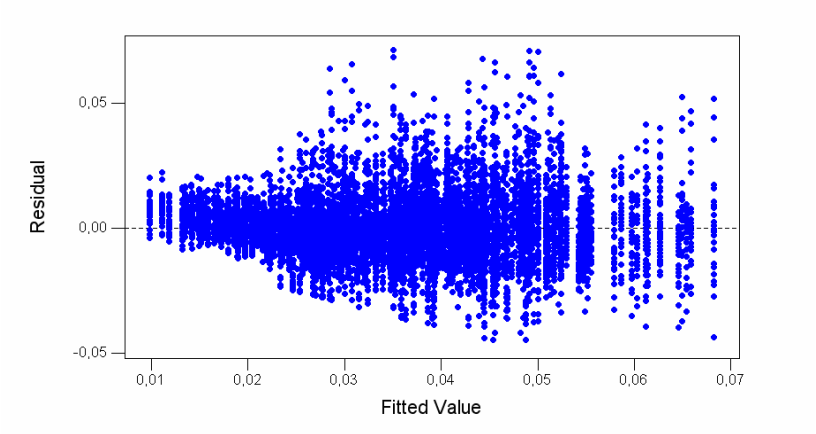
ANOVA Table for the Parameters
General Linear Model: Deviation versus Problem; Parents; ...

```
Factor     Type Levels Values
Problem    fixed      (8) 1 2 3 4 5 6 7 8
Parents    fixed      (3) 2 3 6
Children   fixed      (2)  1 10
Etha       fixed      (3) 1,0 1,5 2,0
Replc.     fixed      (2) 1 2
```

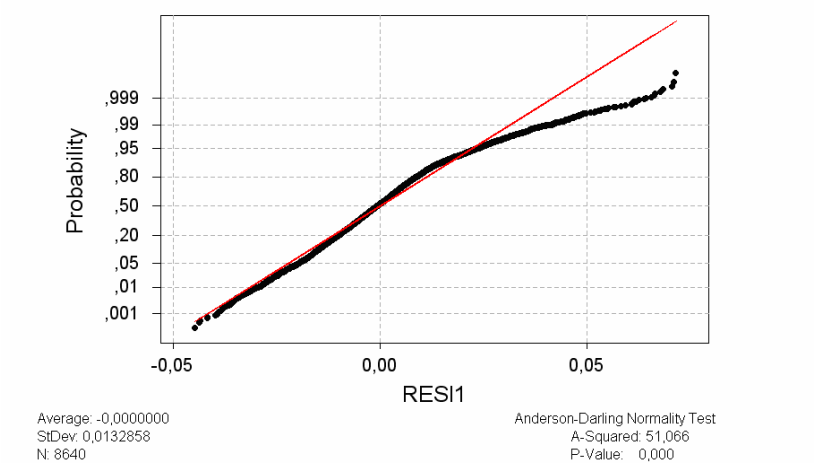
Analysis of Variance for Deviation, using Adjusted SS for Tests

Source	DF	Seq SS	Adj SS	Adj MS	F	P
Problem	7	0,647442	0,671209	0,095887	540,84	0,000
Parents	2	0,010072	0,013079	0,006540	36,89	0,000
Children	1	0,490198	0,468050	0,468050	2639,99	0,000
Etha	2	0,032811	0,030376	0,015188	85,67	0,000
Replc.	1	0,061018	0,061012	0,061012	344,13	0,000
Parents*Children	2	0,000545	0,000466	0,000233	1,31	0,269
Parents*Etha	4	0,001947	0,001690	0,000423	2,38	0,049
Parents*Replc.	2	0,020516	0,020515	0,010257	57,86	0,000
Children*Etha	2	0,004488	0,004047	0,002023	11,41	0,000
Children*Replc.	1	0,007121	0,007127	0,007127	40,20	0,000
Etha*Replc.	2	0,001875	0,001874	0,000937	5,28	0,005
Parents*Child*Etha	4	0,001274	0,001272	0,000318	1,79	0,127
Parents*Child*Repl	2	0,000828	0,000828	0,000414	2,34	0,097
Parents*Etha*Replc.	4	0,000757	0,000757	0,000189	1,07	0,371
Children*Etha*Replc.	2	0,000854	0,000854	0,000427	2,41	0,090
Error	8601	1,524891	1,524891	0,000177		
Total	8639	2,806638				

Residuals vs. Fits Plot

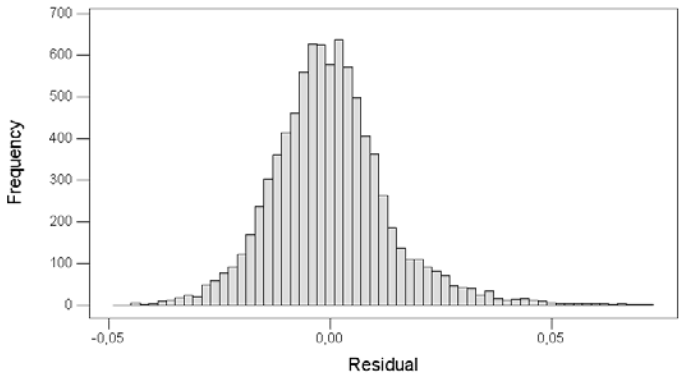


Normal Probability Plot of the Residuals



The residual and normality plots suggest tha the residuals show hetero-stochasticity and the assumptions of ANOVA cannot be verified. More detailed analysis with some transformation of the residuals in required.

Histogram of the Residuals



C. AN EXAMPLE OF CHILD GENERATION OF NNX

The edges of parent I and parent II are used to generate the child given in Figure D.1. The length of the tour represented by parent I is 287597 and the length of the tour represented by parent II is 286918. The child generated by NNX has a length of 287648. Figure D.1 represents both of the parent tours, and the child tour. The common edges on the parents are given in the color of parent II. It can be seen that the child is generated completely by the edges of the parents, thus trying to get the best edges in parents results in a length worse than both of the parents.

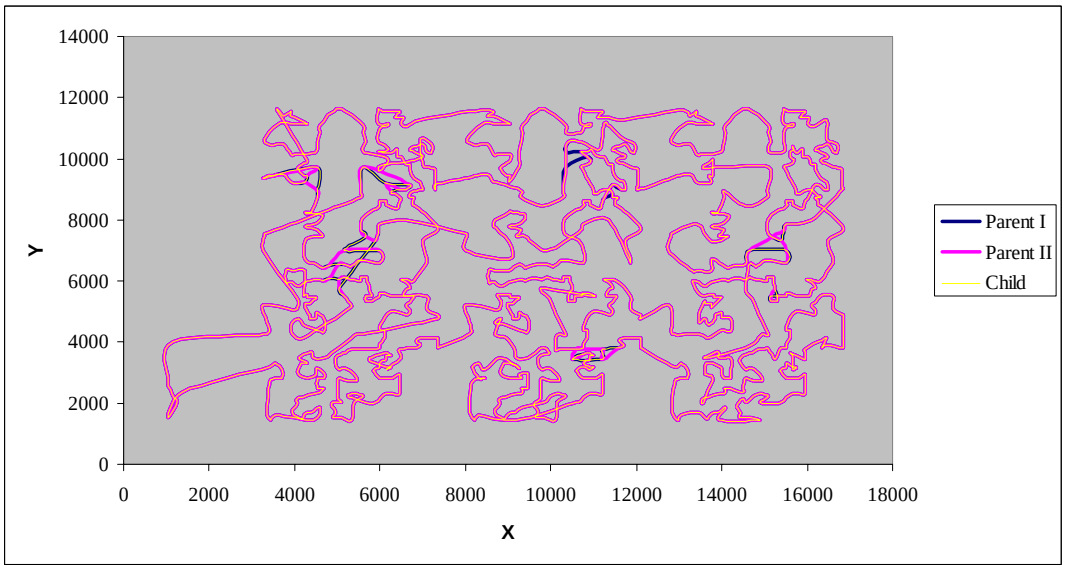


Figure C.1 Plot of the parent I, parent II and child tours for pr1002

The length of the child generated is highly dependent on the starting node when the child is being constructed from the union graph. When this specific parent pair is considered, 159 out of the possible 1002 children does not take any edge from the complete graph. None of these 159 children are better than any of the parents while 99 out of these 159 have a tour longer than both of the parents. 92 out of possible 1002 children result in a tour shorter than both of the parents, all taking edges from the complete graph.

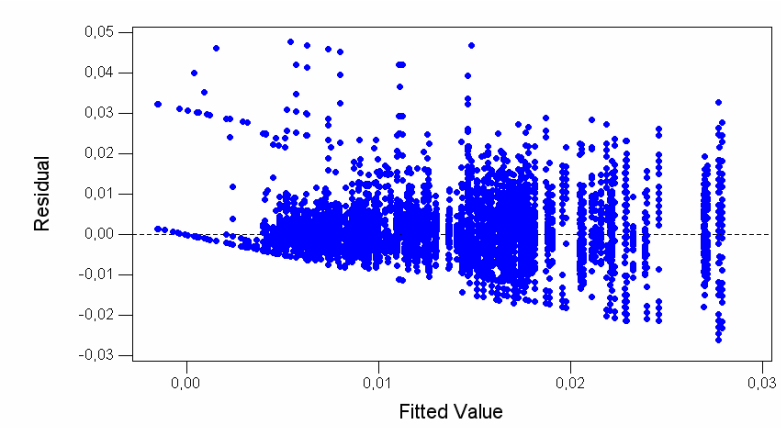
D. ANOVA RESULTS AND RESIDUAL PLOTS FOR FURTHER EXPERIMENTS

The detailed results can be seen in the worksheets provided in the CD, at the back cover.

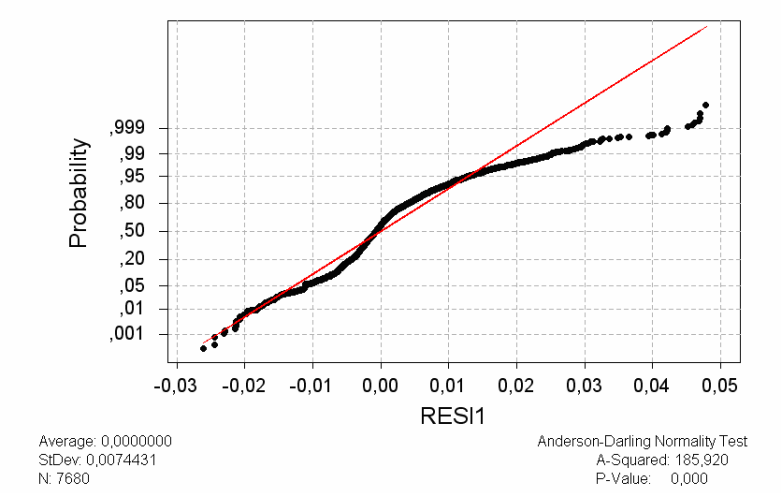
General Linear Model: Deviation versus File; Child; Mut 1; Mut 2; Replc

```
Factor     Type Levels Values
Problem    fixed      (8) 1 2 3 4 5 6 7 8
Child      fixed      (4) 2 4 6 12
Mut 1      fixed      (2) 1 2
Mut 2      fixed      (2) 1 2
Replc      fixed      (2) 1 2

Analysis of Variance for Deviatio, using Adjusted SS for Tests

Source               DF     Seq SS     Adj SS     Adj MS       F      P
File                  7  0,2223711  0,2223711  0,0317673  570,80  0,000
Child                 3  0,1031509  0,1031509  0,0343836  617,81  0,000
LEM                   1  0,0139632  0,0139632  0,0139632  250,89  0,000
CIM                   1  0,0001237  0,0001237  0,0001237    2,22  0,136
Replc                 1  0,0000000  0,0000000  0,0000000    0,00  0,985
Child * LEM           3  0,0029806  0,0029806  0,0009935   17,85  0,000
Child * CIM           3  0,0002519  0,0002519  0,0000840    1,51  0,210
Child * Replc         3  0,0001166  0,0001166  0,0000389    0,70  0,553
LEM * CIM             1  0,0000021  0,0000021  0,0000021    0,04  0,844
LEM * Replc           1  0,0001342  0,0001342  0,0001342    2,41  0,121
CIM * Replc           1  0,0000157  0,0000157  0,0000157    0,28  0,595
Child * LEM * CIM     3  0,0010367  0,0010367  0,0003456    6,21  0,000
Child * LEM * Replc   3  0,0000415  0,0000415  0,0000138    0,25  0,862
Child * CIM * Replc   3  0,0000333  0,0000333  0,0000111    0,20  0,897
LEM * CIM * Replc     1  0,0000003  0,0000003  0,0000003    0,00  0,946
Error              7644  0,4254173  0,4254173  0,0000557
Total              7679  0,7696391
```

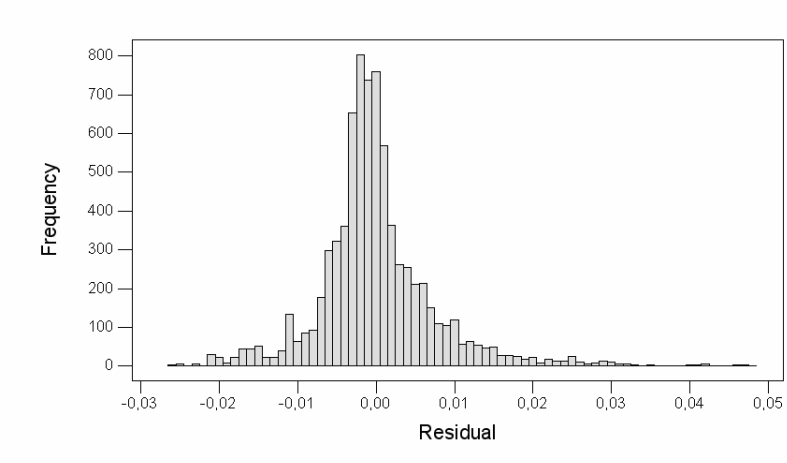
Residuals vs. Fits Plot



Normal Probability Plot of the Residuals



Histogram of the Residuals



E. CONVERGENCE ANALYSIS OF pr1002

This Appendix concentrates on the convergence behavior of the NNX on pr1002 The problem shows a pattern, where the same basic structure is repeated three times. Figure E.1 demonstrates the structure and the optimal solution of the problem.

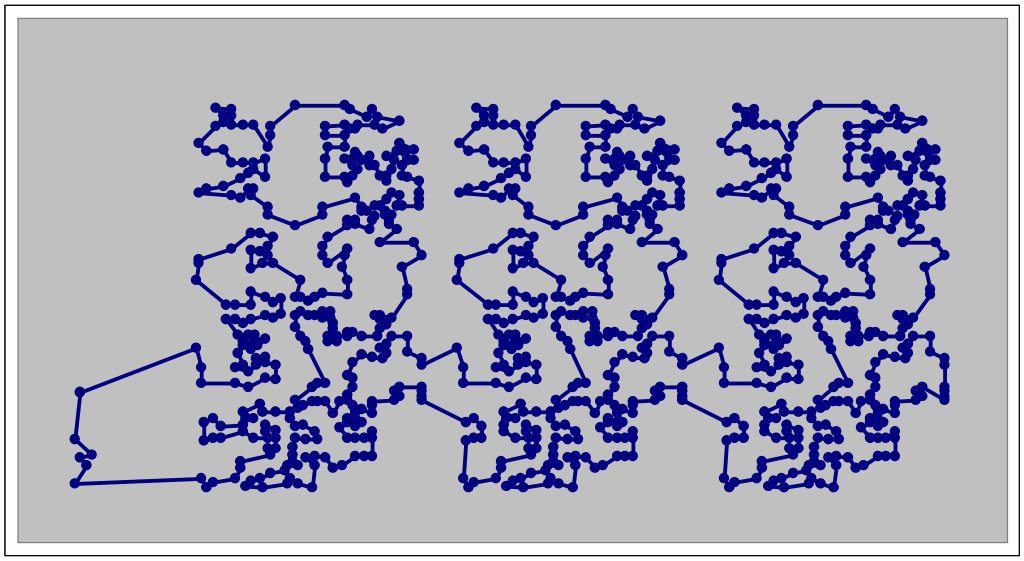


Figure E.1 Optimal solution

Two representative solutions are selected among 10 different replications results (Table 3.8). The first replication (S1) is a relatively poor solution; the second replication (S2) gives the results of an average solution when LEM and CIM is applied.

All of the figures in the Appendix represent the results obtained during 40,000 generations. These figures are based on the averages taken in each 100 generation interval (i.e. 400 different measurements are plotted in each figure).

The population size is kept as 200. The first section demonstrates the graphical results of percentage deviation for different settings of the EA. The percentage of individuals that contain edges taken from the complete graph and percentage of replacements per generation are reported with the average edge difference among individuals in the following section. The performance of mutations is demonstrated by the figures of

successive mutations in the next section E.3 and E.4. This appendix ends with percentages of successive mutation trials and percentages of optimal edges covered by the individuals.

Every section contains results of three different configurations for both of the replications. The fist one contains NNX without any mutation. The second contains NNX with LEM and CIM mutations applied with equal probabilities. The last one is REM and CIM combination again with equal probabilities. The results of LEM and REM are expected to give similar results as LEM is a special of REM and the edges these mutation operators bring in are similar.

E.1. Deviation from the Optimal

The percentage deviation of the best, average, and worst individuals in the population are plot with respect to the number of generations.

There is a sudden decrease in the first 2,000 generations (Figures E.2, E.4, E.6, E.8, E.10, E.12), so the convergences after 2,000th generations are plotted in separate Figures to ease detailed investigation. When no mutation is present, the algorithm is fully converged after 17,000th generation (Figures E.3, E.5); the population average is very close to population best after the 12,000th generation. When LEM and CIM pair is considered, the algorithm cannot be considered (Figures E.7, E.9) as converged at all, as there is a significant difference between the deviations of the best and average individuals at the end of the runs. Improvements are observed at 39,000th generation (Figure E.7). Moreover, 40,000 generations are larger than twice the number of generations required for the convergence of S1 without the mutation operators. With REM and CIM combination, the rate of population average convergence slows after 7000th generation, and gets faster again after 17,000th generation (Figures E.11, E.13). The population average converge the population best after 17,000th generation forming an S-shaped curve. There are minor improvements after 30,000th generation. The algorithm can be assumed as fully converged after 37,000th generations since the population average is very close to the population best. The REM and CIM combination converges at 40,000 iterations to a lower value compared to the pure NNX. LEM and CIM combination still has room for improvement, as the population average is not close to the population best.

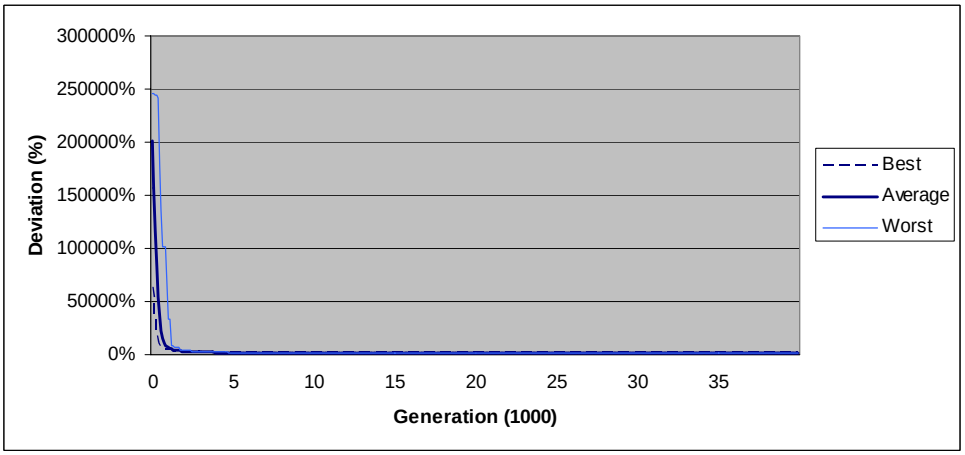


Figure E.2 Percent deviation vs. generations of pure NNX for S1

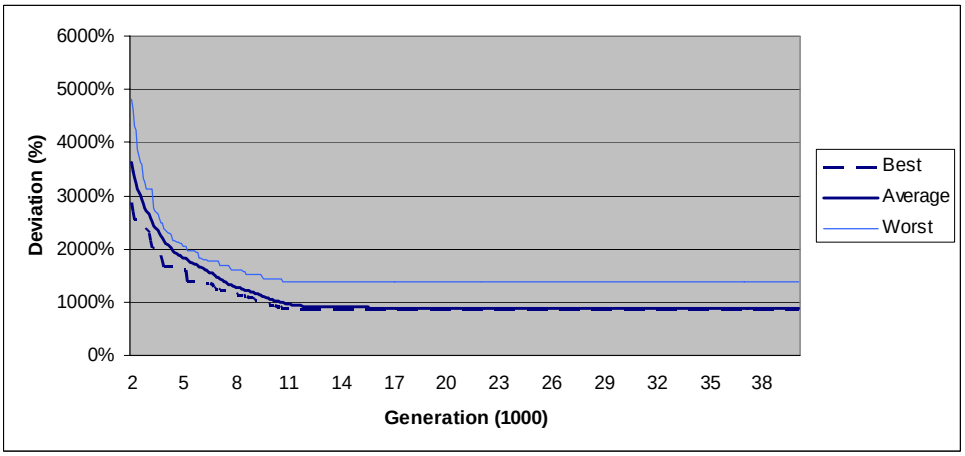


Figure E.3 Percent deviation vs. generations (after 2000) of pure NNX for S1

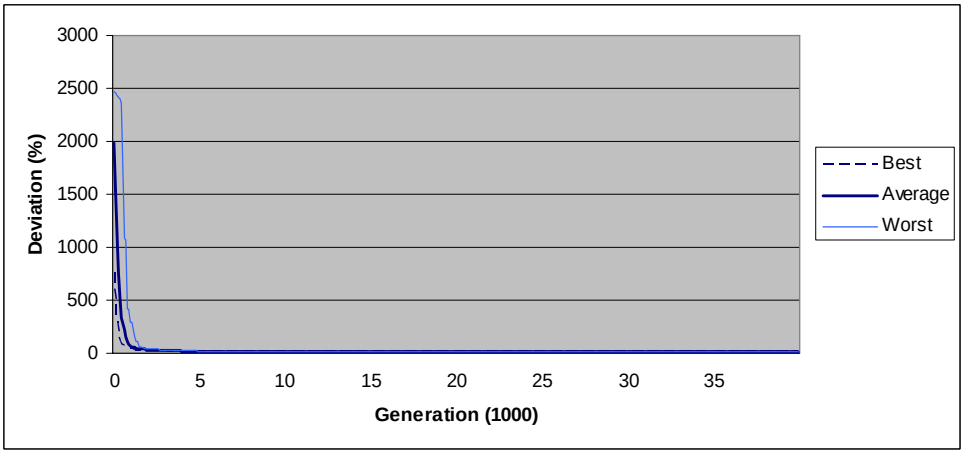


Figure E.4 Percent deviation vs. generations of pure NNX for S2

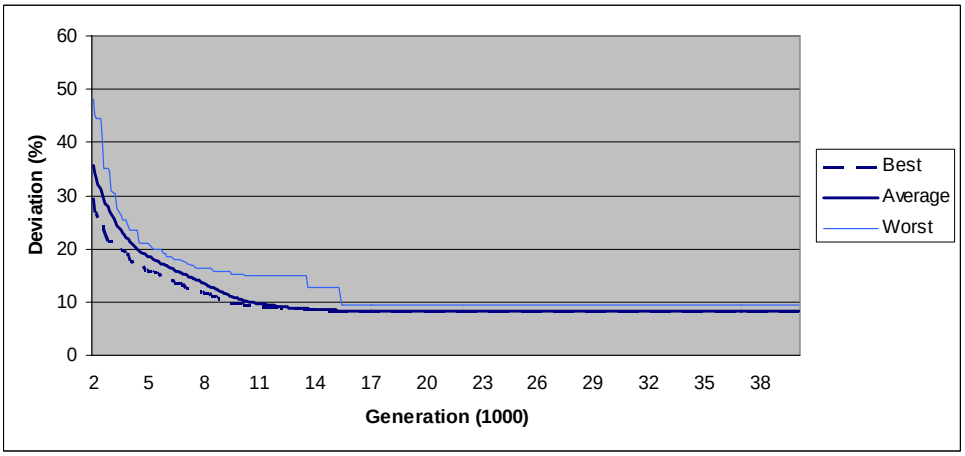


Figure E.5 Percent deviation vs. generations (after 2000) of pure NNX for S2

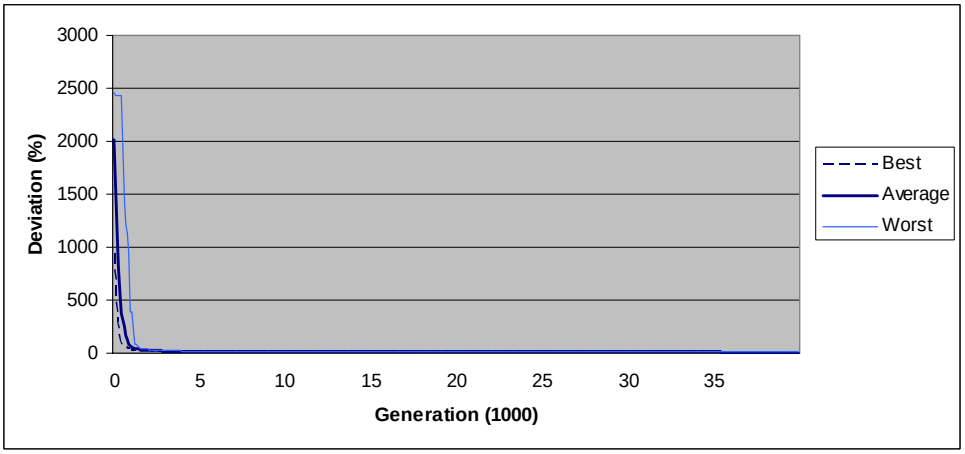


Figure E.6 Percent deviation vs. generations of LEM and CIM for S1

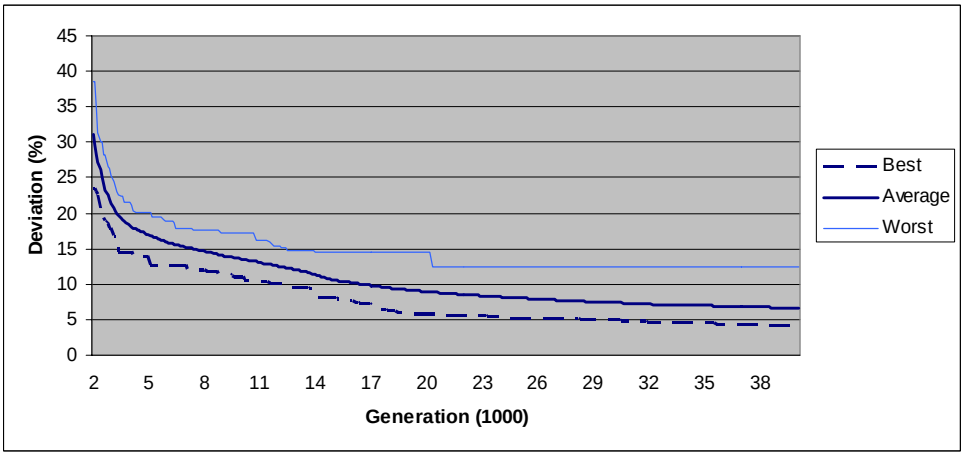


Figure E.7 Percent deviation vs. generations (after 2000) of LEM and CIM for S1

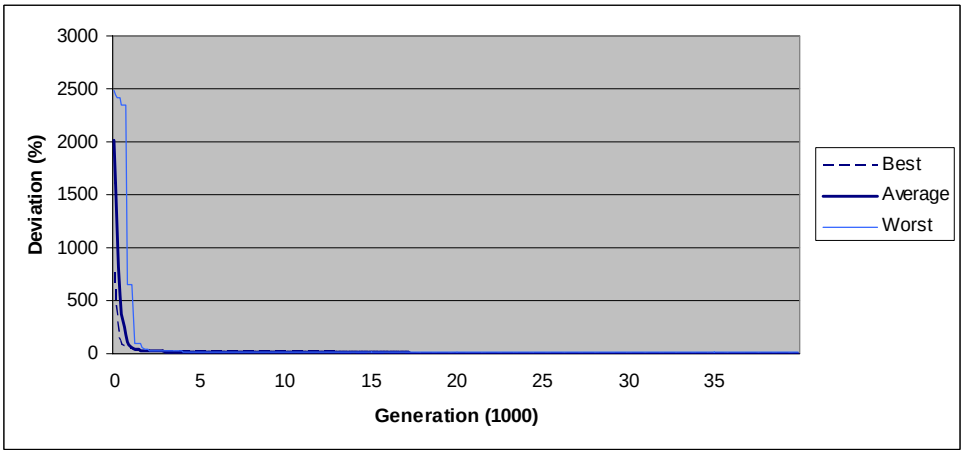


Figure E.8 Percent deviation vs. generations of LEM and CIM for S2

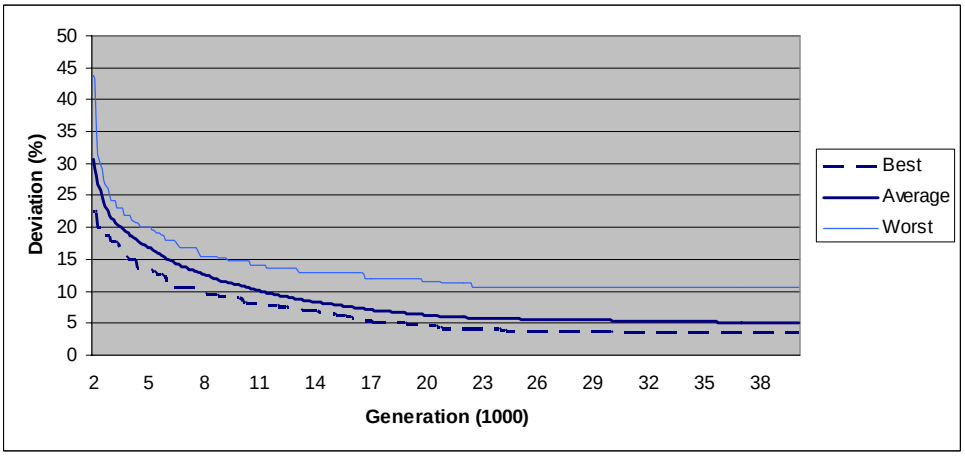


Figure E.9 Percent deviation vs. generations (after 2000) of LEM and CIM for S2

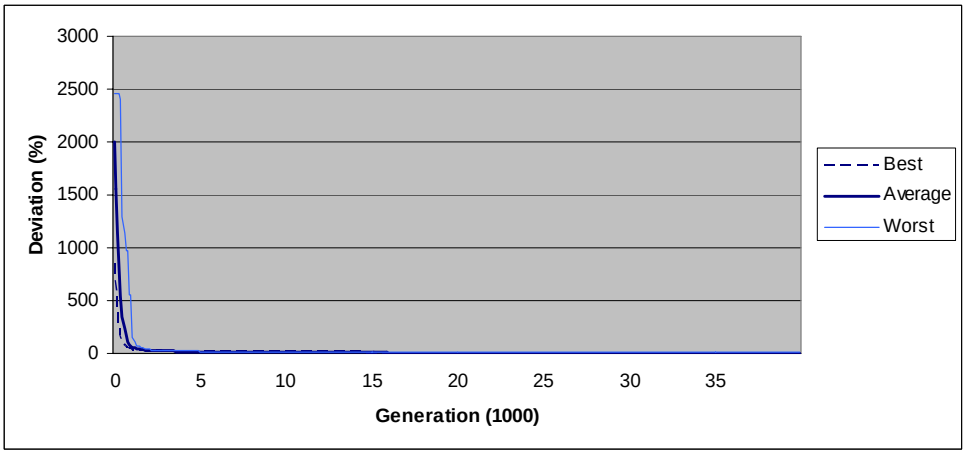


Figure E.10 Percent deviation vs. generations of REM and CIM for S1

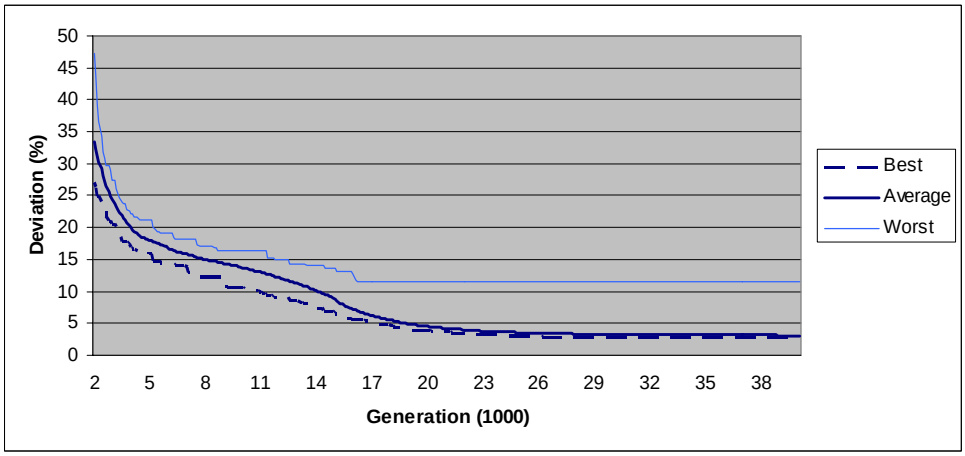


Figure E.11 Percent deviation vs. generations (after 2000) of REM and CIM for S1

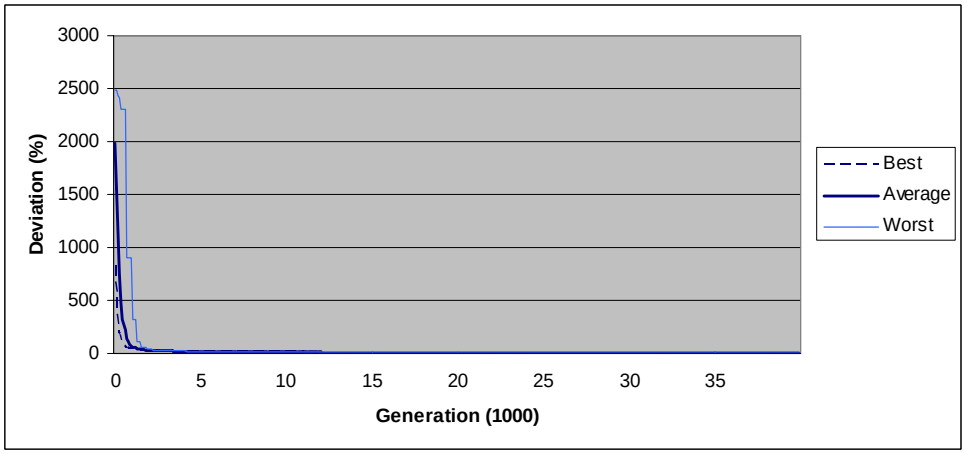


Figure E.12 Percent deviation vs. generations of REM and CIM for S2

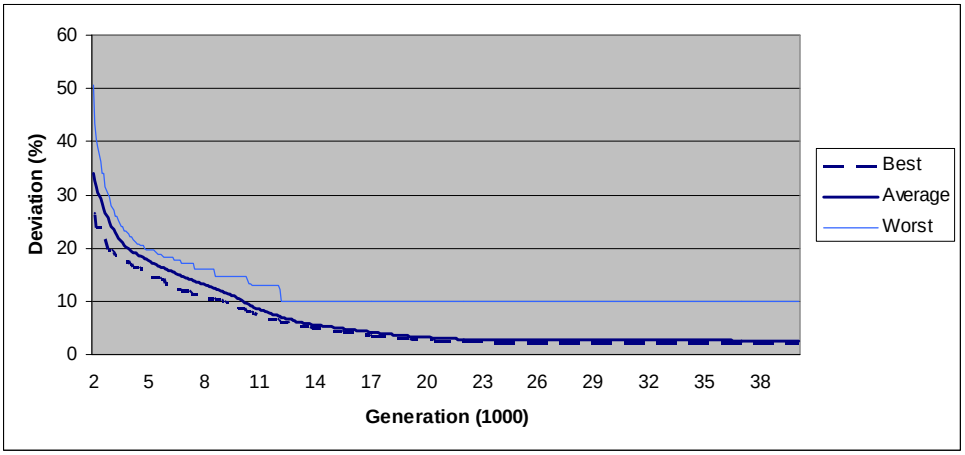


Figure E.13 Percent deviation vs. generations (after 2000) of REM and CIM for S2

E.2. New Edge Introduction and Replacement

Percentage of individuals that contain edges borrowed from complete graph and percentage of replacements can be seen in the following figures. The number of

replacements is the average number of replacements per generation, calculated in every 100 generations. The percentages of replacement an be used to explain the convergence of population average to the best solution, while edges borrowed from complete displays the exploratory power of that NNX configuration.

In pure NNX (Figures E.14, E.15), the replacement ratio is very high at the initial stages of the algorithm where the very bad edges that are generated randomly in the initial population are eliminated. There is a decrease between 5,000th and 10,000th in the percentage of replacements. At this stage, it becomes relatively hard to find good individuals using the edges present in the individuals. When an individual with relatively superior fitness is found, the replacement ratio increases again. This penetration of the good edges generated by NNX to the whole population causes the increase in the average number of replacement per generation.

With LEM and CIM combination (Figures E.16, E.17), the replacement does not increase back to 100% following a drop after the 5,000th generation. Replacement continues at a rate around 15%. The convergence of the population is spread in time, thus the edges from complete graph continue to come up as long as the algorithm continues.

REM and CIM (Figures E.18, E.19) combination preserves the edge replacement characteristic of pure NNX as the average number of replacements per generation decreases after 5,000th generation, but average increases back after 15,000th generation until 30,000th generation (Figure E.19). The replacement ratio never reaches 100% again, but the second peak is obvious. S2 results in two peaks after the initial decrease in percentage of replacements. It can be seen that the population average converges the population best in the period of the second peak (between 15,000th and 20,000th generations) from Figure E.13. These peaks correspond to the points where the number of most popular edges in the population changes according to the results of Figure E.41. There are edges borrowed from the complete graph until 17,000th generation with REM and CIM (Figure E.19), which ended at 12,000th generation in no mutation case (Figure E.15).

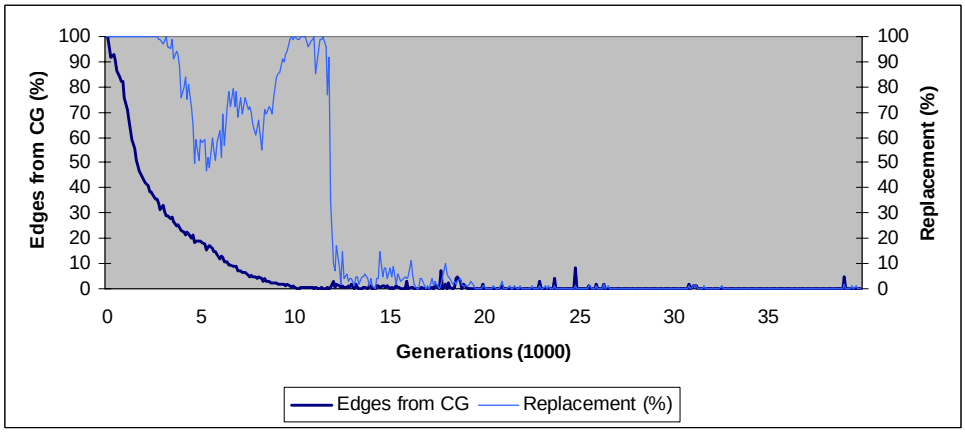


Figure E.14 Individuals that contain edges from complete graph and average number of replacements over generations for S1 with pure NNX

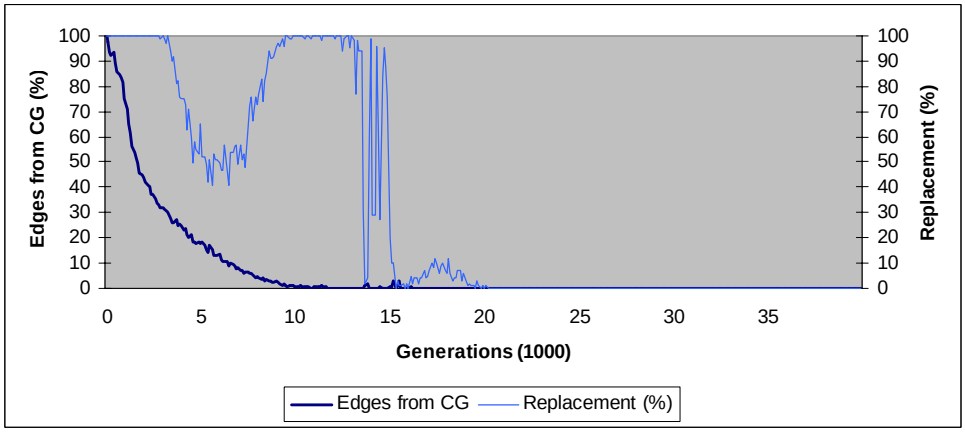


Figure E.15 Individuals that contain edges from complete graph and average number of replacements over generations for S2 with pure NNX

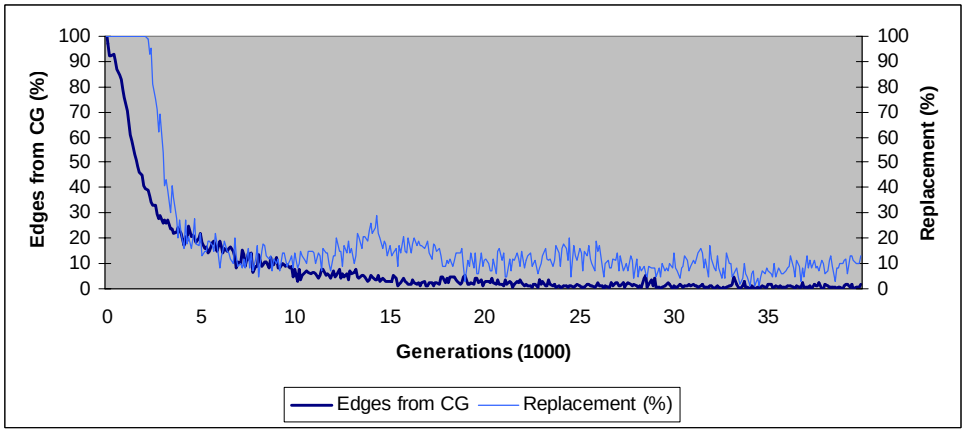


Figure E.16 Individuals that contain edges from complete graph and average number of replacements over generations for S1 with LEM and CIM

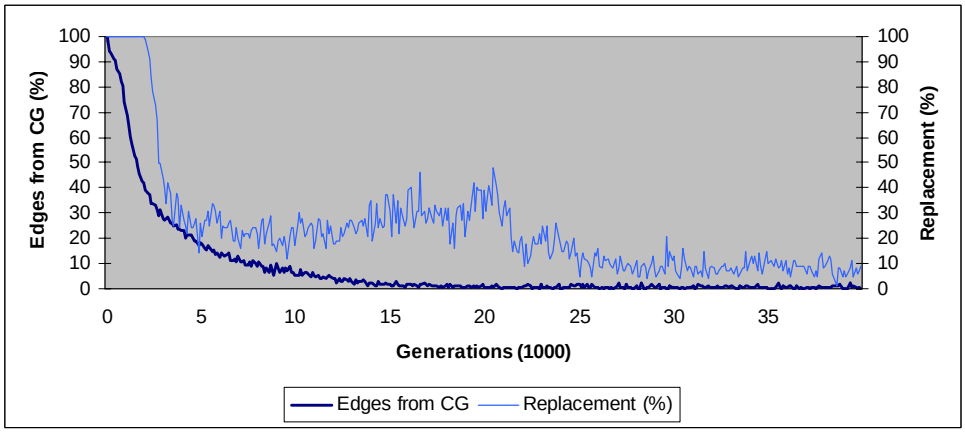


Figure E.17 Individuals that contain edges from complete graph and average number of replacements over generations for S2 with LEM and CIM

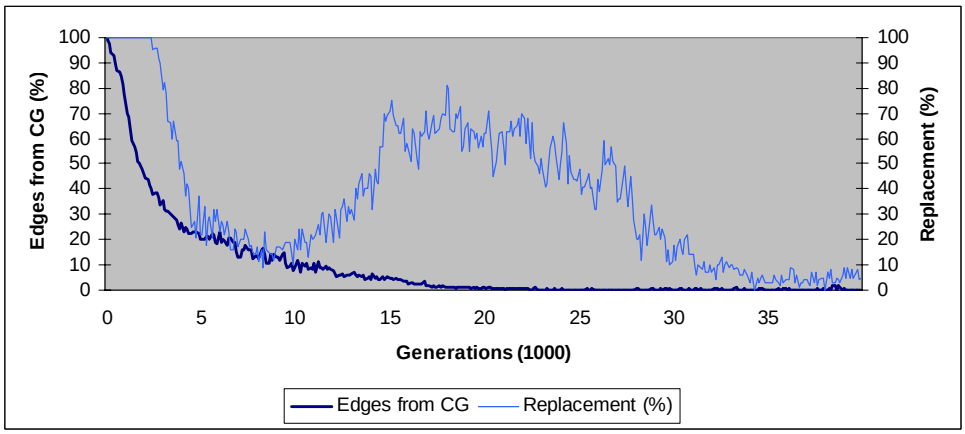


Figure E.18 Individuals that contain edges from complete graph and average number of replacements over generations for S1 with REM and CIM

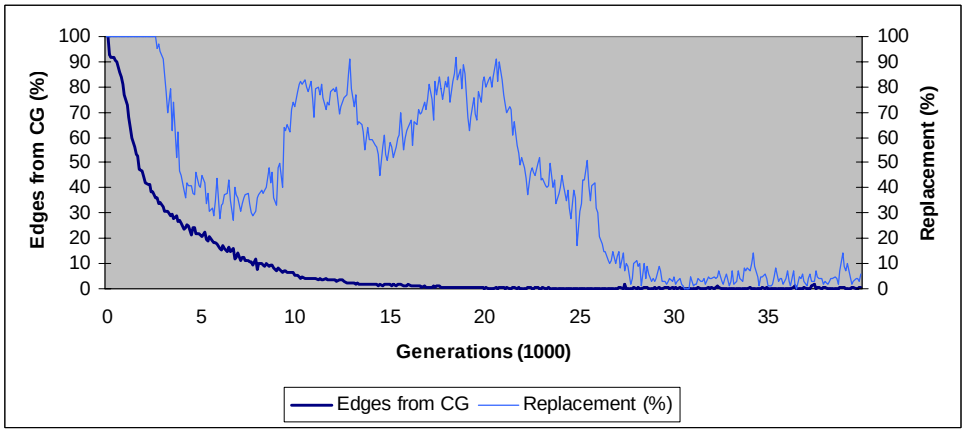


Figure E.19 Individuals that contain edges from complete graph and average number of replacements over generations for S2 with REM and CIM

E.3. Node Insertion of CIM

The nodes inserted by CIM per generation are plotted in the Figures C.20 – 23. CIM is experimented with LEM and REM. The nodes inserted by LEM and CIM (Figures E.20, E.21) combination vary around 0.5 per replacement and 0.01 per generation after the 5,000th generation. When REM is used with CIM (Figures E.22, E.23), the nodes inserted per generation loose importance after 25,000th generation, and increase significantly again after 32,500th iteration. Nodes inserted per generation are similar when both LEM and REM are used.

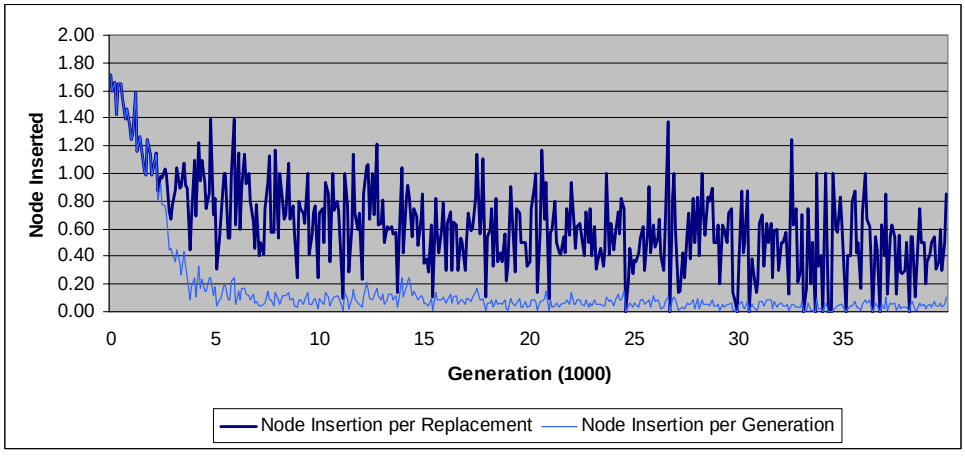


Figure E.20 Nodes inserted using CIM with LEM for S1

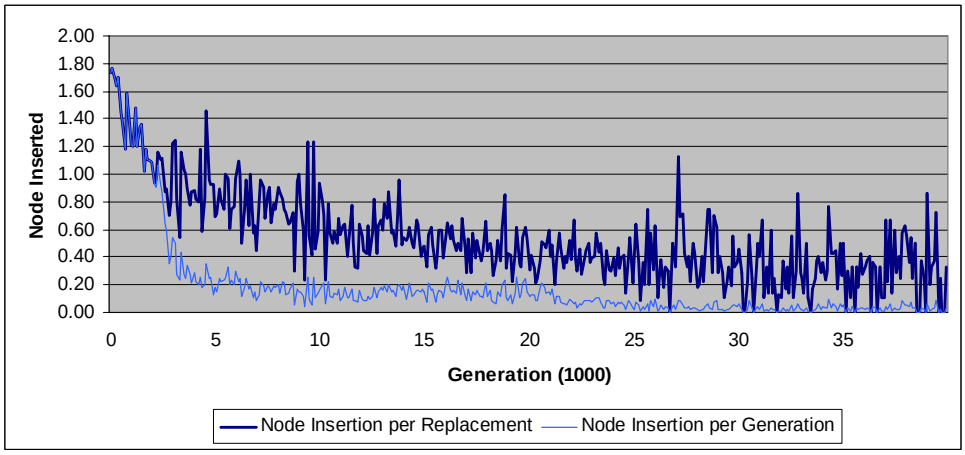


Figure E.21 Nodes inserted using CIM with LEM for S2

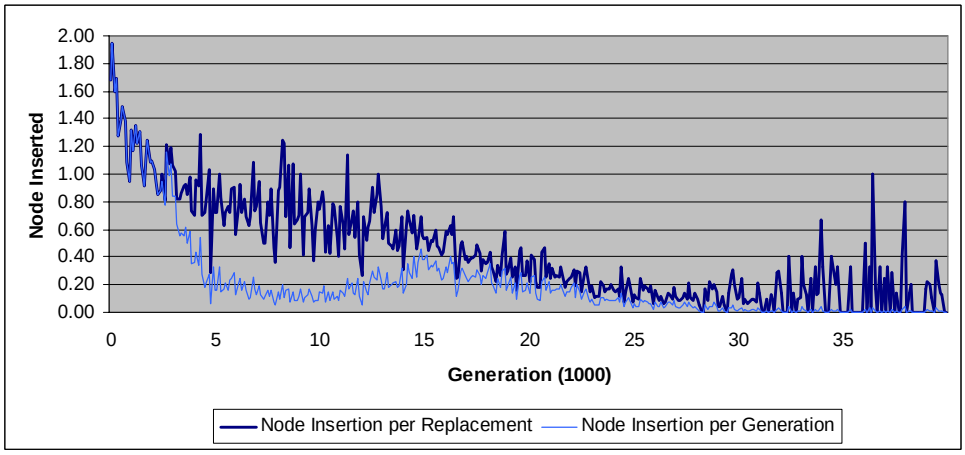


Figure E.22 Nodes inserted using CIM with REM for S1

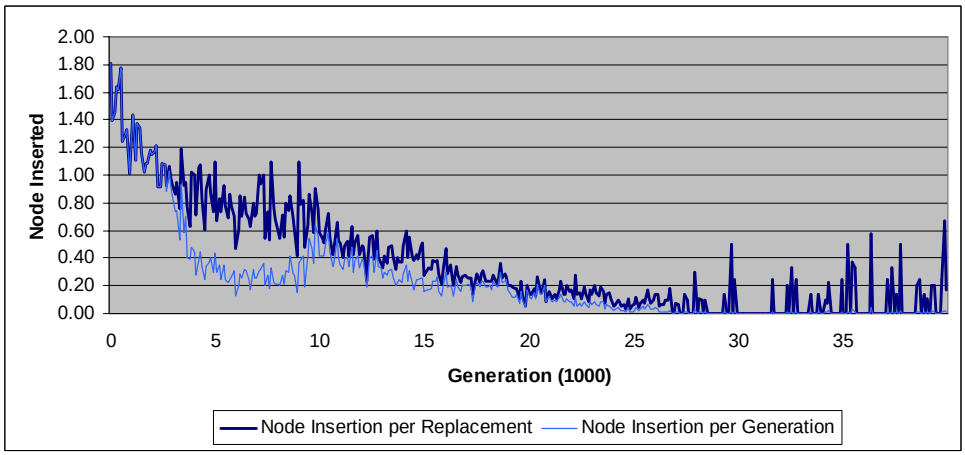


Figure E.23 Nodes inserted using CIM with REM for S2

E.4. Edge Exchanges

Edges exchanged are plotted in Figures E. 24 – 25. LEM and REM are both combined with CIM in all alternatives. The number of edges exchanged per replacement using LEM (Figures E.24, E.25) varies around 1, and on average 0.1 edge is replaced in each generation. On the other hand, the number of edges exchanged per replacement varies around 2.5 until 12,000th generation when REM is used (Figures E.26, E.27). After 12,000th generation, the average number of edges exchanged per replacement decreases to 0.5. Average number of edges inserted per replacement looses importance after 20,000th generation. The figures suggest that concentrating on the longest edge limits the power of edge exchange. The REM has a higher number of edges exchanged per generation and the deviation from the optimal in the resulting population is less when REM is used.

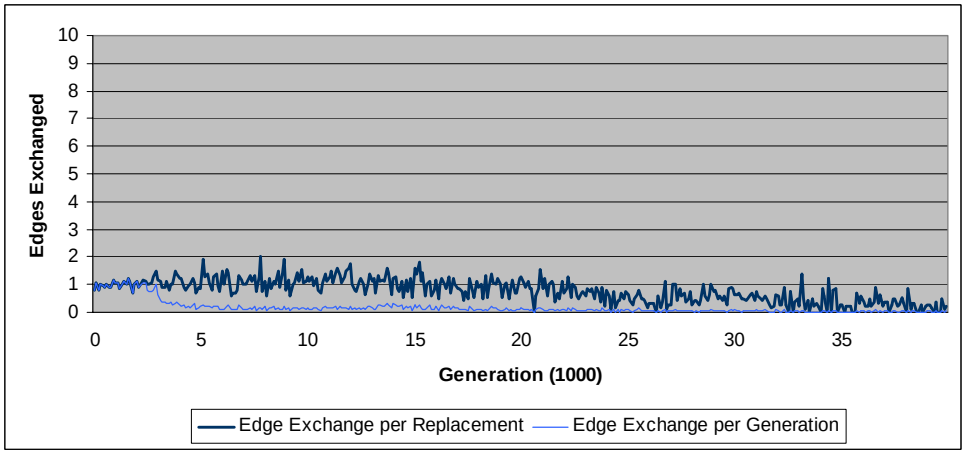


Figure E.24 Edges exchanged by LEM for S1

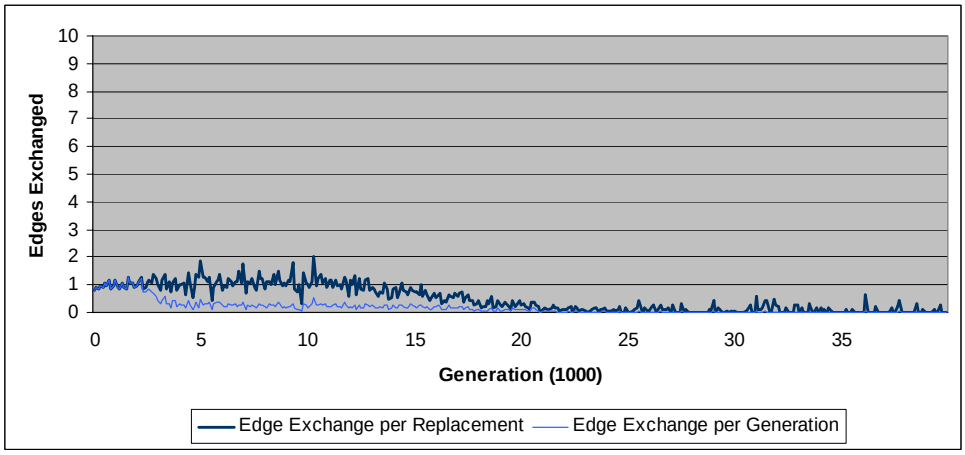


Figure E.25 Edges exchanged by LEM for S2

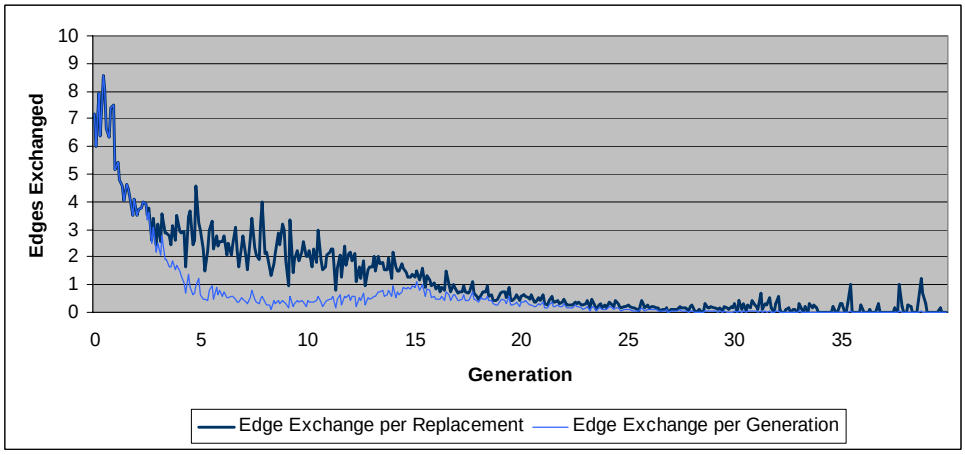


Figure E.26 Edges exchanged by REM for S1

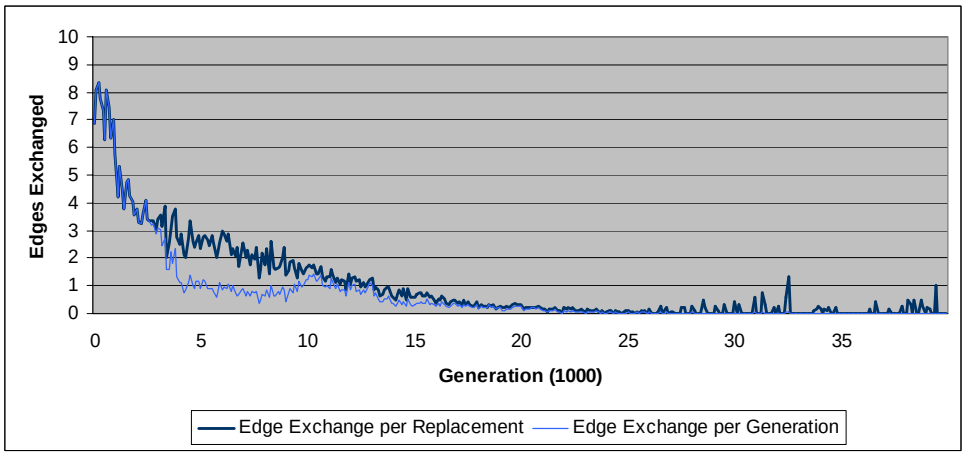


Figure E.27 Edges exchanged by REM for S2

E.5. Optimal Edge Discovery and Preservation

A powerful genetic algorithm must have the ability to discover optimal edges and keep them in the population. Figures E. 28 – 41 are prepared to measure the ability to discover new edges and to keep them within the population.

The edge difference among individuals is plotted with respect to the number of generations. The edge difference is calculated using the most common 1002 edges in the population. The percent of edges that are not in the popular list on each individual are used as the difference measure. The fact that most popular 1002 edges will give the common tour in a fully converged population is the main rationale behind this difference measure. The individual with minimum percent of edges different from the most popular edges, the average percent of different edges, and individual with the maximum percentage of edges different from the most popular edges are plotted in Figures E.28, E.32, E.34, E.36 and E.38. The percentage of edges that are present in the optimal solution but are not included in each individual is plot in Figures E.31, E.33, E.35, E.37, and E.39. The individual that contains minimum and maximum number of optimal edges are plotted with the average number of optimal edges missed in these figures. The inflection points correspond to point where the most popular edges change. For instance in the NNX without mutation with S1 (Figure E.28), the slope of the average difference changes between 3000^{th} and 4000^{th} generations. The edge difference decreases suddenly parallel to the edges borrowed from the complete graph (Figure E.14); the inflection point at 4000 is possibly due to a change in the most popular edges. The plots of the most popular 1002 edges at $30{,}000^{th}$ and $4{,}000^{th}$ generations can be seen in Figures E.29 and E.30.

In the edge difference percentages, there are 3 different points in which the most popular edges change, the effect of these changes can be the possible reason of the small fluctuations in the average number of different edges per individual. Moreover, the edge difference decreases fast for 3,000 generations, and then the difference rate decreases steadily until 10,000. The individual with maximum difference suddenly decreases at 13,000, and then a 1% increase is observed. The result of this sudden decrease is the change in the most popular edges.

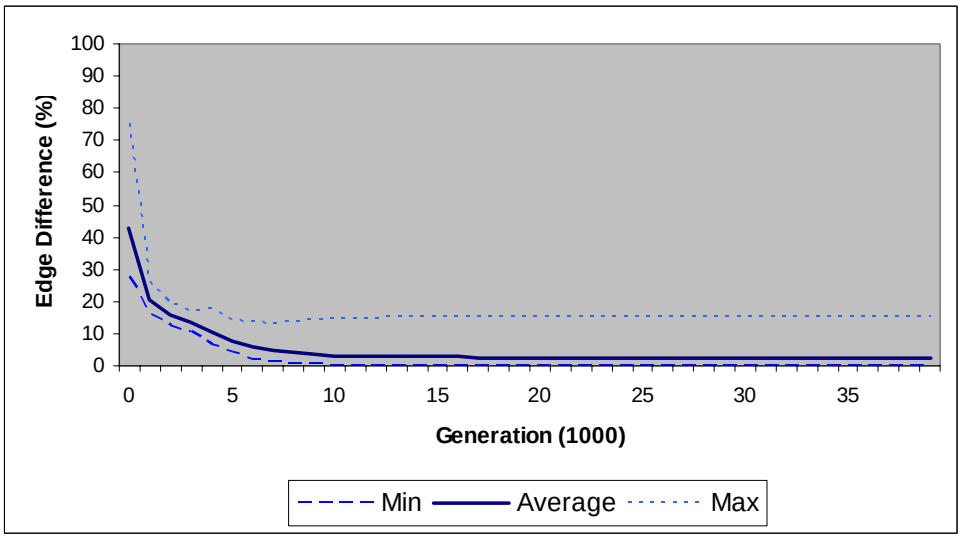


Figure E.28 Edge difference among individuals with pure NNX for S1

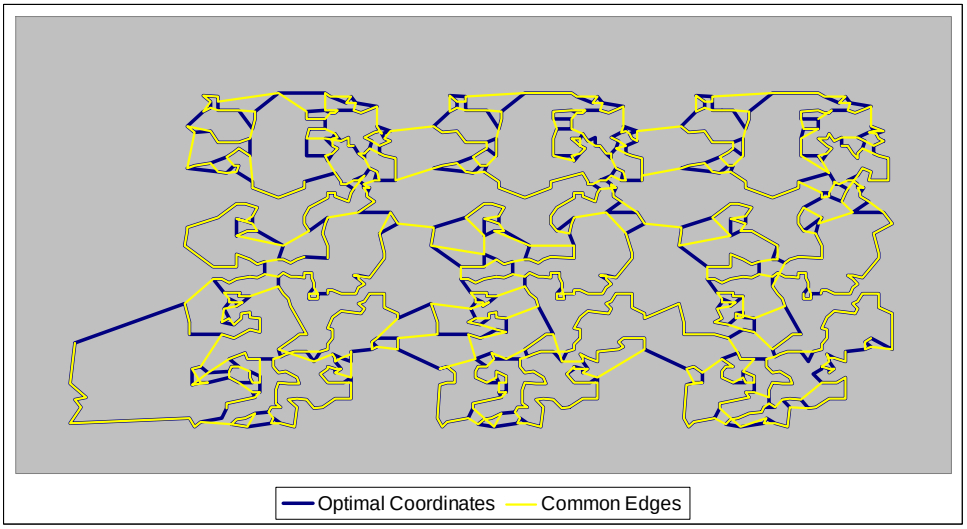


Figure E.29 Most popular edges and optimal edges for pr1002 at 3,000th generation

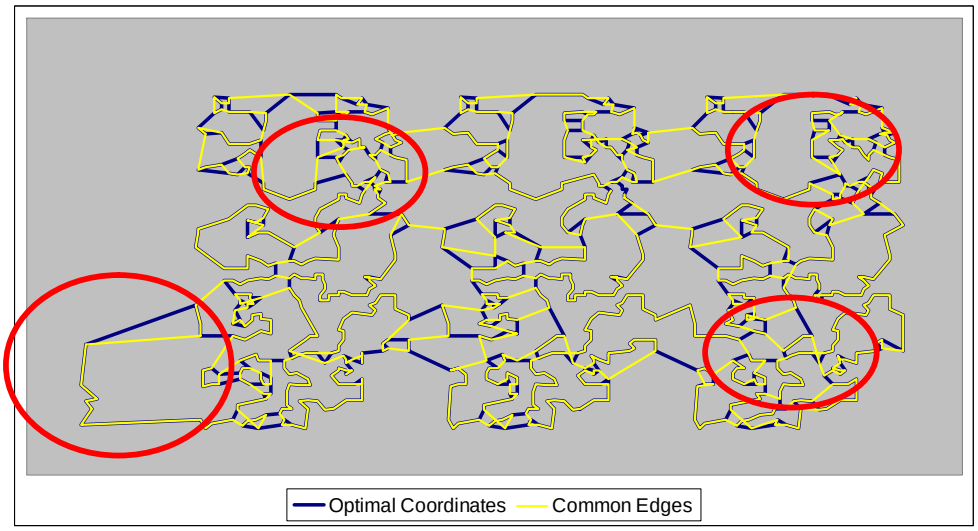


Figure E.30 Most popular edges and optimal edges for pr1002 at 4,000th generation

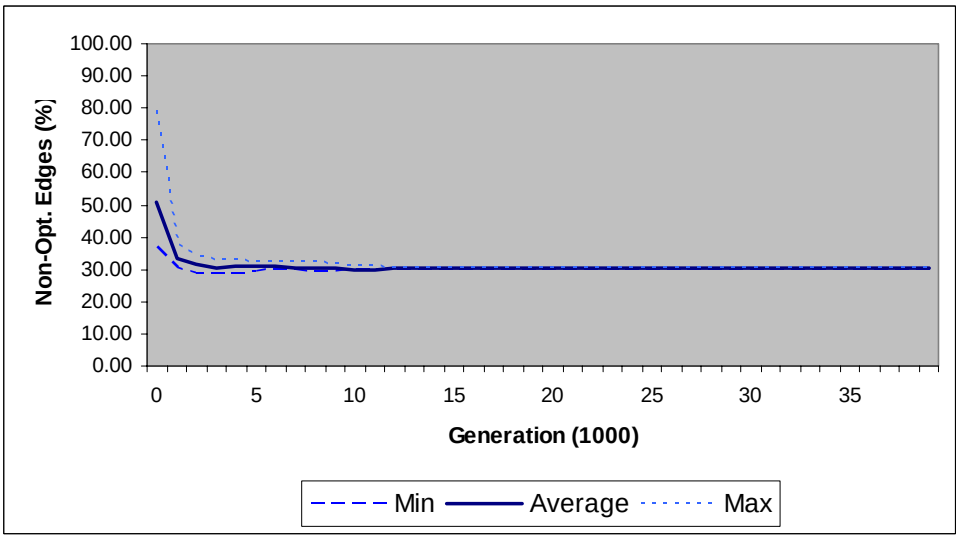


Figure E.31 Percent of optimal edges not covered by individuals with pure NNX for S1

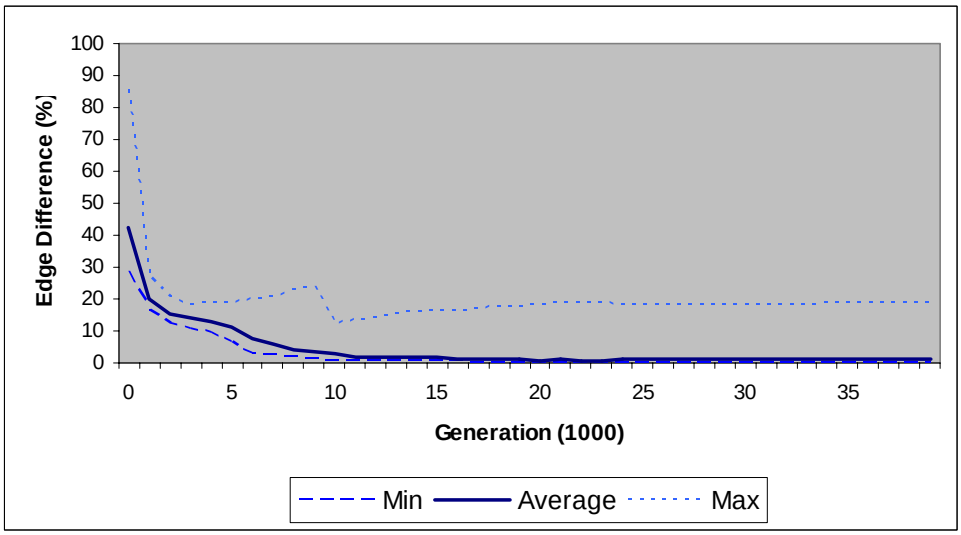


Figure E.32 Edge difference among individuals with pure NNX for S1

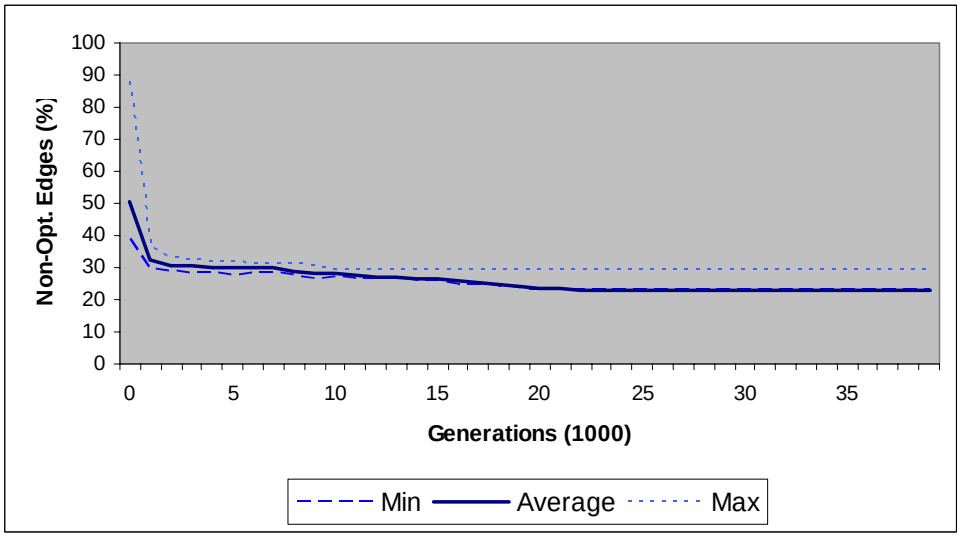


Figure E.33 Percent of optimal edges not covered by individuals with pure NNX for S2

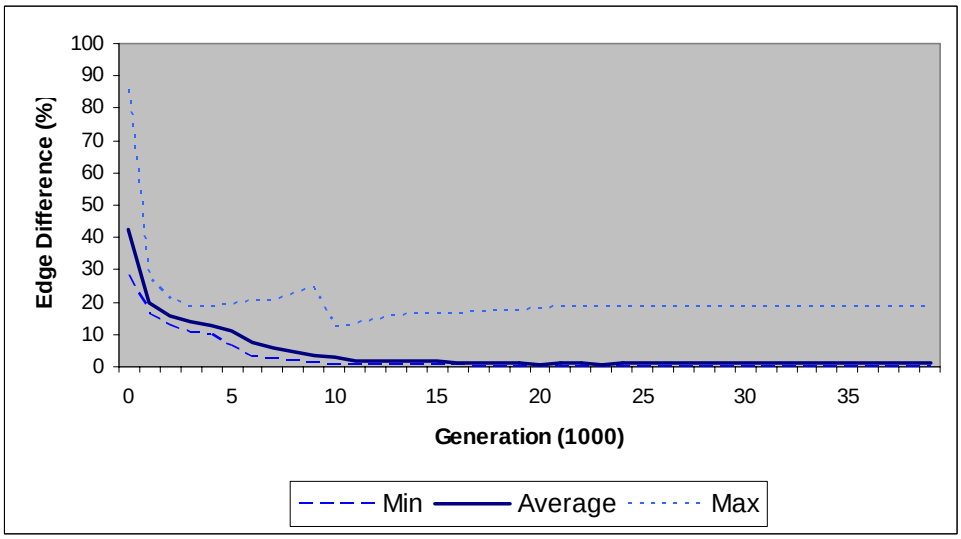


Figure E.34 Edge difference among individuals using LEM and CIM for S1

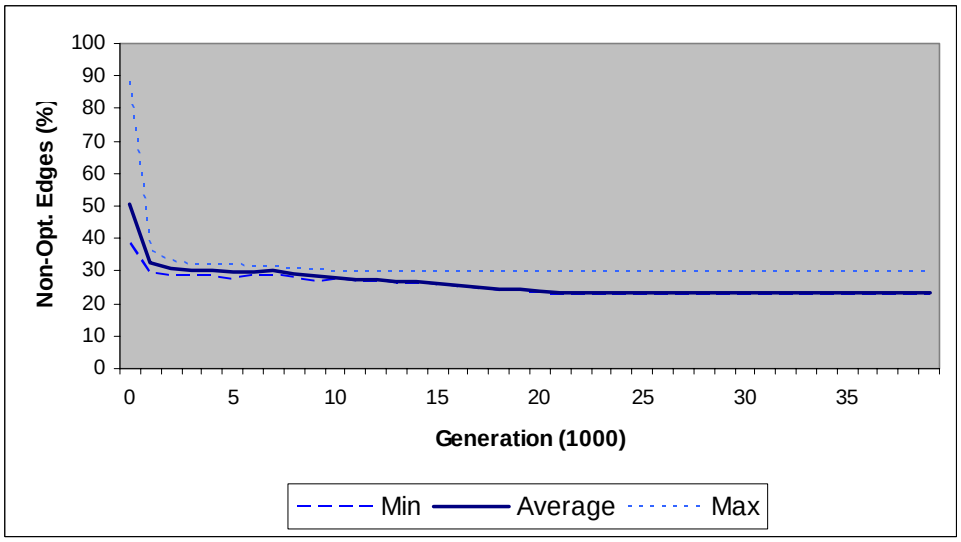


Figure E.35 Percent of optimal edges not covered by individuals with using LEM and CIM for S1

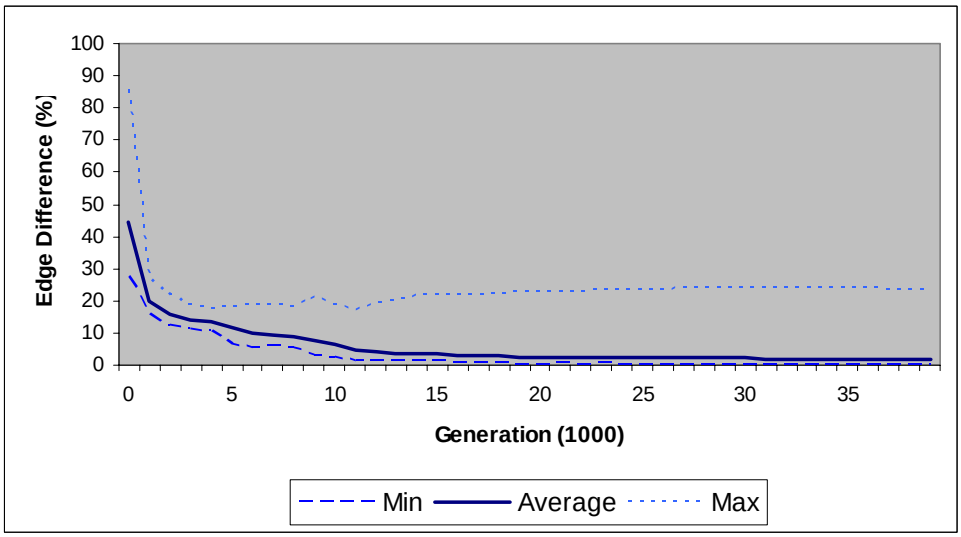


Figure E.36 Edge difference among individuals using LEM and CIM for S2

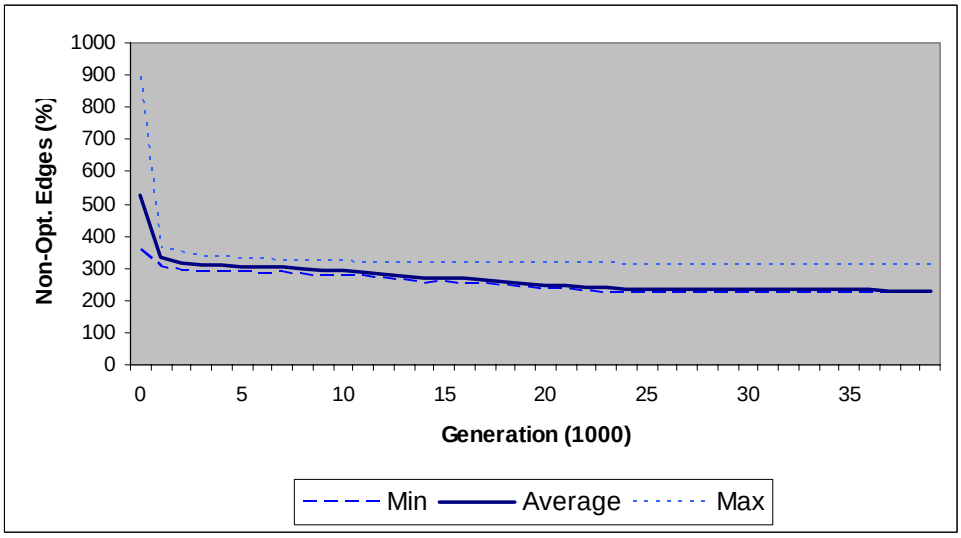


Figure E.37 Percent of optimal edges not covered by individuals with using LEM and CIM for S2

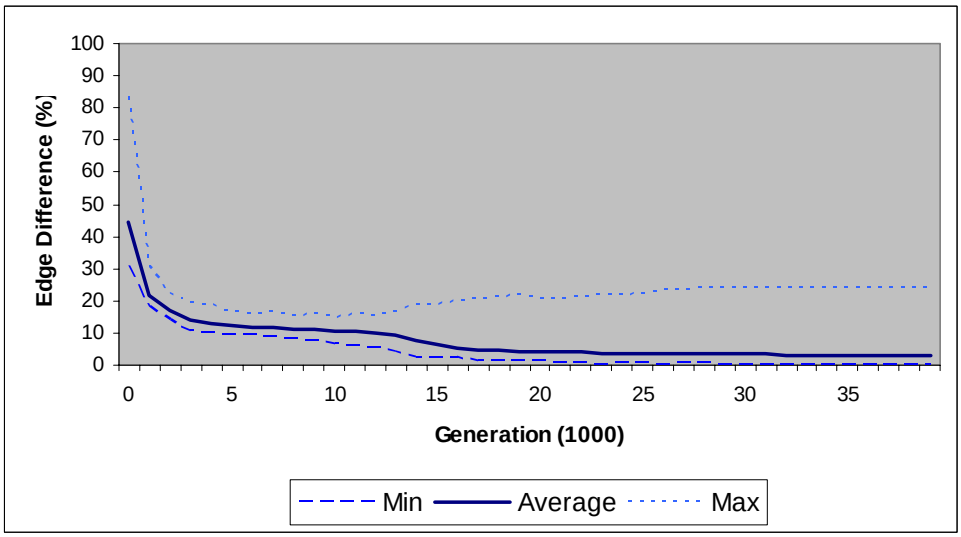


Figure E.38 Edge difference among individuals using REM and CIM for S1

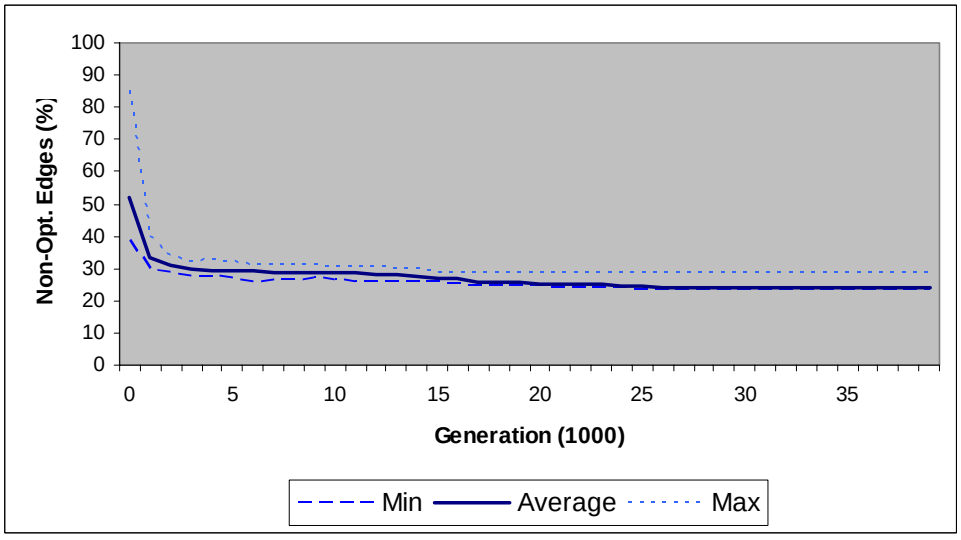


Figure E.39 Percent of optimal edges not covered by individuals with using REM and CIM for S1

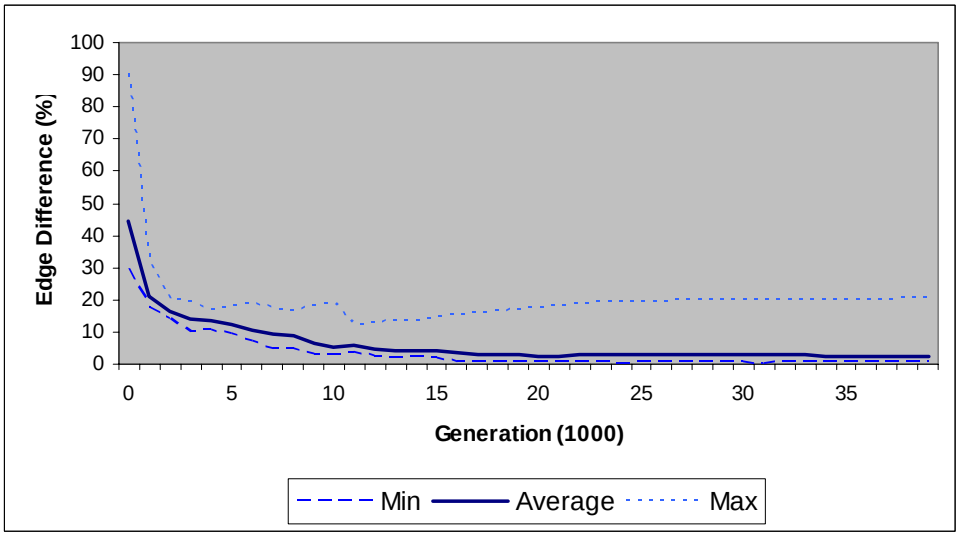


Figure E.40 Edge difference among individuals using REM and CIM for S2

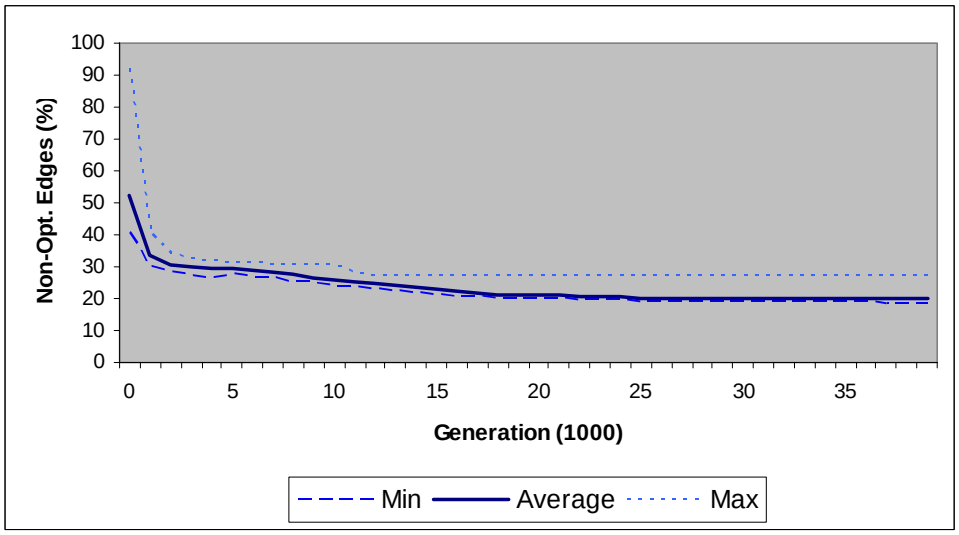


Figure E.41 Percent of optimal edges not covered by individuals with using REM and CIM for S2

F .RESULTS OF PAIRED T-TESTS FOR TSPB

Results for All Problems

Paired T-Test and CI: GENI versus Best of our GA

```
Paired T for GENI - Best

                 N      Mean     StDev   SE Mean
GENI            25      2035       792       158
Best            25      1986       779       156
Difference      25     48,55     18,20      3,64

95% CI for mean difference: (41,04; 56,06)
T-Test of mean difference = 0 (vs not = 0): T-Value = 13,34  P-Value = 0,000
```

Paired T-Test and CI: GENIUS versus Best of our GA

```
Paired T for GENIUS - Best

                 N      Mean     StDev   SE Mean
GENIUS          25      1998       779       156
Best            25      1986       779       156
Difference      25     11,70     11,66      2,33

95% CI for mean difference: (6,89; 16,51)
T-Test of mean difference = 0 (vs not = 0): T-Value = 5,02  P-Value = 0,000
```

Paired T-Test and CI: GEN- VNS versus Best of our GA

```
Paired T for GEN- VNS - Best

                 N      Mean     StDev   SE Mean
GEN- VNS        25      1990       776       155
Best            25      1986       779       156
Difference      25      3,82     11,89      2,38

95% CI for mean difference: (-1,09; 8,72)
T-Test of mean difference = 0 (vs not = 0): T-Value = 1,60  P-Value = 0,122
```

Paired T-Test and CI: SOFM versus Best of our GA

```
Paired T for SOFM - Best

                 N      Mean     StDev   SE Mean
SOFM            25      2017       780       156
Best            25      1986       779       156
Difference      25     31,29     16,58      3,32

95% CI for mean difference: (24,45; 38,14)
T-Test of mean difference = 0 (vs not = 0): T-Value = 9,43  P-Value = 0,000
```

Paired T-Test and CI: SOFM* versus Best of our GA

```
Paired T for SOFM* - Best

                 N      Mean     StDev   SE Mean
SOFM*           25      1996       778       156
Best            25      1986       779       156
Difference      25      9,78     11,59      2,32

95% CI for mean difference: (5,00; 14,56)
T-Test of mean difference = 0 (vs not = 0): T-Value = 4,22  P-Value = 0,000
```

Paired T-Test and CI: GENI versus Avg of our GA

```
Paired T for GENI - Avg

                 N      Mean     StDev   SE Mean
GENI            25      2035       792       158
Avg             25      1989       781       156
Difference      25     45,76     16,47      3,29

95% CI for mean difference: (38,96; 52,55)
T-Test of mean difference = 0 (vs not = 0): T-Value = 13,89  P-Value = 0,000
```

Paired T-Test and CI: GENIUS versus Avg of our GA

```
Paired T for GENIUS - Avg

                 N      Mean     StDev   SE Mean
GENIUS          25      1998       779       156
Avg             25      1989       781       156
Difference      25      8,90     11,75      2,35

95% CI for mean difference: (4,05; 13,75)
T-Test of mean difference = 0 (vs not = 0): T-Value = 3,79  P-Value = 0,001
```

Paired T-Test and CI: GEN- VNS versus Avg of our GA

```
Paired T for GEN- VNS - Avg

                 N      Mean     StDev   SE Mean
GEN- VNS        25      1990       776       155
Avg             25      1989       781       156
Difference      25      1,02     12,68      2,54

95% CI for mean difference: (-4,22; 6,25)
T-Test of mean difference = 0 (vs not = 0): T-Value = 0,40  P-Value = 0,691
```

Paired T-Test and CI: SOFM versus Avg of our GA

```
Paired T for SOFM - Avg

                 N      Mean     StDev   SE Mean
SOFM            25      2017       780       156
Avg             25      1989       781       156
Difference      25     28,50     16,33      3,27

95% CI for mean difference: (21,76; 35,24)
T-Test of mean difference = 0 (vs not = 0): T-Value = 8,73  P-Value = 0,000
```

Paired T-Test and CI: SOFM* versus Avg of our GA

```
Paired T for SOFM* - Avg

                 N      Mean     StDev   SE Mean
SOFM*           25      1996       778       156
Avg             25      1989       781       156
Difference      25      6,98     11,79      2,36

95% CI for mean difference: (2,12; 11,85)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2,96  P-Value = 0,007
```

Results for problems of size 500 and less

Paired T-Test and CI: GENI versus AVG of our GA

```
Paired T for GENI - AVG

                 N      Mean     StDev   SE Mean
GENI            20      1711       486       109
AVG             20      1668       474       106
Difference      20     42,54     15,01      3,36

95% CI for mean difference: (35,51; 49,56)
T-Test of mean difference = 0 (vs not = 0): T-Value = 12,67  P-Value = 0,000
```

Paired T-Test and CI: GENIUS versus AVG of our GA

```
Paired T for GENIUS - AVG

                 N      Mean     StDev   SE Mean
GENIUS          20      1679       477       107
AVG             20      1668       474       106
Difference      20     10,85      9,81      2,19

95% CI for mean difference: (6,26; 15,44)
T-Test of mean difference = 0 (vs not = 0): T-Value = 4,94  P-Value = 0,000
```

Paired T-Test and CI: GENIUS- VNS versus AVG of our GA

```
Paired T for GEN- VNS - AVG

                 N      Mean     StDev   SE Mean
GEN- VNS        20      1673       476       106
AVG             20      1668       474       106
Difference      20      4,67     10,26      2,29

95% CI for mean difference: (-0,13; 9,47)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2,04  P-Value = 0,056
```

Paired T-Test and CI: SOFM versus AVG of our GA

```
Paired T for SOFM - AVG

                 N      Mean     StDev   SE Mean
SOFM            20      1697       475       106
AVG             20      1668       474       106
Difference      20     29,12     13,91      3,11

95% CI for mean difference: (22,60; 35,63)
T-Test of mean difference = 0 (vs not = 0): T-Value = 9,36  P-Value = 0,000
```

Paired T-Test and CI: SOFM* versus AVG of our GA

```
Paired T for SOFM* - AVG

                 N      Mean     StDev   SE Mean
SOFM*           20      1677       477       107
AVG             20      1668       474       106
Difference      20      9,37     10,45      2,34

95% CI for mean difference: (4,48; 14,26)
T-Test of mean difference = 0 (vs not = 0): T-Value = 4,01  P-Value = 0,001
```

Paired T-Test and CI: GENI versus BEST of our GA

```
Paired T for GENI - BEST

                 N      Mean     StDev   SE Mean
GENI            20      1711       486       109
BEST            20      1666       472       106
Difference      20     44,44     16,13      3,61

95% CI for mean difference: (36,90; 51,99)
T-Test of mean difference = 0 (vs not = 0): T-Value = 12,33  P-Value = 0,000
```

Paired T-Test and CI: GENIUS versus BEST of our GA

```
Paired T for GENIUS - BEST

                 N      Mean     StDev   SE Mean
GENIUS          20      1679       477       107
BEST            20      1666       472       106
Difference      20     12,76     10,19      2,28

95% CI for mean difference: (7,99; 17,53)
T-Test of mean difference = 0 (vs not = 0): T-Value = 5,60  P-Value = 0,000
```

Paired T-Test and CI: GENIUS- VNS versus BEST of our GA

```
Paired T for GEN- VNS - BEST

                 N      Mean     StDev   SE Mean
GEN- VNS        20      1673       476       106
BEST            20      1666       472       106
Difference      20      6,58     10,37      2,32

95% CI for mean difference: (1,73; 11,43)
T-Test of mean difference = 0 (vs not = 0): T-Value = 2,84  P-Value = 0,011
```

Paired T-Test and CI: SOFM versus BEST of our GA

```
Paired T for SOFM - BEST

                 N      Mean     StDev   SE Mean
SOFM            20      1697       475       106
BEST            20      1666       472       106
Difference      20     31,02     14,35      3,21

95% CI for mean difference: (24,31; 37,74)
T-Test of mean difference = 0 (vs not = 0): T-Value = 9,67  P-Value = 0,000
```

Paired T-Test and CI: SOFM* versus BEST of our GA

```
Paired T for SOFM* - BEST

                 N      Mean     StDev   SE Mean
SOFM*           20      1677       477       107
BEST            20      1666       472       106
Difference      20     11,28     10,86      2,43

95% CI for mean difference: (6,19; 16,36)
T-Test of mean difference = 0 (vs not = 0): T-Value = 4,64  P-Value = 0,000
```