A GENETIC ALGORITHM FOR THE BIOBJECTIVE TRAVELING SALESMAN PROBLEM WITH PROFITS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNCAL UNIVERSITY

BY

SERDAR KARADEMİR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
INDUSTRIAL ENGINEERING

JULY 2008

Approval of thesis:

A GENETIC ALGORITHM FOR THE BIOBJECTIVE TRAVELING SALESMAN PROBLEM WITH PROFITS

submitted by **SERDAR KARADEMİR** in partial fulfillment of the requirements for the degree of **Master of Science in Industrial Engineering Department**, **Middle East Technical University** by,

Prof. Dr. Canan Özgen _______________
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Nur Evin Özdemirel _______________
Head of Department, **Industrial Engineering**

Assoc. Prof. Dr. Haldun Süral _______________
Supervisor, **Industrial Engineering Dept., METU**

Assoc. Prof. Dr. Esra Karasakal _______________
Co-Supervisor, **Industrial Engineering Dept., METU**

Examining Committee Members:

Prof. Dr. Nur Evin Özdemirel _______________
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Haldun Süral _______________
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Esra Karasakal _______________
Industrial Engineering Dept., METU

Assoc. Prof. Dr. Yasemin Serin _______________
Industrial Engineering Dept., METU

Bora Kat (M.Sc.) _______________
TUBITAK

Date: **14.07.2008**

ABSTRACT

A GENETIC ALGORITHM FOR THE BIOBJECTIVE TRAVELING SALESMAN PROBLEM WITH PROFITS

Karademir, Serdar
M.S., Department of Industrial Engineering
Supervisor : Assoc. Prof. Dr. Haldun Süral
Co-Supervisor : Assoc. Prof. Dr. Esra Karasakal

July 2008, 80 pages

In Traveling Salesman Problem (TSP) with profits, a profit is associated with each city and the requirement to visit all cities is removed. The purpose is to simultaneously minimize cost (excluding as many cities as possible) and maximize profit (including as many cities as possible). Although the reduced single-objective case of the problem has been well-studied, the true biobjective problem has been studied only by a few researchers. In this paper we study the true biobjective problem using the Multiobjective Genetic Algorithm NSGA II and the Lin-Kernighan Heuristic. We propose several improvements for NSGA II in solving the problem. Based on these improvements, we provide computational results of the approximated Pareto-optimal front for a set of practically large size TSP instances. Finally, we provide a framework and its computational results for a post-optimality analysis to guide the decision maker, using the data mining software Clementine.

Key Words: TSP with Profits, Evolutionary Multiobjective Combinatorial Optimization, Data Mining.

ÖZ

ÇOK AMAÇLI KAR GETİREN GEZGİN SATICI PROBLEMİ İÇİN GENETİK BİR ALGORİTMA

Karademir, Serdar
Yüksek Lisans, Endüstri Mühendisliği Bölümü
Tez Yöneticisi : Doç. Dr. Haldun Süral
Ortak Tez Yöneticisi : Doç. Dr. Esra Karasakal

Temmuz 2008, 80 sayfa

Kar getiren Gezgin Satıcı Problemi'nde (KGSP) her şehire bir kar atanır ve tüm şehirleri gezme zorunluluğu ortadan kalkar. KGSP'de amaç aynı zamanda hem maliyeti en aza indirmek (en az sayıda şehir gezerek) hem de kazancı en yüksek seviyede tutmaktır (en fazla sayıda şehir gezerek). Tek amaçlı KGSP türleri literatürde fazlaca yer edinmiş olsa da, gerçek anlamda iki amaçlı KGSP çok az araştırmacı tarafından çalışılmıştır. Bu çalışmada iki amaçlı KGSP için Çok Amaçlı Genetik Algoritma NSGA-II ve Lin-Kernighan sezgisel yöntemlerine dayalı bir çözüm sunmaktayız. Bu çalışmada NSGA-II için bir kaç iyileştirme önerilmektedir. Bu iyileştirmelerin sonucunda literatürdeki orta ve büyük boyutlarda pek çok problem için tahmini etkin sınırlar (approximate efficient frontier) verilmektedir. Son olarak, veri madenciliği yazılımı Clementine kullanarak, yaratılan etkin sınırlarda karar vericiyi yönlendirebilecek bir eniyileme-sonrası analiz taslağı sunmaktayız.

Anahtar Kelimeler: Kar Getiren GSP, Evrimsel Çok Amaçlı Kombinasyonal Eniyileme, Veri Madenciliği.

To My Lovely Family

ACKNOWLEDGEMENTS

I would like to express my gratitude to my Supervisor Assoc. Prof. Dr. Haldun SÜRAL and Co-Supervisor Assoc. Prof. Dr. Esra KARASAKAL for their guidance, advice, criticism, encouragements and insight throughout this study.

Most importantly, I would like to express my deepest thanks to my parents, Ayten KARADEMİR and Ziya KARADEMİR. I am forever indebted to them for their understanding, endless patience, love and encouragement.

I am also grateful to my brother Serhat and my sister Naciye who listened to my complaints and motivated me during this study. I want also express my gratitude for my friends Nermin, Ömer, Engin, and Cemal.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In this chapter we introduce preliminary concepts regarding Traveling Salesman Problem with Profits and present an overview of the work conducted by authors of this paper.

Problem Definition:

Given a graph ***G(****E*,*V**)*** with edge set *E*, vertex set *V*, and costs on edge set, Travelling Salesman Problem (TSP) can be defined as finding the shortest Hamiltonian tour on ***G***. Assuming that there is also a prize associated with each vertex in ***V***, in TSP with Profits (TSPP) the aim is to simultaneously minimize the route cost (excluding as many vertices as possible) and maximize the prize (including as many vertices as possible). When compared to TSP, in TSPP, there is a prize (i.e., profit) associated with each vertex and the constraint for including all vertices into the tour is removed. Moreover, since it is not required to visit all vertices, a depot vertex is generally defined for TSPP and inclusion of this vertex to all considered tours is enforced.

Since TSPP can be reduced to TSP, it is also NP-complete. Considering an instance for TSPP where all vertices are included in the tour (i.e., the instance with maximum prize), finding the shortest Hamiltonian tour corresponds to TSP. Another difficulty introduced with TSPP is the combinatorial number of subsets of vertices that are to be considered. An enumeration algorithm has to consider 2^n subsets of vertices for a graph with *n* vertices. This also means that the enumeration algorithm has to solve 2^n TSPs. Hence, as number of vertices considered increases linearly, the number of TSPs to be solved and the difficulty of solving these TSP instances (i.e., time required to consider all permutations of vertices for a given subset of vertices) increase exponentially.

After providing the formal definition of TSPP, we want to state that in the rest of this paper we will refer “vertices” as “cities” and “prizes” as “profits”. This notation is a practical and commonly used one in literature.

Since there are two conflicting objectives in TSPP, the problem is a multiobjective optimization problem. There is not generally a single optimal solution for TSPP. Solution of a TSPP is a set of nondominated solutions. However, since the problem is a very difficult problem, it is generally studied as a single objective problem by some sort of transformation. In Profitable Tour Problem (PTP), objectives are converted in commensurable units and the objective function for problem is defined as minimization of cost minus profit. In Orienteering Problem (OP), objective for cost is included in constraint set as a knapsack constraint and profit is maximized. In Prize Collecting TSP (PCTSP), profit objective is included into constraint set as a general covering constraint and cost is minimized.

Although PTP, OP, and PCTSP are all well studied in literature, the true biobjective problem is considered only by Keller et al. (1988), Şimşek (2007), and Berube et al. (2008). These authors use exact and heuristic methods on OP and PCTSP settings and generate a set of nondominated solutions for the problem. These authors could not solve larger problem instances, limiting themselves with instance sizes up to 150 cities. In this study we examine a Multiobjective Genetic Algorithm (MOGA) to apply to TSPP, propose several improvements for this genetic algorithm, and demonstrate computational results for a set of problems in literature the largest of which being a problem with 400 cities. We present two metrics to evaluate results of our proposed algorithm; Hyper Volume and Generational Distance. Both metrics are able to evaluate performance of a MOGA in terms of both convergence and dispersion properties.

A Post-optimality analysis framework is also provided in this paper. We use datamining to enhance decision making process after optimization. Rules and indications derived from generated nondominated solution sets are provided to enable a DM to make more conscious decisions.

Real Life Applications of TSPP:

One of the applications of TSPP is where it is not possible to visit all customers due to some constraints enforced. Then we have to make the best selection among customers subject to enforced constraints. For instance, when a manufacturer can not visit all of his suppliers due to some constraints, he has to choose a set of suppliers to visit first and then make the routing decision. In another case, a salesman may want to sell a given quota of a product as soon as possible, hence, make a decision on which locations and in what order to visit. Another application area of TSPP is daily scheduling of steel-rolling mills. In this context a producer should select a set of steel slabs and order them for hot or cold rolling to produce steel sheets. Orienteering competitions are also one of the application areas of TSPPs. In orienteering games a player should collect maximum amount of hidden prize in a preset time limit. TSPPs are also encountered as subproblems in solution procedures of other problems. Reader is referred to study of Feillet et al. (2005) for a review on TSPP.

Motivation:

In this study we focus on the true biobjective TSPP. The aim of this study is to generate the whole Pareto front, up to a certain extent, for TSPP. We argue that exact algorithms practically are not feasible due to harmful complexity of the problem and propose a Metaheuristic that can approximate the Pareto-optimal front with a reasonable error. Such a Metaheuristic could also be used to demonstrate the structure of Pareto front for very large problem instances (e.g., problems including 5000 cities) where deviation from Pareto-optimal front may be relatively considerable. With the hope to present all available actions to Decision Maker (DM), we approximate complete Pareto front. However, we also provide a Post-optimality analysis framework that could guide the DM to preferred solutions on generated Pareto front.

Organization of the thesis is as follows: In Chapter 2 we present a mathematical model for TSPP and summarize literature review on TSPP, MOGA, and preference incorporation issues for MOGAs. In Chapter 3, a review of one of the state-of-the-art MOGAs is presented and several improvements for the algorithm are proposed. Chapter 4 explains the selection of test problems and gives computational results for selected problems. In Chapter 5 we provide a framework for post-optimality analysis of a given pareto front and conclude our study in Chapter 6.

CHAPTER 2

LITERATURE REVIEW

In this chapter we provide a mathematical model for TSPP, review studies on TSPP, summarize literature on Multiobjective Genetic Algorithms (MOGAs), and give preference incorporation strategies proposed for MOGAs in literature. We also provide several definitions regarding multiobjective optimization.

2.1 Mathematical Model For TSPP

The mathematical model for TSPP is a linear integer program. It includes assignment constraints and subtour elimination constraints for TSP. Different from TSP, in the formulation for TSPP, the right hand side for assignment constraints is also a decision variable. Moreover, there are two conflicting objective functions to be optimized.

The model is as follows:

Decision Variables:

x_{ij} = 1 if city *j* is visited after city *i*, 0 otherwise.
y_i = 1 if city *i* is visited, 0 otherwise.

Parameters:

c_{ij}: cost of visiting city j right after city *i*.
p_i: profit associated with city *i*.

The model:

$$\text{MIN} \sum_{i,j} c_{ij} * x_{ij} \tag{1}$$

$$\text{MAX} \sum_{i} p_i * y_i \tag{2}$$

Subject to

$$\sum_{j \neq i} x_{ij} = y_i \qquad \forall\, i \tag{3}$$

$$\sum_{i \neq j} x_{ij} = y_j \qquad \forall\, j \tag{4}$$

$$\mathit{Subtour\ Elimination\ Constraints} \tag{5}$$

$$x_{ij} \ \text{BIN} \tag{6}$$

$$y_i \ \text{BIN} \tag{7}$$

(1) and (2) represent the cost and profit objectives respectively for the tour to be constructed. (3) ensures that if a city is included in the tour, then, there is an arc leaving this city. (4) similarly ensures that if a city is included in the tour, then, there is an arc entering this city. (5) represents a set of constraints that ensures construction of subtours are not allowed. (6) and (7) are 0-1 constraints that enforces related variables to take only 0 or 1 value.

We should also note that a depot constraint that ensures depot city is always included in the tour could be added to this formulation. We would simply force the binary variable y_{depot} to take a value of 1. Inserting such a constraint ensures that depot city is included in all tours generated. This extra constraint does not make the problem harder.

2.2 TSPP Literature

TSPP is mainly studied as a single objective problem in the literature under different settings. Feillet et al. (2005) report the following three problem settings.

Profitable tour problem (PTP): Objectives (1) and (2) are combined as minimization of cost minus prize. Constraint set remains unchanged. However, cost and prize must be of the same type (i.e., Dollars) in order to be able to use this setting. Dell'Amico et al. (1995) propose lower bounds for problem and show that large problem instances for PTP could be solved efficiently.

Orienteering problem (OP): Objective (1) is formulated as a constraint with a right hand side value corresponding to an upper bound on tour cost in terms of time. This is generally the case in orienteering problems where players try to collect maximum amount of points in a limited time. This problem is also known as Selective TSP.

Tsiligirides (1984) proposes several algorithms for OP which he calls SOE (Score Orienteering Event). He proposes S-Algorithm (Stochastic Algorithm), D-Algorithm (Deterministic Algorithm), and a RI-Algorithm (Route Improvement Algorithm). S-Algorithm constructs a tour based on stochastic rules, D-Algorithm constructs a tour based on deterministic algorithm, and RI-Algorithm improves a given route using "savings" principle. Then Tsiligirides combines S, D, and RI algorithms where tours constructed by S and D algorithms are improved with RI-Algorithm. He also conducts a study on effect of profit structure on problem, however, he concludes that no satisfactory results are obtained.

Laporte and Martello (1990) propose a Branch and Bound (B&B) algorithm for the problem. They first solve the LP-relaxation of the problem and then use B&B to solve for violated constraints.

Ramesh et al. (1992) propose a Branch-and-Bound (BB) algorithm for OP. First they transform the original problem formulation by moving prizes from nodes to edges. The reasoning is that, the profit on two nodes can be carried on the edge between these two nodes by taking average of profits on the nodes. If a node is visited, then there is exactly two arcs connecting this node to other nodes in the tour and hence profit of that node will be exactly collected. Using this approach authors drop the binary variables showing whether a city is visited and update the cost matrix using profits. They use Lagrangian relaxation at root node which they call Phase 1 of their algorithm. Then they use subgradient optimization. If optimal solution is not found, then they start the BB approach by branching, which they call Phase 2 of their algorithm. They provide five nondominated solutions for problems sizes up to 80 cities and three nondominated solutions for problem sizes larger than 80 cities. They generate problems randomly and the largest problem they solve includes 150 cities.

Chao, et al. (1996) propose a two-step heuristic for selective TSP. The first step in their heuristic is to generate several routes and select the best route as the initial solution. Then they use several greedy heuristics to improve this initial tour in the second phase.

Millar and Kiragu (1997) propose a time-based formulation and an upper bounding scheme for OP. In their model they define flow variables to reduce the number of constraints in original OP formulation. They use CPLEX to solve their model and use problems including at most 10 nodes. They also apply their methodology to a 15-zone fisheries surveillance problem.

Awerbuch et al. (1998) propose an algorithm based on an approximation schema for *k-MST* that has a poly-logarithmic performance. Their algorithm is both applicable to OP and PCTSP which is described next.

Gendreau et al. (1998a) propose a branch-and-cut algorithm for OP. They provide two heuristics that they use to update their bounds and quickly fathom nodes in branch-and-cut tree. They provide results for problem sizes up to 300 cities. However, again only a few nondominated solutions for each problem type are found by their algorithm.

Gendreau et al. (1998b) propose Tabu Search (TS) algorithm for OP. First they propose a heuristic which they call Insert and Shake to obtain an initial solution for their TS algorithm. In their TS algorithm the neighborhood structure is defined as solutions obtained by removing or inserting clusters of cities. Hence, the algorithm differs from other algorithms performing insertion of a city into the tour or deletion of a city from the tour. As in Gendreau (1998a), they provide a maximum of 5 solutions for problem sizes less than 100 cities and number of solutions they provide for a maximum problem size of 300 cities decreases to one.

Fischetti et al. (1998) propose a Branch-and-Cut algorithm that solves OP optimally. They calculate three nondominated solutions for Vehicle Routing Problems and one nondominated solution for TSPs.

Tasgetiren and Smith (2000) propose a genetic algorithm (GA) to solve the orienteering problem. They propose several crossover and mutation operators that are designed for TSP. Four test sets are used. Three of the sets are originally from Tsiligirides (1984) and one is corrected by Chao, et al. (1996). Their algorithm is able to solve problems with at most 33 cities. Since their genetic algorithm also tries to find the best TSP tour for a given set of cities besides choosing this set of cities, they are unable to solve large problem instances.

Liang et al. (2002) present two metaheuristics for OP; an ant colony optimization algorithm and a tabu search algorithm. Heuristics described are used to solve OP, the single objective TSPP. Hence, Liang et al. (2002) do not introduce a new algorithm but apply existing single objective metaheuristics using a different local search schema. The largest problem instance their algorithm is able to solve includes 33 cities.

Prize collecting TSP (PCTSP): Objective (2) is formulated as a constraint with a right hand side value corresponding to a lower bound on prize to be collected. This problem is encountered when a salesman has to sell a given quota of a product, hence also known as Quota TSP.

Dell'Amico et al. (1998) report that PCTSP was introduced by Balas and Martin (1985 and 1991) for scheduling of the daily operations of a steel rolling mill. Balas (1989 and 1995) also presents structural properties of the PCTSP related to the TSP and Knapsack polytopes.

Dell'Amico et al. (1998) propose a lagrangian heuristic for PCTSP where they start from a feasible solution and use a method they call "Extension and Collapse" to improve this initial feasible solution.

Balas (1999) introduces a special case of PCTSP where he provides an algorithm of polynomial time. Besides, he also notes that this algorithm could be used as a heuristic when solving general PCTSP.

To our knowledge, the true biobjective problem is studied only by Keller et al. (1988), Şimşek (2007) and Berube et al. (2008). Keller et al. (1988) generate nondominated solutions for TSPP which they call Multiobjective Vending Problem. They propose seven routines to construct an initial tour and improve that tour with under a knapsack constraint. They give results for a 25-city problem. Şimşek (2007) uses ε-constraint method to generate the Pareto front. He uses OP setting and solves ε-constraint problems through CGW heuristic developed by Chao et al. (1996). Results very close to Pareto-optimal front are reported for problem sizes up to 121 vertices. Berube et. al. (2008), on the other hand, uses ε-constraint method to solve

PCTSP. They use a Branch-and-Cut algorithm and report Pareto-optimal fronts for problem sizes up to 150 vertices where integer cost and prize values are used.

2.3 Definitions For Multiobjective Optimization

The general MO problem can be represented as follows;

Min $f_1(x)$
Min $f_2(x)$
…………
Min $f_n(x)$

Subject To
$x \in X$

f_i are objective functions, x is a decision space vector, X is called the "decision space", and objective function values are represented in "objective space", Z, with objective space vector z, which is the corresponding objective space vector for x. Figure 1 demonstrates the decision space and objective space in a two dimensional case.

The following definitions are also needed before we continue our discussion (Deb, 2001);

Definition 1. A solution s_1 is said to dominate s_2 if and only if it is no worse than s_2 in all objectives and strictly better than s_2 in at least one objective.

Definition 2. A decision space vector x is said to be efficient if its corresponding objective space vector z is nondominated.

Definition 3. The set of all nondominated solutions of entire feasible solution space is called the Pareto-optimal set.

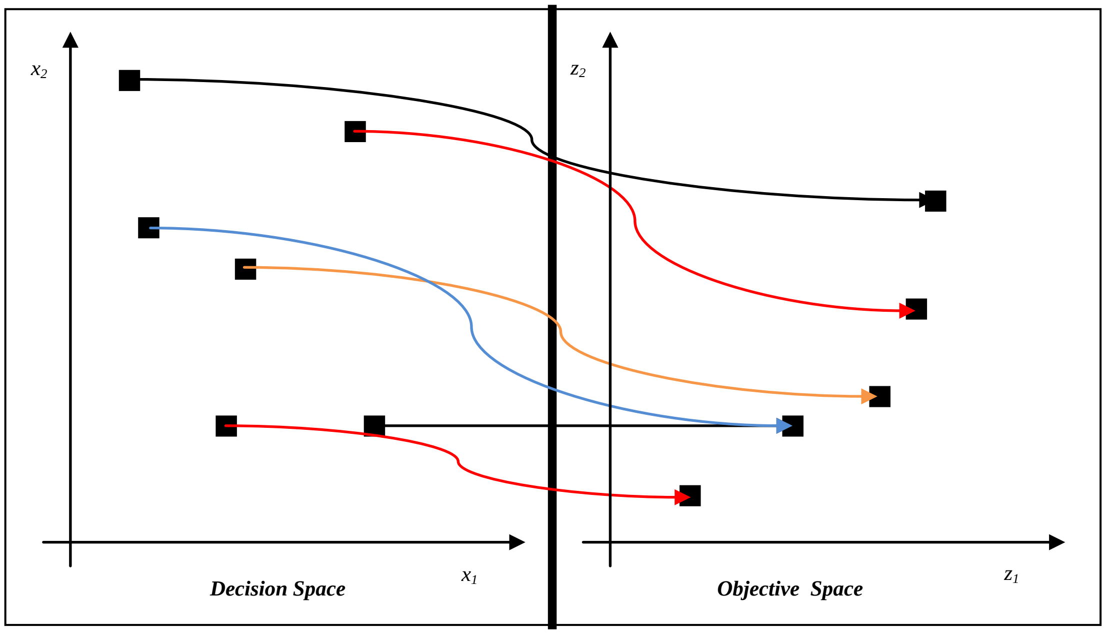


Figure 1 *The correspondence between decision space and objective space in a Multiobjective Optimization Problem.*

Pareto-optimal set is also called “Pareto-optimal Front” or “Efficient Frontier”. Several efficient solutions may correspond to a single nondominated solution which is the case for alternate optima. Hence the size of *distinct* nondominated solutions set is always less than or equal to the size of efficient solutions set. However, since alternative optimal solutions are nondominated to each other, the size of efficient solutions set is equal to the size of nondominated solutions set.

In Figure 1, all solutions in objective space are nondominated, assuming a problem minimizing z_1 and maximizing z_2. Hence, all solutions in decision space are efficient.

Moreover, number of distinct nondominated solutions is less than number of efficient solutions in Figure 1.

We also want to state some relevant concepts regarding Evolutionary Optimization (Deb, 2001);

Definition 4. *Nondominated Sorting* is a methodology that separates a set of solutions into smaller sets of solutions which are nondominated within themselves. Given a set of solutions S, in nondominated sorting, nondominated members of S are put in another set P_1. Then, nondominated solutions of set $S \setminus P_1$ are put in P_2. This operation continues until $S = \emptyset$. Hence, when routine is completed, all solutions are separated into fronts where none of solutions in a front dominate each other, whereas, fronts strictly dominate each other with P_1 being the Pareto-optimal set of S.

Nondominated sorting is used by algorithms that uses domination concept as fitness function.

Definition 5. *Niching Mechanism* is the methodology used to keep a diverse set of nondominated solutions in Evolutionary Algorithms. Counting number of neighboring solutions and average distance between neighboring solutions are two diversity mechanisms used in literature.

Without such a diversity keeping mechanism (i.e., niching mechanism), most of algorithms fail to generate a uniformly distributed range of solutions.

Nondominated Sorting is the primary selection criterion in most of the recent MOGAs, whereas, Niching Mechanism is the secondary selection criterion.

2.4 MOGA Literature

Deb (2001, p. 80) reports that the concept of a genetic algorithm was first conceived by John Holland of the University of Michigan, Ann Arbor. Genetic Algorithms (GAs), which are population based metaheuristics, simulate the theory of evolution. The process starts with generation of an initial population of solutions. Initial population may be randomly created or seeded with heuristic solutions. In GAs each solution has a genotype and a phenotype. Genotype is the decoding schema of solutions. A solution may be represented as a binary string or as a real valued string, called "chromosome". Also each solution (i.e., chromosome) has a fitness value (e.g., corresponding objective function value) which corresponds to phenotype of the chromosome. Fitness for a solution represents the desirability of that solution and is crucial for its survival and reproduction. Chromosomes are crossed together to generate new solutions (crossover operator) and then generated solutions are mutated (mutation operator). These two operators are applied with some appropriate probabilities depending on the structure of the problem considered. After generation of new solutions, called "child population", their fitness values are assigned. Finally, based on fitness values, a child either replaces its parent or is discarded (i.e., natural selection). After updating parent population, process is repeated. Four parameters are to be set in a GA; crossover probability, mutation probability, population size, and number of generations to be performed.

GAs are well-suited for multiobjective optimization (MO) due to their population based optimization structures (see Coello, 2006, for a recent discussion on MOGA). Aim in multiobjective optimization is to find a set of nondominated solutions. Since in a genetic algorithm we always work with a population of solutions, result of a GA is naturally a set of nondominated solutions (see Chinchuluun and Pardalos, 2007, for a recent discussion on Multiobjective Optimization). Hence, convergence to the Pareto-optimal front and ability to generate a diverse set of nondominated solutions are the most important features of MOGAs and research on these properties of MOGAs is very active. In fact, design of a MOGA itself is a multiobjective problem in nature. Many different strategies are used in literature to design good quality algorithms that can satisfy both goals adequately. Deb (2001, p. 161-273) classifies multiobjective algorithms into two groups: Non-Elitist MOGAs and Elitist MOGAs.

Non-Elitist MOGAs:

These algorithms are the first proposed simple MOGAs and they do not use any elite-preserving operator. VEGA, VOES, WBGA, RWGA, MOGA, NSGA, NPGA, PPES, DSGA, DRLA, NCGA, MNA, and NGA are algorithms of this class suggested in literature. These algorithms are easy to understand and implement. However they have found very good results in their original studies.

Elitist MOGAs:

In elite-preserving algorithms, elites have an opportunity to be directly included in next generations. No matter how elitism is introduced in these algorithms, they ensure that fitness of population-best solution does not deteriorate. REMEA, NSGA-II (Deb et al., 2000), DPGA, SPEA, SPEA2 (Zitzler et al., 2001), TDGA, PAES, NSAGA, PCGA, MμGA, ERMOCS, ε-MOEA (Deb et al., 2005), SMS-EMOE (Beume et al., 2007), MOTGA (Alves and Almedia, 2007), and FWEA (Soylu and Köksalan, 2007) are elitist MOGAs proposed in literature. Many of these algorithms use external populations to preserve elite solutions. They exploit elitism in different ways.

Refer to Deb (2001) for a more detailed study on MOGAs.

2.5 Literature Review on Preference Incorporation in MOGAs

No matter how many nondominated solutions exist for a decision making problem, DM has to choose only one of these solutions. Hence, the only tie breaking rule is preference. We refer reader to the work of Rachmawati and Srinivasan (2006) for a survey on preference articulation in MOGAs.

Three classes of methods exist for preference incorporation in MOGAs: a priori methods, interactive methods, and a posteriori methods.

A Priori methods: In this class preferences of DM is formulated prior optimization. Preferences are incorporated in fitness function and optimization is done accordingly. Weighted sum and Lexicographic approaches are examples of this method. However, this method is difficult to use due to lack of sufficient problem knowledge and non-convexity issues.

Interactive Methods: Few parameters are fixed prior optimization and rest is tailored during the search process based on guidance of DM. Difficulties faced when using this method are the extensive effort required from DM and the more apparently observed disagreement in the case of more than one decision maker.

A Posteriori Methods: In this method optimization and decision making processes are separated. DM selects her/his most preferred solution from set of solutions generated. Some computational problems are avoided by delaying decision making process to post-optimization stage, however, new problems arises. Finding a set of solutions with varying tradeoffs and navigating through Pareto front may be difficult. Moreover, as number of objectives increases, size of pareto front increases exponentially and algorithms lack an effective selection force.

Deb (2001, p. 162) argues that it is better to find a set of Pareto-optimal solutions first and then choose one solution from this set using some other higher-level information. This approach is reported as an *ideal* approach. After finding a set of nondominated solutions, if preference knowledge about the trade-off among objectives is known, one of these nondominated solutions could be chosen based on this preference information. This method, at least, gives a user an overall perspective of other possible optimal solutions that the underlying multiobjective optimization problem offers before choosing one of the nondominated solutions.

A posteriori method is used in this paper due to the following reasons. In TSPP we have only two objectives, hence, the problem of scalability (i.e., exponential increase in the size of Pareto front as the number of objectives increases) is not a severe issue for our case. Secondly, for practically large problem sizes, we are able to find extreme solutions and a set of solutions that uniformly sketches Pareto front with solution methodology proposed in this paper. Finally, we also propose a tool for navigating Pareto front without putting an extra burden on DM in Chapter 7.

CHAPTER 3

A MOGA FOR TSPP

Since search space increases exponentially for NP-complete problems, a failure at large problem sizes is inevitable for exact algorithms. We observe that none of exact algorithms proposed in literature are able to solve problems with more than 150 nodes for TSPP, which empower our former statement. Hence, usage of heuristics that can approximate optimal solution in single objective case or Pareto-optimal front in multiobjective case within a reasonable error is quite often, natural, and necessary.

In this chapter we examine an existing MOGA, NSGA-II, and summarize modifications we propose for this algorithm. In section 3.1 original NSGA-II algorithm is reviewed, in section 3.2 we propose first modification for NSGA-II, Modified NSGA-II is presented in section 3.3, section 3.4 summarizes performance metrics used, and finally we set parameters for the Modified NSGA-II and present preliminary results in section 3.5.

3.1 NSGA-II

In this work we use one of state-of-the-art metaheuristics: NSGA-II. NSGA-II is proposed by Deb et al. (2000) as a Multiobjective Genetic Algorithm. Algorithm uses nondominated sorting mechanism and an explicit diversity preserving mechanism. Niching mechanism used in algorithm is based on average distance of neighboring solutions in all objectives. In NSGA-II we use a binary chromosome representation. The length of chromosome is equal to *n*, which is the number of cities considered. A “1” at location *j* on chromosome means j^{th} city is included in the tour, “0” otherwise. We use Lin-Kernighan TSP heuristic (Concorde, 2008) to find cost based objective for a given chromosome (i.e., for a given subset of cities). Cost for a tour is generally the length of the tour. For some problem instances length between two cities is calculated as the Euclidean Distance between the two cities, for some instances Geometric Distance is used, and for a small number of instances ATT Distance metric is used. Profit based objective is simply calculated by adding up profits corresponding to nodes included in the subset defined by chromosome. Table 1 demonstrates how objective function values are calculated for a given chromosome.

Table 1 *Calculation of cost and profit for a chromosome.*

NODES	1	2	3	4	5	6	7	8	9	10
COORDINATES	[59,68]	[70,86]	[26,64]	[15,75]	[64,15]	[77,62]	[85,38]	[46,57]	[57,9]	[89,51]
PROFITS	0	42	9	43	84	20	34	1	75	75
CHROMOSOME	**1**	**1**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**0**
COST:	221	For sequence generated with Lin-Kernighan Heuristic: **1-2-4-9-8-1**								
PROFIT:	161	=**1***0 + **1***42 + **0***9 + **1***43 + **0***84 + **0***20 + **0***34 + **1***1 + **1***75 + **0***75								

In the literature of TSPP, nodes are generally treated as “cities”. This is mainly due to terms used in TSP literature. Since we do not have to visit all cities, a depot city is defined for TSPPs. Generally profit assigned to depot city is zero. All subsets considered in TSPP should include depot city. However, this constraint does not make problem more difficult since one could easily force any algorithm to include depot in the constructed tour by assigning a very large profit to it.

In NSGA-II we generate initial population randomly but we always include extreme solutions. Extreme solutions are easy to find: staying at the depot is the solution with least cost and visiting all cities is the solution with largest profit. We include these two extreme solutions as two binary chromosomes where in one of the chromosomes all genes take a value of 1, and in the other chromosome only the gene corresponding to the depot city takes a value of 1. Figure 2 shows a flow chart for NSGA-II.

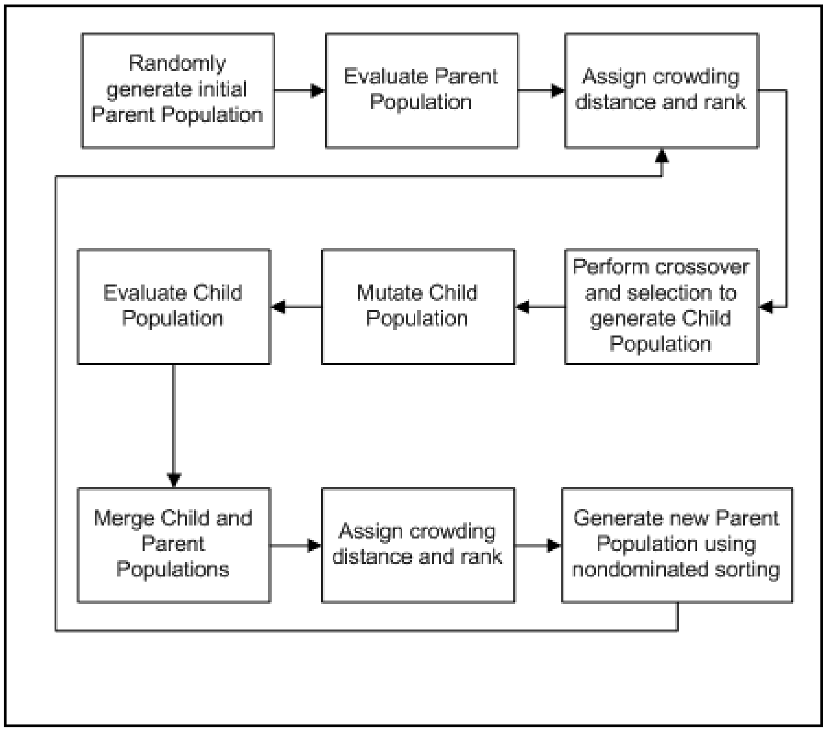

Randomly generate initial Parent Population
Evaluate Parent Population
Assign crowding distance and rank
Perform crossover and selection to generate Child Population
Mutate Child Population
Evaluate Child Population
Merge Child and Parent Populations
Assign crowding distance and rank
Generate new Parent Population using nondominated sorting

Figure 2 *Flowchart for NSGA-II.*

To summarize the flow in Figure 2, algorithm starts with creation of random initial population. However, depot city is included in all members of population (i.e., in all subset of cities generated). The two extreme solutions, minimum cost and maximum profit cases, are also included in population. Then cost and profit values are assigned to population as explained in Table 1. Based on objective values, nondominated sorting and niching mechanism is used to sort members of population from best to worst. Then, based on tournament selection, two-point crossover operator is executed to generate child population. Child population is mutated according to bit-wise mutation operator with given mutation probability as the next step. Afterward objective values for members of child population are calculated. Then, child and parent populations are merged and rank and crowding distance is assigned to all members of merged population. Finally, parent population for next generation is selected from the merged population based on rank and crowding distance of solutions.

The pseudo code for original NSGA-II is given in Appendix A.

3.2 First Improvement for Population Size Limit of NSGA-II

In MOGAs, as initial population evolves, the number of nondominated solutions increases. When the number of these solutions exceeds population size, a niching schema is used to select individuals from less crowded regions so as to escape local optima and access less searched regions of the search space. Thus, some nondominated solutions are cut-off from population. The number of such wasted nondominated solutions increases exponentially in TSPP as problem size *n* increases due to combinatorial number of subsets of cities possible. However, in NSGA-II, number of nondominated solutions returned eventually is at most equal to the population size.

The first and most apparent solution to the problem stated is to keep all nondominated solutions generated during execution of NSGA-II. We modified NSGA II in the following sense. We generate an external population of nondominated individuals generated during the execution of NSGA-II. Figure 3 shows a flow chart for proposed improvement.

As it can be seen in Figure 3, execution of the original algorithms is not modified. The only change is inclusion of an archive population which we call “Elite Pop”. Elite Pop preserves all nondominated solutions generated. Hence, Elite Pop returned by algorithm always covers final population returned.

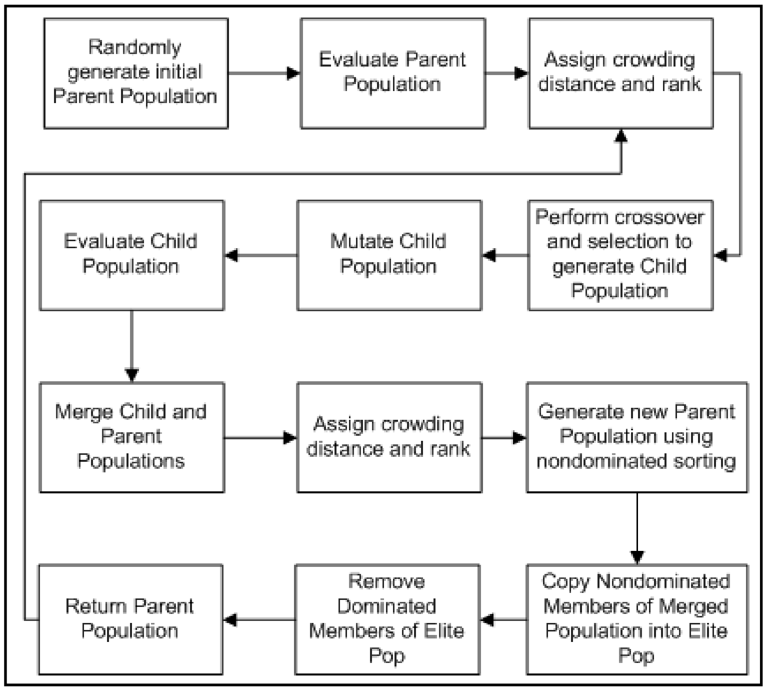

Randomly generate initial Parent Population
Evaluate Parent Population
Assign crowding distance and rank
Evaluate Child Population
Mutate Child Population
Perform crossover and selection to generate Child Population
Merge Child and Parent Populations
Assign crowding distance and rank
Generate new Parent Population using nondominated sorting
Return Parent Population
Remove Dominated Members of Elite Pop
Copy Nondominated Members of Merged Population into Elite Pop

Figure 3 *Flow chart for proposet improvement to NSGA-II.*

3.3 Final Improvement: The Modified NSGA-II (mNSGA-II)

One further modification we propose is to use nondominated solutions archived in Elite Pop to modify Parent Population in each iteration of the algorithm. To apply this modification, we modified NSGA II, called mNSGA II, in the following sense. We define two external populations of nondominated individuals consisting of solutions not accepted to parent population, and use them to update parent population in each iteration of algorithm. These external populations are named “Elite Pop 1” and “Elite Pop 2”. Elite Pop 2 is a temporary population which is reset at the end of each generation performed. On the other hand, Elite Pop 1 is a permanent population which preserves all nondominated solutions generated and not included in parent population. Hence, at the end of run, Elite Pop 1 and Parent Population are merged to generate the complete pareto front. Figure 4 shows a flow chart for NSGA-II.

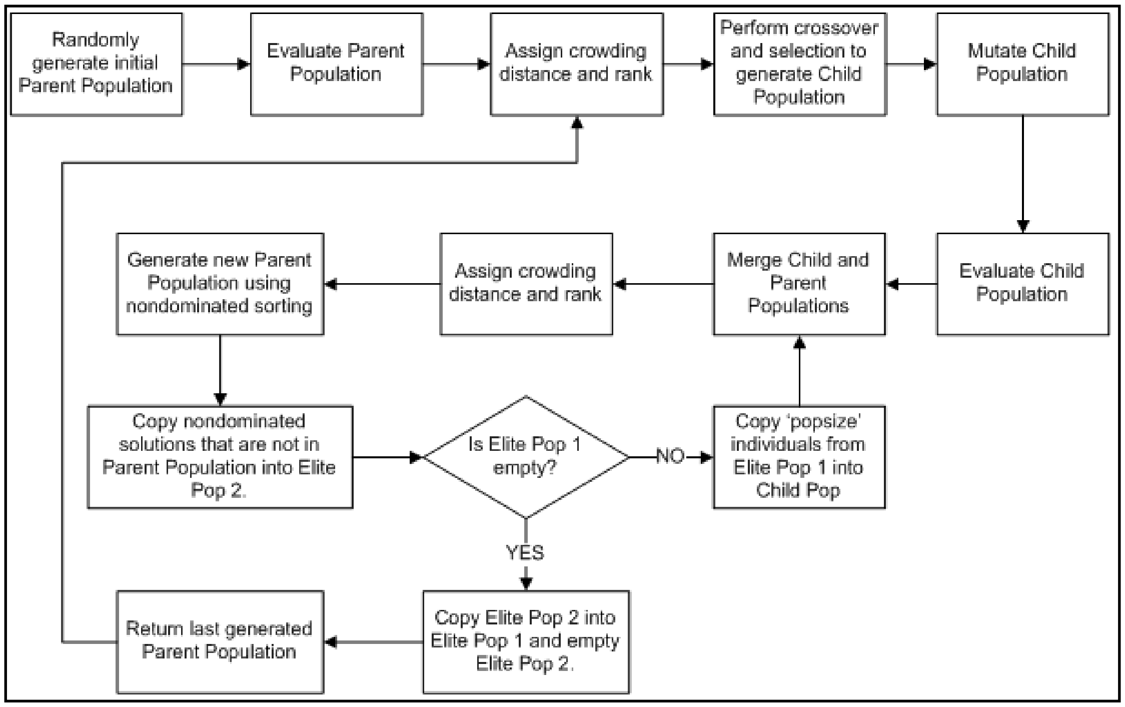

Randomly generate initial Parent Population
Evaluate Parent Population
Assign crowding distance and rank
Perform crossover and selection to generate Child Population
Mutate Child Population
Evaluate Child Population
Merge Child and Parent Populations
Assign crowding distance and rank
Generate new Parent Population using nondominated sorting
Copy nondominated solutions that are not in Parent Population into Elite Pop 2.
Is Elite Pop 1 empty?
NO
Copy 'popsize' individuals from Elite Pop 1 into Child Pop
YES
Copy Elite Pop 2 into Elite Pop 1 and empty Elite Pop 2.
Return last generated Parent Population

Figure 4 *Flowchart for NSGA-II.*

In Figure 4, the first nine steps are exactly the same as in original NSGA-II algorithm as given in Figure 2. After generation of the first parent population in each generation, mNSGA-II starts a loop as seen in Figure 4. First, nondominated solutions that were not included in generated parent population are copied into Elite Pop 2. Then, we check whether Elite Pop 1 is empty or not. If Elite Pop 1 is not empty, we copy "popsize" solutions from this population into child population. "Popsize" is the size of population which is set at the beginning of algorithm. After copying popsize solutions into child population, standard "merge, sort, and select" steps are performed and the loop is restarted. Loop is executed until all members of Elite Pop 1 are used. Then, we exit the loop, copy all individuals in Elite Pop 2 into Elite Pop 1, empty Elite Pop 2, and report last generated parent population for next generation.

The loop described here ensures that all nondominated solutions generated during execution of algorithm and not included in parent population are preserved. Moreover, preserved solutions interact with parent population at each generation and if they are re-included in parent population, they get a chance to be used in evolution process (i.e., they are used in crossover and mutation operators) and affect future generations.

The pseudo code for mNSGA-II is given in Appendix B.

3.4 Performance Metrics Used

To make comparisons, basically two metrics from MOGA literature are used; Hyper Volume (HV) and Generational Distance (GD) (Deb, 2001, p. 306-324).

For a given set of nondominated solutions, the hyper volume they enclose in objective space is the union of the volumes each point generates with respect to a reference point. The reference point can be the nadir point or estimation to it. In Figure 5, HV enclosed by a given set of nondominated solutions in biobjective case where both objectives are to be minimized is the sum of areas of blue rectangles. It can be shown that a set of nondominated solutions with better convergence and dispersion will always constitute equal or a larger HV.

Another metric is defined based on this argument; Hyper Volume Ratio (HVR). HVR shows the relative distance between two set of nondominated solutions or two Pareto fronts. It can be stated as;

$$\mathrm{HVR} = \frac{\mathrm{HV}_{\mathrm{Approximation}}}{\mathrm{HV}_{\mathrm{Optimal}}} \qquad (8)$$

(8) can be used to make a comparison between an approximation set and a Pareto-

optimal set of solutions. In this case HVR can take a maximum value of 1. Values larger than 0.95 would imply a very good approximation to the Pareto-optimal front.

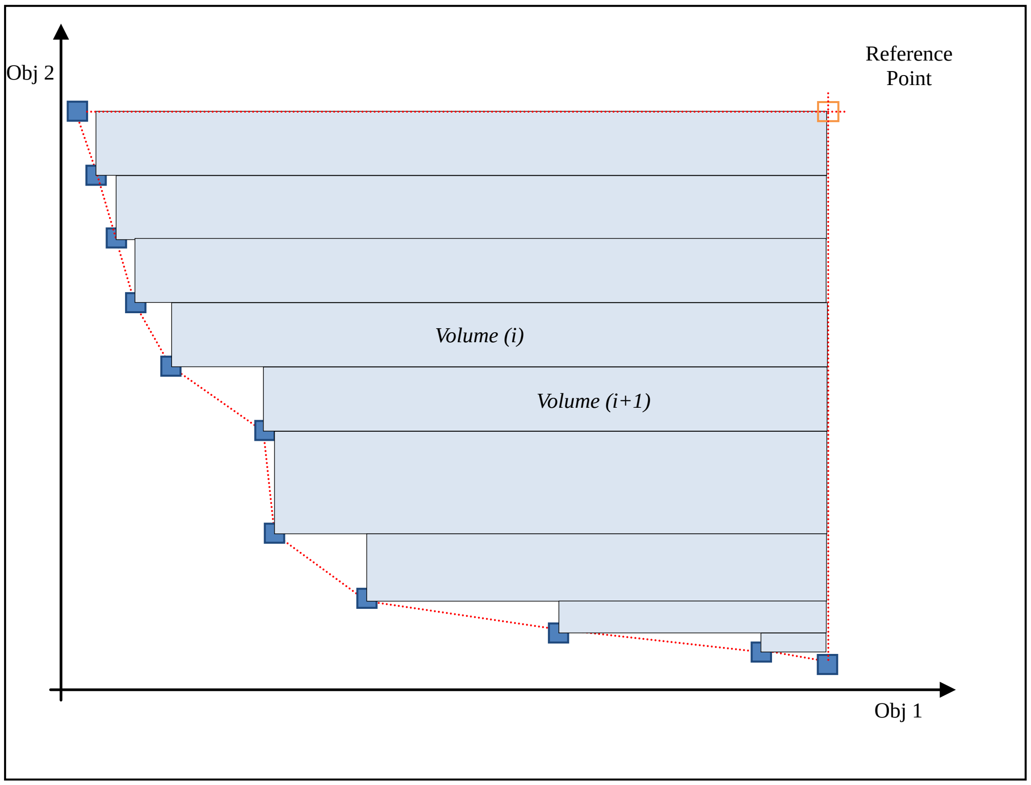


Figure 5 *HV enclosed by a given set of nondominated solutions.*

On the other hand, GD shows the mutual distance between a Pareto-optimal front and an approximation set to this front. First, for each point on Pareto-optimal front the nearest point on approximate front is found using a relevant distance metric. Then, GD is calculated as the average of these distances. Since such a metric requires knowledge of the range of objectives, it is more meaningful to define a metric that shows relative distance between two fronts. If we define Z_{opt}^{i} as ith point on Pareto-optimal front and Z_{heur}^{i} as corresponding nearest point on approximate Pareto front based on Euclidean distance, we define the following two measures of distance between two fronts;

$$GD_{cost}^{\%} = \frac{\sum_i \frac{Z_{heur}^{i,cost} - Z_{opt}^{i,cost}}{Z_{opt}^{i,cost}}}{n} * 100 \qquad (9)$$

$$GD_{prof}^{\%} = \frac{\sum_i \frac{Z_{opt}^{i,prof} - Z_{heur}^{i,prof}}{Z_{opt}^{i,prof}}}{n} * 100 \qquad (10)$$

$GD_{cost}^{\%}$ shows average percent deviation in cost objective, $GD_{prof}^{\%}$ shows average percent deviation in profit objective, and *n* is the size of Pareto-optimal front. These two metrics show relative distance of two Pareto fronts in terms of cost and profit objectives. However, any kind of objectives could be used with these metrics. We will refer to these metrics as Percent GD in cost objective and Percent GD in profit objective.

Several other metrics exist in literature: Error Ratio, Set Coverage Metric, Maximum Pareto Optimal Front Error, Spacing, Spread, Maximum Spread, Chi-Square-Like Deviation Measure, Attainment Surface Based Statistical Method, Weighted Method, and Nondominated Evaluation Metric. All these metrics try to measure convergence and dispersion properties of algorithms. Hence, we would like to select two metrics: one for convergence and one for dispersion properties of the algorithm of interest. Since two metrics are adequate to measure these two properties and since HV and GD give more insight to the algorithm of interest and more frequently used in literature, we used only these two metrics. We should also note that Percent GD is proposed for the first time in this study and aims to demonstrate the percent deviation from Pareto-optimal front.

3.5 Parameter Setting for mNSGA-II

Population Size: Since we deal with a combinatorial optimization problem, search space to be considered is intractable even for very small problem sizes. Note that a large population size is desirable even for small problem sizes. On the other hand, since finding the objective function values for each member of population is also a very time consuming process for TSPP due to the requirement for solving a TSP for each member of population, a small population size is desirable. The reasoning behind the desire of a small population size is to obtain a converged population in a reasonable time. If population size were very large, each generation of the algorithm would require a very long time and we would have to stop before the algorithm converges due to time considerations. Moreover, in NSGA-II population size determines the size of pareto front returned by algorithm. However, since in mNSGA-II all generated efficient solutions are kept in an external population, size of pareto front is independent from population size for mNSGA-II. Hence we decided to use a population size of 100. This selection is based on previous studies on best parameter settings for other algorithms, the time required to perform a single iteration in mNSGA-II, and the size of pareto front we target.

Crossover Probability: After defining metrics to be used, we first set crossover and mutation probabilities. To allow adequate inheritance of good schemata, the original crossover probability of 0.9 in NSGA-II is kept unchanged.

Mutation Probability: Table 2 shows hyper volumes enclosed by generated Pareto fronts for test problem EIL76 at several mutation probability levels for mNSGA-II. A generation number limit of 250 is used. We observed that high mutation probabilities cause loss of good quality solutions and very low ones result in premature convergence. Therefore we use 0.02 as the mutation probability, P_m.

Table 2 *Hyper volumes at various mutation probabilities.*

	P_m	Hyper Volume
EIL76	0.0125	399,402
	0.0200	420,114
	0.0500	408,121
	0.1000	395,180

Generation Number Limit: Convergence to the Pareto-optimal front gets difficult and probability of being stuck at a local optima increases rapidly as *n* increases because of exponential increase of search space. Thus, we used different generation number limits for a population size of 100. After various experiments we set the limit to 1,000 for $n<100$, to 5,000 for $100 \leq n<200$, and to 10,000 for $n \geq 200$. These limits are defined considering convergence of algorithm at various problem sizes (independent from difficulty of TSP instances solved) and time requirements.

3.6 Evaluation of Proposed Modifications for NSGA-II

Firs we conducted a preliminary experiment for NSGA-II and proposed improvement in section 3.2. Table 3 shows results of experiment, where a generation number limit of 250 and a population size of 100 is used. As it can be seen in Table 3, number of nondominated solutions for NSGA-II never exceeds 100, which corresponds to the population size used. On the other hand, with the first improvement we are able to find a more populated pareto front. Also hyper volumes are better for proposed algorithm on all six problem instances and accordingly HVR values are less than 1. However, HVR are very close to 1 since we do not modify execution of algorithm.

To visually show and exemplify pareto fronts generated, in Figure 6 we present Pareto front generated for REINELT1084 instance which include 1084 cities.

Table 3 *Results for first modification to NSGA-II*

Problem Instance	**# of Nondominated solutions generated**		**Hyper Volume**		**HVR**
	NSGA-II	**Proposed**	**NSGA-II**	**Proposed**	
EIL30	66	119	2,616,200	2,630,350	0.9946
CMT101	92	147	444,830	448,370	0.9921
F135	98	219	6,479,901	6,535,792	0.9914
GIL262	97	223	13,862,466	14,003,940	0.9899
E484	99	181	3,986,497	4,009,011	0.9944
REINELT1084	99	256	4,626,044,046	4,684,320,515	0.9876

Consequently, as it can be observed from Figure 6, with proposed algorithm we are able to find a pareto front in terms of convergence, dispersion, and number of nondominated solutions.

Secondly, fixing population size to 100 and performing 20 replications, we conducted another preliminary experiment using NSGA-II and mNSGA-II over 3 instances from TSP literature with problem sizes of 100, 200, and 300 cities. Table 4 presents average hyper volumes enclosed by Pareto fronts generated for both algorithms over 20 replications. mNSGA-II is better in all cases, except for one instance shown in red ink.

Moreover, the size of Pareto front generated by the original algorithm for three problems is always 100, whereas, it is 500, 300, and 700 on average for mNSGA II for three problems respectively.

A preliminary analysis is also conducted to find out effect of usage of a heuristic and an exact method on quality of generated pareto front. Lin-Kernighan is used as heuristic and Concorde TSP solver used to obtain exact results.

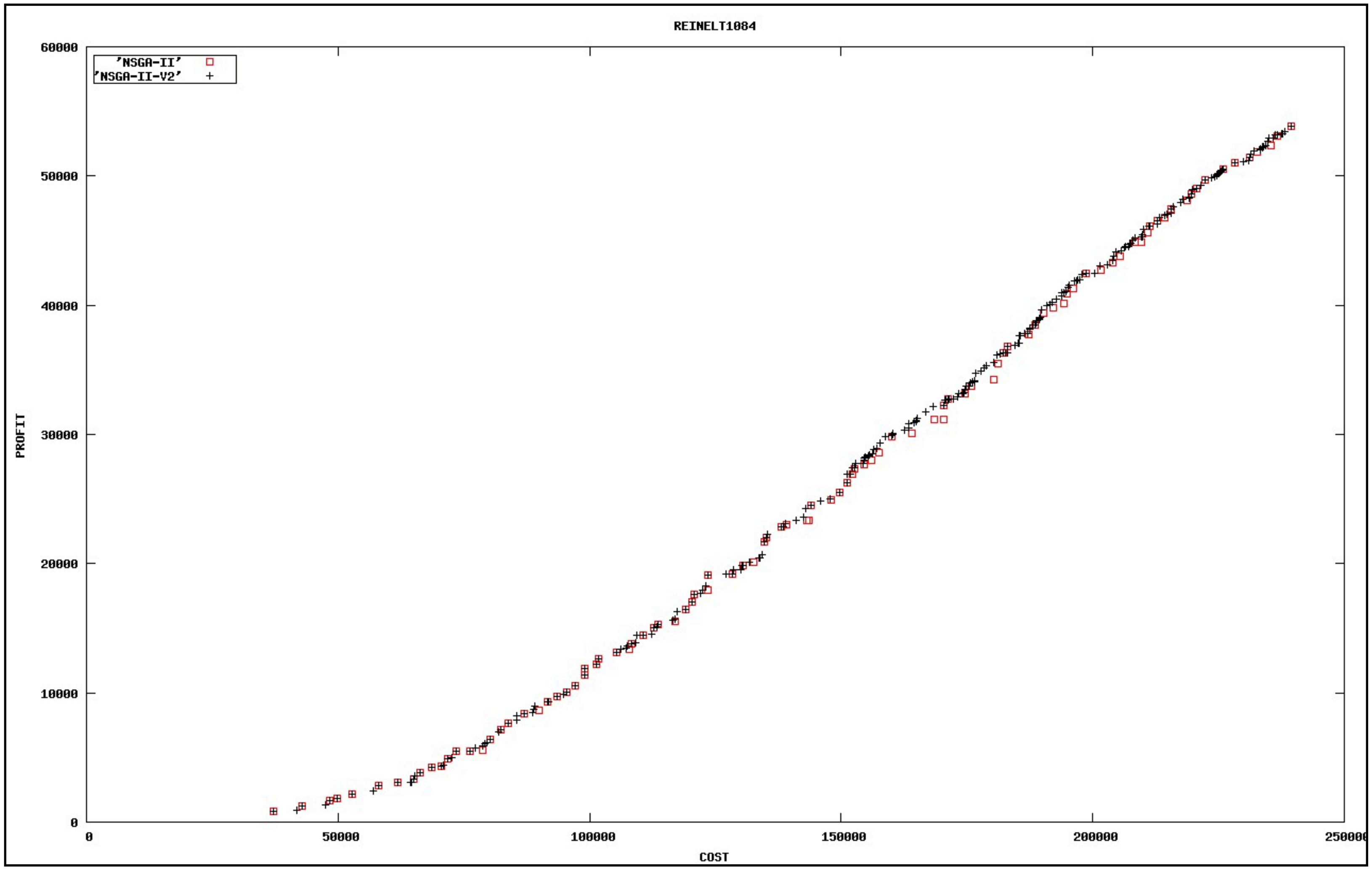


Figure 6 *Pareto fronts generated for REINELT1084 by NSGA-II and Proposed algorithms.*

Table 4 *Hyper volumes at various generations for three test problems with 20 replications.*

	Generation Number	NSGA II	mNSGA II
CMT121	0	96,396	96,396
	100	324,125	326,456
	200	350,476	349,531
	300	360,784	365,237
	400	366,324	372,234
	500	369,006	376,245
E200	0	441,148	441,148
	100	1,193,185	1,201,742
	200	1,259,501	1,262,951
	300	1,290,893	1,304,978
	400	1,314,062	1,331,932
	500	1,328,356	1,351,339
LIN318	0	99,159,717	99,159,717
	100	279,237,849	287,690,064
	200	297,581,093	300,516,775
	300	305,733,034	311,836,548
	400	312,160,786	319,694,935
	500	316,361,711	325,594,095

Table 5 shows hyper volumes for three test problems and Figures 7 and 8 show generated pareto fronts when solving TSP instances under mNSGA-II using exact method and Lin-Kernighan heuristic. For experiments a generation number limit of 250 and a population size of 100 are used.

As it can be seen from Table 5 results generated are very close to each other. Figures 7 and 8 also suggest that the two methods generate equally good solutions. For OP33 instance results are almost the same. For EIL76, in Figure 8, algorithms perform better in different segments of generated pareto front. Results indicate that for moderate problem sizes Lin-Kernighan heuristic and exact method are nondominated to each other. For larger problem sizes usage of exact method becomes impossible due to large computational requirements. Thus, we will use Lin-Kernighan heuristic to solve TSP instances faced during execution of mNSGA-II.

Table 5 *Hyper volumes for three test problems.*

	Hyper Volume	
	Exact	**Lin-Kernighan**
OP33	37,860	37,860
EIL51	194,001	194,366
EIL76	425,301	430,624

In summary, the second improvement explained in section 3.3 is a more general improvement and it includes the first improvement explained in section 3.2. Also results shown in Table 4 show that mNSGA-II is better than original NSGA-II algorithm. Hence, we will use only mNSGA-II to compare our results with results in literature. Moreover, due to reasons explained above, we will use Lin-Kernighan heuristic in our computational experiments which are summarized in the next chapter.

3.7 A Preliminary Analysis on Robustness of Modified NSGA-II

Since mNSGA-II is a heuristic, results generated by this heuristic will vary based on initial conditions selected. Hence, performance of a heuristic is effectively measured if adequate number of replications is conducted. If the results of experiments prove that heuristic is robust and correlation between initial conditions and the results of runs is weak, then, results of a single run could be assumed to be adequate to measure performance of heuristic studied.

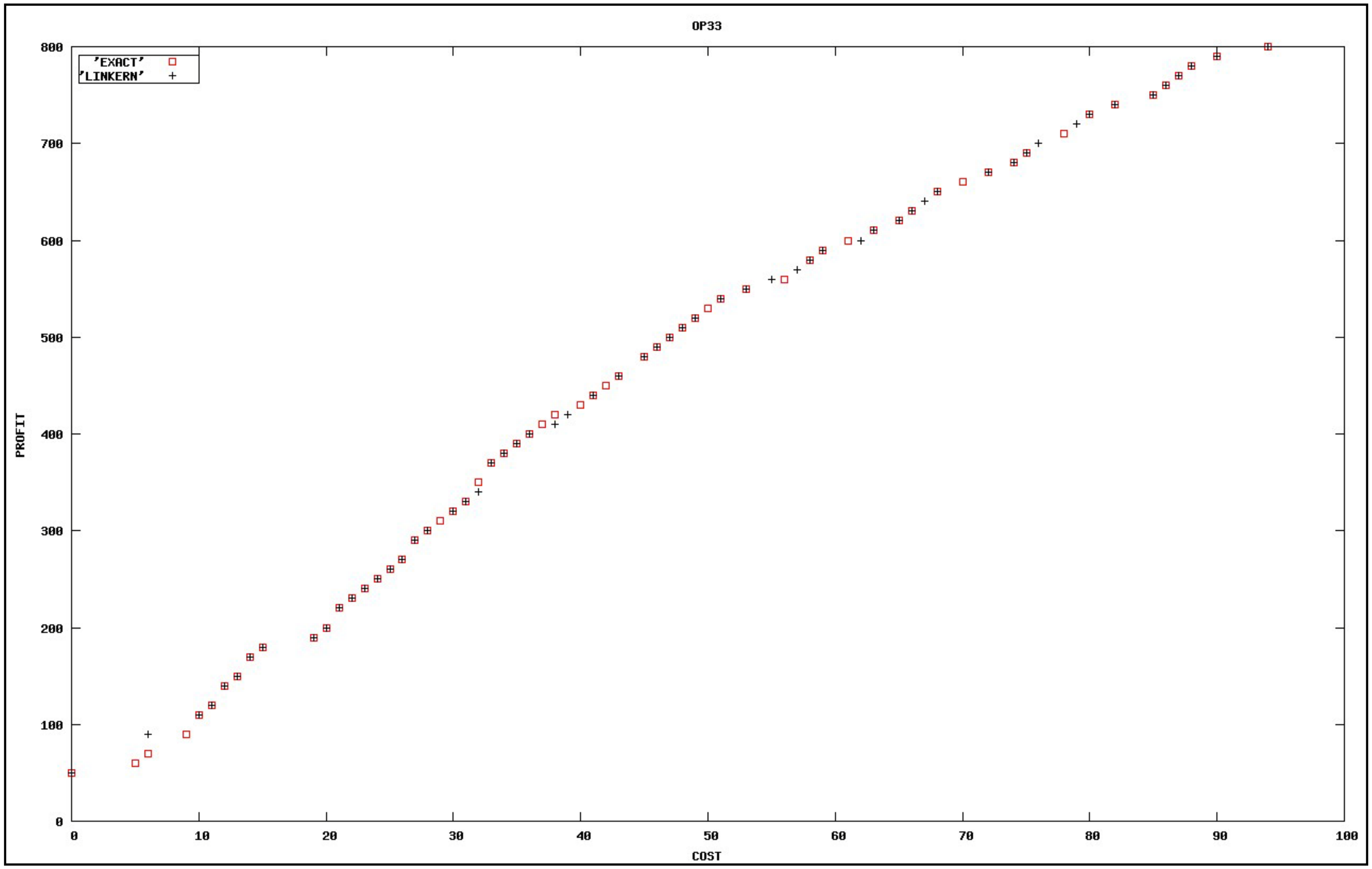


Figure 7 *Results for OP33.*

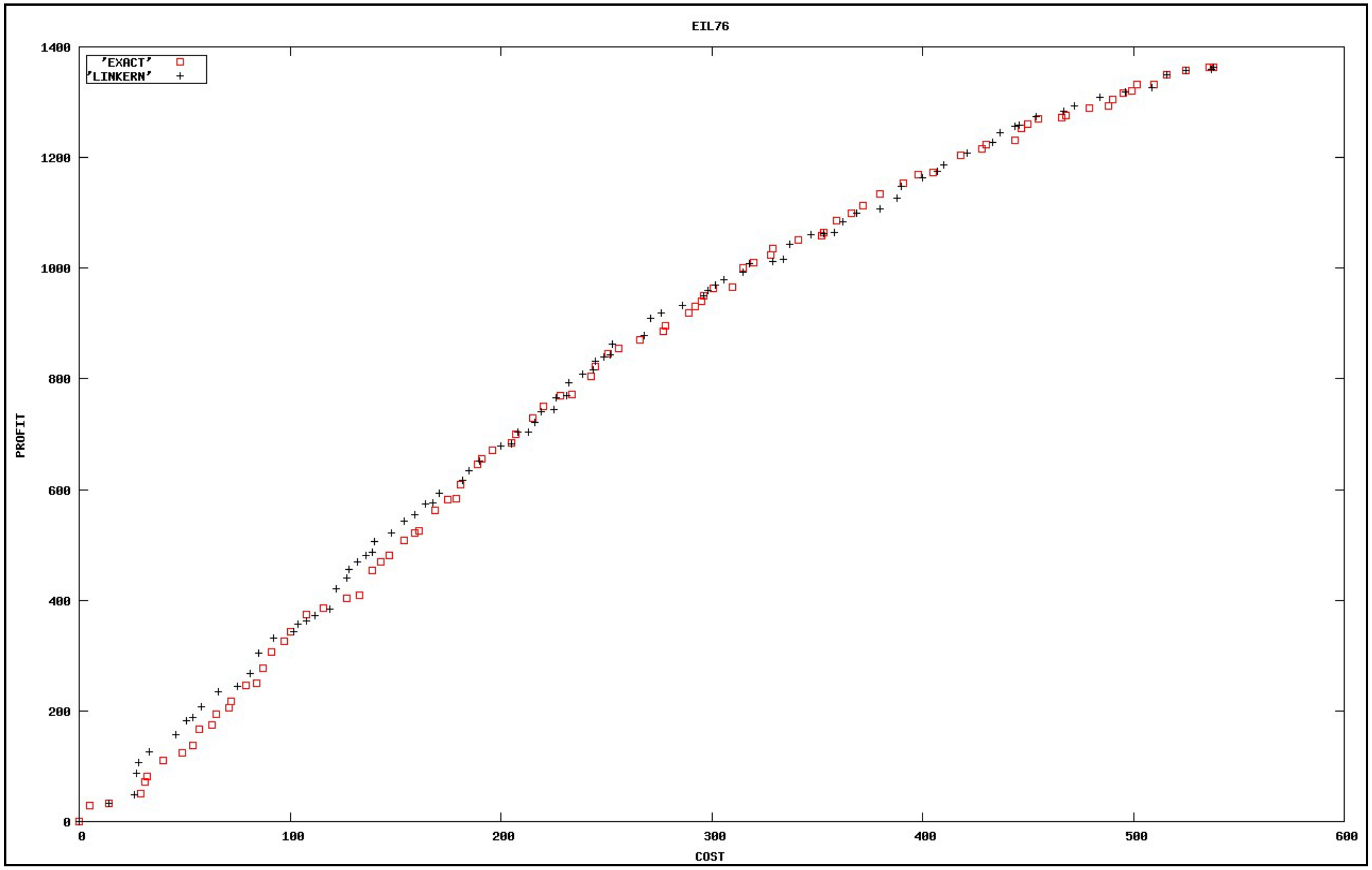


Figure 8 *Results for EIL76.*

For mNSGA-II, we generate initial parent population randomly where we control randomness using initial seed value for random number generator it uses. To observe the effect of randomness in initial population on results of simulation, we have conducted a preliminary experiment. The aim of this experiment is to observe robustness of the modified heuristic and decide whether it is required to conduct more than one simulation for problems with long runtimes.

Tables 6, 7, and 8 give results for VRP and TSP instances including less than 100 cities. We have conducted 10 runs for each problem instance. Tables 6, 7, and 8 summarize averages for these instances. Demand data for VRP instances is accepted as profits for cities. For TSP instances we have generated two types of profits randomly, named as TSP-1 and TSP-3. In the next chapter we provide details of this profit assignment schema.

In Tables 6, 7, and 8 we provide minimum, average, maximum, and standard deviation for parameters considered. For each problem instance, we give number of efficient solutions generated (# of sol), hyper volume enclosed by generated pareto front (HV), hyper volume ratio (HVR), Percent GD in cost objective (PGD-cost(%)), Percent GD in profit objective (PGD-prof(%)), CPU time for modified NSGA-II (Time NSGA-II (sec)), and CPU time for Lin&Kernighan Heuristic (Time L&K (sec)). Considering standard deviation for number of efficient solutions generated, the largest deviation is 22 solutions for EIL76 where 339 solutions are generated on average as it can be seen in Table 8. Considering hyper volume, standard deviation is less than 1% of average hyper volume in all cases and it is less than 1‰ in most of the results. HVR values show that in all cases results found are very close to each other, less than 5‰ in all cases. Deviation in Percent GD in cost and profit objectives is always less than 6‰. This means that pareto fronts found are very close to each other and there is no major deviation from one simulation to another. CPU times reported show larger deviation when compared to results summarized up to now.

Table 6 *Preliminary experiments on robustness of mNSGA-II for VRP instances.*

		Min	Ave	Max	Stdev
EIL22	**# of sol**	67	69	70	1
	HV	3,286,400	3,292,630	3,297,900	5,508
	HVR	0.9964	0.9983	0.9999	0.0017
	PGD-cost (%)	0.02	0.19	0.41	0.14
	PGD-prof (%)	0.01	0.04	0.07	0.02
	Time NSGA-II (Sec.)	75	91	116	14
	Time L&K (Sec.)	31	50	116	25
EIL23	**# of sol**	74	75	77	1
	HV	3,403,618	3,403,997	3,404,600	241
	HVR	0.9997	0.9998	1.0000	0.0001
	PGD-cost (%)	0.06	0.07	0.12	0.02
	PGD-prof (%)	0.00	0.00	0.01	0.00
	Time NSGA-II (Sec.)	67	82	100	11
	Time L&K (Sec.)	52	80	162	33
EIL30	**# of sol**	126	129	132	2
	HV	2,689,973	2,690,377	2,690,550	184
	HVR	0.9997	0.9999	1.0000	0.0001
	PGD-cost (%)	0.01	0.03	0.10	0.03
	PGD-prof (%)	0.00	0.01	0.02	0.01
	Time NSGA-II (Sec.)	82	89	107	8
	Time L&K (Sec.)	538	741	886	104
EIL33	**# of sol**	159	161	162	1
	HV	6,564,433	6,565,384	6,565,820	386
	HVR	0.9998	0.9999	1.0000	0.0001
	PGD-cost (%)	0.01	0.07	0.09	0.03
	PGD-prof (%)	0.00	0.01	0.02	0.00
	Time NSGA-II (Sec.)	84	95	114	9
	Time L&K (Sec.)	348	551	730	104
EIL51	**# of sol**	226	235	245	5
	HV	197,504	197,564	197,597	29
	HVR	0.9993	0.9996	0.9998	0.0001
	PGD-cost (%)	0.03	0.05	0.08	0.01
	PGD-prof (%)	0.02	0.03	0.04	0.01
	Time NSGA-II (Sec.)	118	140	177	20
	Time L&K (Sec.)	247	357	474	65
EIL76	**# of sol**	307	328	345	12
	HV	438,783	440,991	441,891	1,038
	HVR	0.9907	0.9957	0.9977	0.0023
	PGD-cost (%)	0.28	0.58	1.23	0.32
	PGD-prof (%)	0.11	0.16	0.23	0.03
	Time NSGA-II (Sec.)	153	178	221	21
	Time L&K (Sec.)	750	1,009	1,240	137

Table 7 *Preliminary experiments on robustness of mNSGA-II for TSP-1 instances.*

		Min	Ave	Max	Stdev
EIL51	**# of sol**	82	91	117	11
	HV	11,257	11,405	11,426	52
	HVR	0.9850	0.9979	0.9998	0.0046
	PGD-cost (%)	0.02	0.12	0.56	0.16
	PGD-prof (%)	0.00	0.14	1.45	0.46
	Time NSGA-II (Sec.)	73	160	226	46
	Time L&K (Sec.)	395	589	838	152
BERLIN52	**# of sol**	52	52	52	0
	HV	246,685	246,892	246,966	105
	HVR	0.9988	0.9997	1.0000	0.0004
	PGD-cost (%)	0.01	0.07	0.16	0.06
	PGD-prof (%)	0.00	0.04	0.24	0.08
	Time NSGA-II (Sec.)	134	162	202	21
	Time L&K (Sec.)	2,159	2,721	4,279	685
ST70	**# of sol**	94	100	110	5
	HV	26,057	26,319	26,419	105
	HVR	0.9853	0.9952	0.9990	0.0040
	PGD-cost (%)	0.15	0.24	0.40	0.07
	PGD-prof (%)	0.00	0.51	1.60	0.49
	Time NSGA-II (Sec.)	167	208	268	36
	Time L&K (Sec.)	1,217	1,465	2,016	271
EIL76	**# of sol**	110	132	152	12
	HV	22,675	22,727	22,824	45
	HVR	0.9911	0.9934	0.9976	0.0020
	PGD-cost (%)	0.42	0.57	0.68	0.09
	PGD-prof (%)	0.14	1.10	1.79	0.51
	Time NSGA-II (Sec.)	118	155	247	46
	Time L&K (Sec.)	1,072	1,579	3,227	739
PR76	**# of sol**	76	78	80	1
	HV	4,627,952	4,636,060	4,647,299	6,063
	HVR	0.9920	0.9938	0.9962	0.0013
	PGD-cost (%)	0.34	0.40	0.52	0.06
	PGD-prof (%)	0.55	1.04	1.44	0.24
	Time NSGA-II (Sec.)	163	206	268	30
	Time L&K (Sec.)	2,844	3,455	4,409	510
RAT99	**# of sol**	116	126	138	7
	HV	59,658	60,012	60,338	188
	HVR	0.9812	0.9870	0.9924	0.0031
	PGD-cost (%)	0.38	0.46	0.56	0.06
	PGD-prof (%)	0.52	1.31	1.89	0.40
	Time NSGA-II (Sec.)	180	217	251	20
	Time L&K (Sec.)	1,146	1,270	1,412	87

Table 8 *Preliminary experiments on robustness of mNSGA-II for TSP-3 instances.*

		Min	Ave	Max	Stdev
EIL51	**# of sol**	213	246	263	15
	HV	534,607	537,148	538,709	1,277
	HVR	0.9913	0.9960	0.9989	0.0024
	PGD-cost (%)	0.15	0.55	1.03	0.25
	PGD-prof (%)	0.14	0.25	0.36	0.08
	Time NSGA-II (Sec.)	142	171	192	16
	Time L&K (Sec.)	350	440	524	54
BERLIN52	**# of sol**	411	417	425	5
	HV	7,240,064	7,312,420	7,342,129	38,058
	HVR	0.9857	0.9956	0.9996	0.0052
	PGD-cost (%)	0.05	0.10	0.21	0.05
	PGD-prof (%)	0.09	0.65	1.80	0.60
	Time NSGA-II (Sec.)	159	176	197	13
	Time L&K (Sec.)	1,501	1,783	2,968	423
ST70	**# of sol**	397	422	454	20
	HV	1,276,653	1,287,499	1,295,376	6,228
	HVR	0.9804	0.9887	0.9947	0.0048
	PGD-cost (%)	0.61	1.48	2.48	0.58
	PGD-prof (%)	0.23	0.35	0.49	0.09
	Time NSGA-II (Sec.)	165	196	213	16
	Time L&K (Sec.)	946	1,097	1,213	95
EIL76	**# of sol**	314	339	389	22
	HV	1,144,746	1,153,412	1,159,252	4,869
	HVR	0.9823	0.9898	0.9948	0.0042
	PGD-cost (%)	0.78	1.43	2.20	0.49
	PGD-prof (%)	0.30	0.38	0.46	0.05
	Time NSGA-II (Sec.)	170	208	239	23
	Time L&K (Sec.)	680	918	1,316	182

We also want to note that the major portion of CPU time for a simulation comes from the routing decision, the CPU time for Lin&Kernighan Heuristic. Figure 9 shows percentages of average CPU times for mNSGA-II and Lin&Kernighan heuristic.

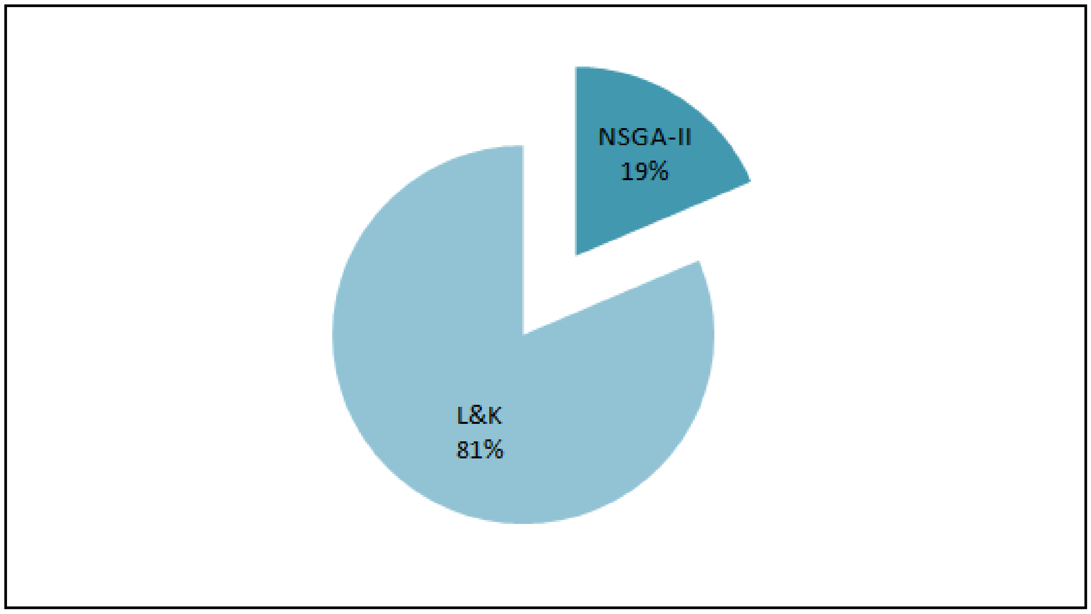


Figure 9 *Percentage of CPU times for mNSGA-II and Lin&Kernighan.*

As it can be seen from Figure 9, only a small portion of CPU time for simulations comes from mNSGA-II. Thus, if a better algorithm could be found to solve routing problem, then, mNSGA-II would also improve in terms of runtime.

CHAPTER 4

COMPUTATIONAL RESULTS

In this chapter we present computational results of mNSGA-II and compare them to the benchmark results available in literature. In section 4.1 selection of problem instances is explained and results regarding selected problems are given in section 4.2.

4.1 Selection of Test Problems

Two types of problems are solved in this study: VRP and TSP instances in the routing literature.

The data file for a VRP instances includes coordinates of cities and demands assigned to cities. In our study we considered demands as profits as previously done by Fischetti et al. (1998) and Berube et al. (2008).

For TSP instances we generated profits, P_i, according to method proposed by Fischetti et al. (1998) and used by Berube et al. (2008). Three types of profits are considered for TSP instances:

Type 1 Profits: $P_i = 1$ (11)

Type 2 Profits: $P_i = 1 + (7141 * i + 73) mod(100)$ (12)

Type 3 Profits: $P_i = 1 + \lfloor 99 * c_{1i}/\theta \rfloor \; where \; \theta = max_{i \epsilon V \setminus \{1\}} c_{1i}$ (13)

In (11), (12), and (13) $i=1$ is considered as depot city and P_1 is set to 0. In (13), c_{1i} is the distance between depot city and ith city and V is the set of all cities. Thus, Type 1 profits are constant and same for all cities, Type 2 profits are generated uniformly between [0,100], and Type 3 profits are generated between [0,100] where high profits are assigned to cities far away from the depot city.

We give computational results for most of VRP and TSP instances considered in literature by other researchers. 61 problems with problem sizes up to 400 cities are solved using mNSGA-II.

4.2 Computational Results

Up to now, it has been shown that mNSGA-II performs better in terms of convergence and size of Pareto front generated. Hence, only mNSGA-II algorithm is used to solve a wide range of problem types and sizes. As previously stated, a crossover probability of 0.9, a mutation probability of 0.02, a population size of 100, a generation number limit of 1.000, 5.000, and 10.000 for problems including less than 100, between 100 and 200, and between 200 and 400 cities respectively are used. mNSGA-II is coded in C programming language on a Linux platform with an Intel Core2Duo 2 Ghz processor and 2 GB of RAM. Tables 9, 10, 11, and 12 show results for problems from VRP and TSP literature (TSPLIB, 2008). Table 9 shows results for VRP instances, Table 10 gives results for TSP instances with Type 1 profits, Table 11 gives results for TSP instances with Type 3 profits for which results exist in the literature, and finally Table 12 presents results regarding TSP instances with Type 3 profits for which complete approximate pareto fronts are given for the first time in this study.

We use results of studies conducted by Fischetti et al. (1998) and Berube et al. (2008). We refer to the former FGT and latter BGP. The followings are summarized in Tables 9, 10, 11, and 12;

Problem Name: The name of problem as in TSPLIB (2008) is given.

Size of Pareto Front: Number of efficient solutions reported in the literature and number of efficient solutions generated by mNSGA-II are given. In literature, BGP provides complete distinct nondominated solutions, whereas FGT reports only three solutions for VRP instances and only one solution for TSP instances.

Hyper Volume of Pareto Front: HV enclosed by Pareto fronts reported by BGP and FGT and HV enclosed by Pareto front found by mNSGA-II are presented. For problems where only one (see Tables 9 and 12) or three solutions is reported (problems reported by FGT), we did not compute hyper volume since three solutions are not adequate to correctly estimate Pareto-optimal front, hence a HV for Pareto-optimal front.

HVR: Hyper Volume Ratio for reported Pareto fronts is given. Pareto front reported by BGP is the Pareto-optimal front. On the other hand, since results provided by FGT includes only 3 solutions (1 solution in the case of TSP), we did not compute HV for solutions given by FGT and HVR for problems reported by FGT.

Percent GD: Average percent deviation in cost and profit objective is calculated for all problems.

NGP: Number of generations performed. It shows how many generations are performed for the given problem size.

CPU Tims: It shows total CPU time spent for each problem reported by BGP and used by mNSGA-II. FGT reports that they put time limit of 5 hours to generate a single solution. Also since the study reported by FGT is old, considering technological advancements in computer industry, CPU times for FGT are not included in tables. FGT have used a DEC station 5000/240 and on a Hewlett Packard Apollo 9000/720 computer and CPLEX 3.0. BGP have used an AMD Opteron 2.4 Ghz processor and CPLEX 9.3.

To summarize, Fischetti et al. (1998) provide 3 solutions for VRP instances and 1 solution for TSP instances. The largest VRP instance they solve is GIL260 and the largest TSP instance they solve is RD400. They use a time limit of 5 hours to generate a solution. However, since their work dates back to 1998, technology they have used to implement their algorithm is incomparable with current computer technology we have used. Thus, we will not present a comparison between our results and results of Fischetti et al. (1998). On the other hand, study of Berube et al. (2008) is a recent one. They generate the whole Pareto front for problems they were able to solve under a time limit of 72 hours. We compare our computation times with theirs since their study is recent. The largest VRP instance for which they report Pareto-optimal front is EIL101. The largest TSP instance with Type 1 profits they are able to solve includes 150 cities and the largest TSP instance with Type 3 profits for which they are able to generate the whole Pareto front in 72 hours is EIL101. However, we should note that they are unable to solve some medium size problems such as PR76 for Type 3 profits. They report that instances with Type 3 profits are the hardest problems. Based on their claim we have solved only TSP instances with Type 1 and Type 3 profits. Instances with Type 2 profits are not considered in this paper since we believe that instances with type 1 and type 3 profits are adequate to show performance of mNSGA-II.

Before discussing results regarding hyper volumes and hyper volume ratios given in Tables 9, 10, 11, and 12, we want to point out an issue regarding hyper volumes reported for the problems with Pareto front size of 1 and 3 solutions. For these problems, actually, HV and HVR are meaningless. All convexities and concavities of Pareto-optimal front would be disregarded if we calculated HV for these problem instances. Hence, it is more meaningful to look at Percent GD for such problems.

For problems with complete Pareto-optimal fronts, results reported by BGP, HVR values for VRP instances are above 0.99 as can be seen in Table 9. This means that the results very close to Pareto-optimal fronts are found by mNSGA-II. HVR values for TSP instances with Type 1 profits, in Table 10, are 0.96 for one instance, 0.98 for 5 instances and above 0.99 for other 14 problem instances. It can be concluded that mNSGA-II is able to find results very close to the Pareto-optimal front for other 6

Table 9 *Computational results for VRP instances.*

Problem Name†	**Size of Pareto Front**		**Hyper Volume of Pareto Front**		**HVR‡**	**Percent GD**		**NGP**	**CPU Time**	
	BGP/FGT	**mNSGA- II**	**BGP/FGT**	**mNSGA-II**		**Cost**	**Profit**		**BGP**	**mNSGA-II**
EIL22	67	67	3,298,300	3,297,500	0.9998	0.41	0.06	1,000	00:00:08	00:05:35
EIL23	75	77	3,404,732	3,404,732	1.0000	0.00	0.00	1,000	00:00:08	00:05:40
EIL30	125	132	2,690,675	2,689,974	0.9997	0.10	0.00	1,000	00:00:27	00:09:40
EIL33	159	161	6,565,900	6,564,433	0.9998	0.00	0.00	1,000	00:00:55	00:06:32
EIL51	223	238	197,640	197,565	0.9996	0.06	0.04	1,000	00:09:00	00:04:55
EIL76	355	337	442,888	441,599	0.9971	0.36	0.14	1,000	00:43:25	00:13:28
EIL101	498	644	584,767	583,624	0.9980	0.34	0.07	5,000	01:24:07	02:25:50
CMT101	3	794	-	487,490	-	0.45	0.00	5,000	*	05:36:27
CMT121	3	494	-	390,436	-	0.49	0.02	5,000	*	10:49:44
CMT151	3	603	-	955,292	-	0.80	0.06	5,000	*	04:06:49
CMT200	3	535	-	1,469,317	-	2.37	0.17	10,000	*	10:39:24
GIL262	3	668	-	16,837,105	-	3.25	0.09	10,000	*	20:37:02

* Complete Pareto front is not available. Complete Pareto front for the problem is published for the first time here.
† Format for problem names is “ABC#”, where “ABC” is a keyword for problem and “#” indicates the problem size *n*.
‡ Hyper volume ratio is the ratio between hyper volume enclosed by Pareto front found by mNSGA II and hyper volume enclosed by Pareto optimal front.

Table 10 *Computational results for TSP instances with Type 1 profits.*

Problem Name†	Size of Pareto Front		Hyper Volume of Pareto Front		HVR‡	Percent GD		NGP	CPU times	
	BGP/FGT	mNSGA-II	BGP/FGT	mNSGA-II		Cost	Profit		BGP	mNSGA-II
EIL51	51	91	11,428	11,425	0.9997	0.05	0.00	1,000	00:00:10	00:12:05
BERLIN52	52	52	246,976	246,707	0.9989	0.16	0.14	1,000	00:00:10	01:12:52
ST70	70	101	26,445	26,340	0.9960	0.31	0.30	1,000	00:01:02	00:23:06
EIL76	76	128	22,878	22,690	0.9918	0.45	1.66	1,000	00:00:37	00:30:02
PR76	76	78	4,665,108	4,627,952	0.9920	0.38	1.23	1,000	48:42:14	00:50:21
RAT99	99	122	60,802	60,081	0.9881	0.41	1.34	1,000	00:03:53	00:23:10
KROB100	100	101	1,213,670	1,199,239	0.9881	0.45	1.61	5,000	00:17:56	03:58:50
KROE100	100	100	1,220,850	1,215,885	0.9959	0.35	0.31	5,000	00:13:57	03:34:13
RD100	100	105	443,357	440,870	0.9944	0.30	0.78	5,000	00:06:05	02:57:56
EIL101	101	248	36,747	36,503	0.9934	0.38	0.98	5,000	00:01:30	03:49:36
LIN105	105	126	843,792	842,515	0.9985	0.14	0.04	5,000	01:32:39	05:58:14
PR107	107	6620	2,275,961	2,270,681	0.9977	0.08	0.25	5,000	00:01:14	08:03:13
PR124	124	137	3,978,325	3,930,137	0.9879	0.25	2.60	5,000	00:49:51	11:17:55
BIER127	127	132	10,617,461	10,579,952	0.9965	0.82	3.56	5,000	00:17:53	25:30:42
CH130	130	140	448,458	444,017	0.9901	0.44	1.51	5,000	00:11:59	07:33:39
PR136	136	562	6,876,821	6,860,664	0.9977	0.30	0.08	5,000	17:56:30	06:08:37
PR144	144	161	4,359,979	4,324,206	0.9918	0.42	1.16	5,000	04:51:21	10:27:54
CH150	150	168	536,330	527,830	0.9842	0.56	2.45	5,000	01:01:46	06:37:34
KROA150	150	152	2,155,970	2,076,616	0.9632	0.64	6.06	5,000	22:30:24	07:12:38
KROB150	150	153	2,122,032	2,081,561	0.9809	0.54	3.38	5,000	18:54:50	07:36:51

† Format for problem names is “ABC#”, where “ABC” is a keyword for problem and “#” indicates the problem size *n*.
‡ Hyper volume ratio is the ratio between hyper volume enclosed by Pareto front found by mNSGA II and hyper volume enclosed by Pareto optimal front.

Table 11 *Computational results for TSP instances with Type 3 profits for which results exist in the literature.*

Problem Name†	**Size of Pareto Front**		**Hyper Volume of Pareto Front**		**HVR‡**	**Percent GD**		**NGP**	**CPU Time**	
	BGP/FGT	**mNSGA-II**	**BGP/FGT**	**mNSGA-II**		**Cost**	**Profit**		**BGP**	**mNSGA-II**
EIL51	267	242	539,301	538,304	0.9982	0.37	0.26	1,000	00:19:56	00:07:04
BERLIN52	439	419	7,344,871	7,331,620	0.9982	0.09	0.38	1,000	00:21:32	00:51:20
ST70	452	454	1,302,221	1,276,653	0.9804	2.48	0.44	1,000	02:34:18	00:16:41
EIL76	383	350	1,165,318	1,152,535	0.9890	1.55	0.34	1,000	01:22:51	00:23:34
RD100	1513	1137	21,453,308	21,434,376	0.9991	0.23	0.49	5,000	50:16:00	03:24:25
EIL101	499	495	1,869,369	1,865,935	0.9982	0.26	0.12	5,000	10:37:07	02:35:17

† Format for problem names is “ABC#”, where “ABC” is a keyword for problem and “#” indicates the problem size *n*.
‡ Hyper volume ratio is the ratio between hyper volume enclosed by Pareto front found by mNSGA II and hyper volume enclosed by Pareto optimal front.

***Table 12** Results for TSP instances with Type 3 profit for which complete pareto fronts are given for the first time.*

Problem Name†	**Size of Pareto Front**		**Hyper Volume of Pareto Front**	**Percent GD**		**NGP**	**CPU Time**
	BGP/FGT	**mNSGA-II**		**Cost**	**Profit**		
PR76	1	560	227,374,231	0.06	0.08	1,000	00:45:52
RAT99	1	727	3,230,875	0.61	0.16	1,000	00:17:06
KROB100	1	862	53,183,779	0.05	0.54	5,000	03:29:30
KROE100	1	1124	56,941,724	0.04	0.00	5,000	02:46:37
LIN105	1	916	38,784,808	0.28	0.80	5,000	05:03:05
PR107	1	1162	115,318,278	0.41	0.89	5,000	05:48:02
PR124	1	822	204,782,671	0.18	0.95	5,000	04:39:23
BIER127	1	1104	252,861,992	0.02	0.18	5,000	10:19:27
CH130	1	903	22,785,953	0.00	0.00	5,000	05:03:51
PR136	1	1007	384,209,556	0.02	0.32	5,000	06:39:20
PR144	1	972	216,310,640	0.16	1.30	5,000	12:03:28
CH150	1	1080	28,269,844	0.00	0.00	5,000	06:31:31
KROA150	1	843	117,424,746	0.02	0.13	5,000	04:50:54
KROB150	1	942	121,774,833	0.14	0.80	5,000	06:04:51
PR152	1	1034	293,545,081	0.00	0.03	5,000	57:12:27
U159	1	846	191,262,327	0.02	0.15	5,000	04:48:02
RAT195	1	714	12,454,180	1.66	0.01	5,000	03:37:14
D198	1	682	83,462,482	0.48	0.74	5,000	25:34:12
KROA200	1	1012	159,004,259	0.02	1.18	10,000	22:33:44
TS225	1	751	817,585,504	0.07	1.86	10,000	07:58:30
PR264	1	827	336,548,607	0.70	2.97	10,000	33:42:34
LIN318	1	682	360,287,518	0.65	1.79	10,000	36:48:34
RD400	1	751	160,862,912	2.75	1.72	10,000	37:33:31

† Format for problem names is "ABC#", where "ABC" is a keyword for problem and "#" indicates the problem size *n*.

instances. In Table 11, we observe that HVR values are 0.98 for 2 out of 6 problems, for which complete Pareto-fronts are available, and 0.99 for the remaining 4 problem instances. For all other problem instances we will use Percent GD as comparison tool.

From Table 9 we observe that Percent GD in both objectives for the problems including less than 200 cities is smaller than 1%. It is 2% for a 200-city problem and 3% for a 262-city problem. Considering Table 10, Percent GD is less than 0.5% for most of the problems in the cost objective. However, we also observe values up to 6% in the profit objective. The reason for such a slightly larger deviation in profit objective will become clearer after investigating Table 9. However, we can conclude that results of mNSGA-II are very close to the Pareto-optimal front. In Tables 11 and 12, we observe that Percent GD in cost objective is below 1% except for two problems. For these two problems it is 1% and 2%. In profit objective we observe that PGD is below 1% up to problem sizes of 200 cities and below 2% up to problem sizes of 400 cities. Thus for these problems, mNSGA-II generate good approximations to Pareto-optimal front. If we reconsider 6% deviation in profit objective for TSP instances with Type 1 profits in Table 10 together with results presented in Tables 11 and 12, the deviation is most probably due to range of profit objective. Because that in Type 1 profits all cities, except depot, have a profit value of 1, even missing 1 or 2 points on Pareto-optimal front causes large deviations. This argument is also enhanced with results found for TSP instances with Type 3 profits, where range of profit objective is moderately larger.

We have seen that mNSGA-II performs well in finding near-optimal results for all problems considered. Now, we would like to compare our computation times with the ones reported by BGP. We observe from Tables 9 and 10 that for small problem sizes their algorithm is better than mNSGA-II in general. However, for some small problems and for large problem sizes their runtimes are much larger than runtimes of mNSGA-II. For instance, the CPU time reported by BGP for PR76 in Table 10 is 10 times of CPU time of mNSGA-II. The same situation is observed for some small problems in Tables 9 and 10. Another observation is that as problem size gets larger, CPU times of BGP increases more rapidly than that of mNSGA-II. Figure 10 shows pareto front for PR76 instance with Type 1 profits. One solution provided by FGT is used in figure since BGP report that they could not fount complete pareto front in 72 hours of time limit. We have also included extreme solutions although FGT do not report. As it can be seen from Figure 10, approximation of mNSGA-II is very close to optimal. Figures 11 and 12 show pareto fronts for two other problem instances: CH150 with Type 1 profits and TS225 with Type 3 profits. Again approximations of mNSGA-II are very good.

If we consider Tables 11 and 12, CPU times for mNSGA-II are much better than CPU times of BGP. Except for one small problem, BERLIN52, in all cases CPU times for mNSGA-II are much shorter than that of BGP. In fact, BGP reports that their algorithm is very dependent on profit structure of the problem instances and that they are unable to solve most of problem instances with type 3 profits in a time limit of 72 hours, for problems reported in Table 12. On the other hand, since mNSGA-II is a search heuristic, we observe from Tables 10, 11, and 12 that it generates similar results for all profit structures. Such a robustness property of mNSGA-II is one of the most important features of our proposed algorithm. We observe from Table 12 that RAT99 could not be solved in 72 hours by BGP, whereas, mNSGA-II finds a result very close to optimal pareto front in 17 minutes. Runtimes of mNSGA-II is below 38 hours for the largest problem solved, which indicates a logarithmic-like attitude in terms of runtime of the algorithm. Hence, we believe that mNSGA-II is a very powerful heuristic especially for large sized problems, where exact algorithms fail.

From Tables 9, 10, 11, and 12, we conclude that mNSGA-II is able to find Pareto fronts very close to Pareto-optimal fronts. We also observe a trend in results we have found. We observe that as problem size increases deviation from Pareto-optimal front also increases. The deviation seems to increase in a slow fashion when number of generations performed is increased as we previously stated. This result is expected

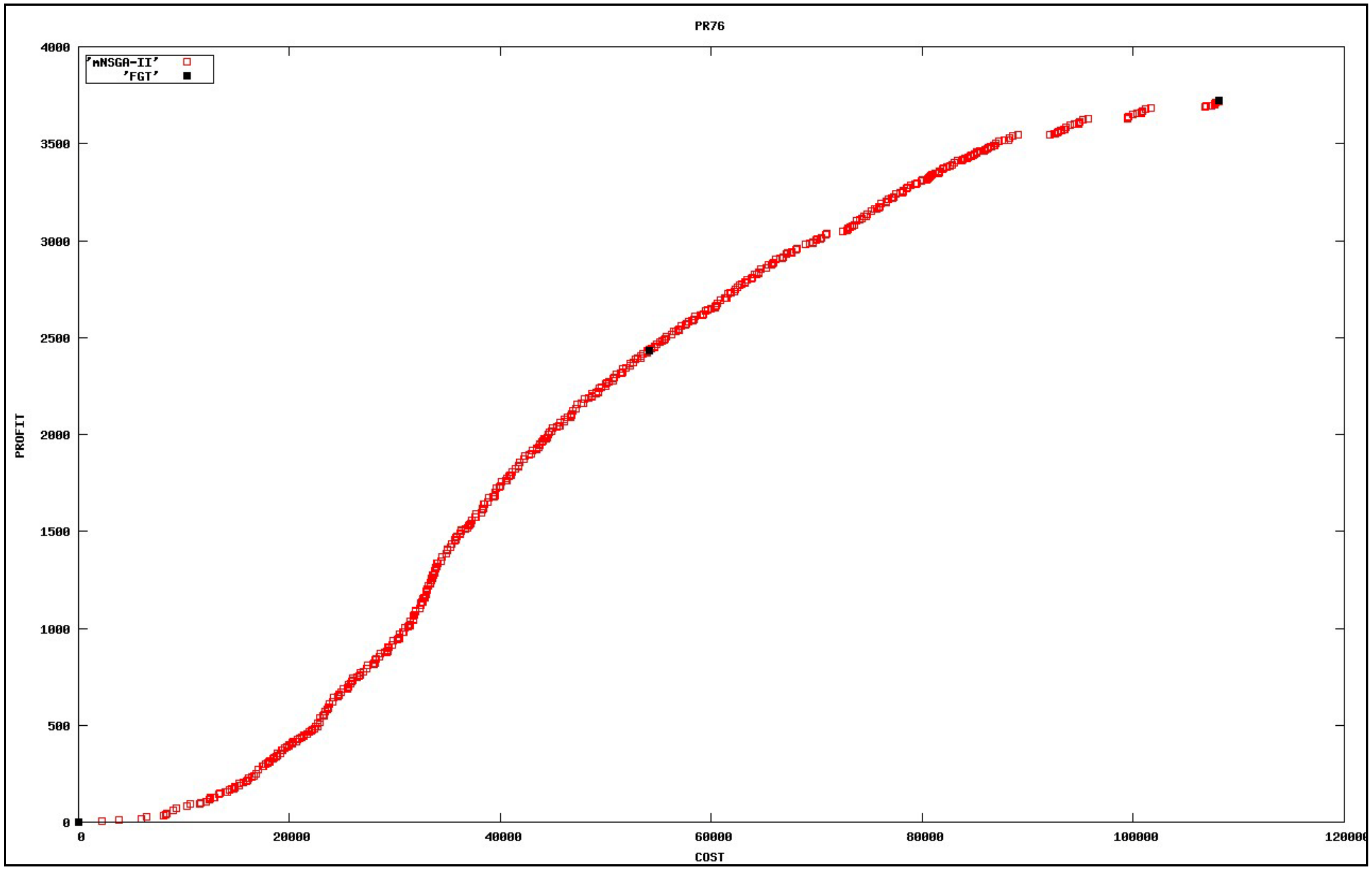


Figure 10 *Pareto front generated by mNSGA-II and solutions provided by FGT for PR76 with Type 3 profits.*

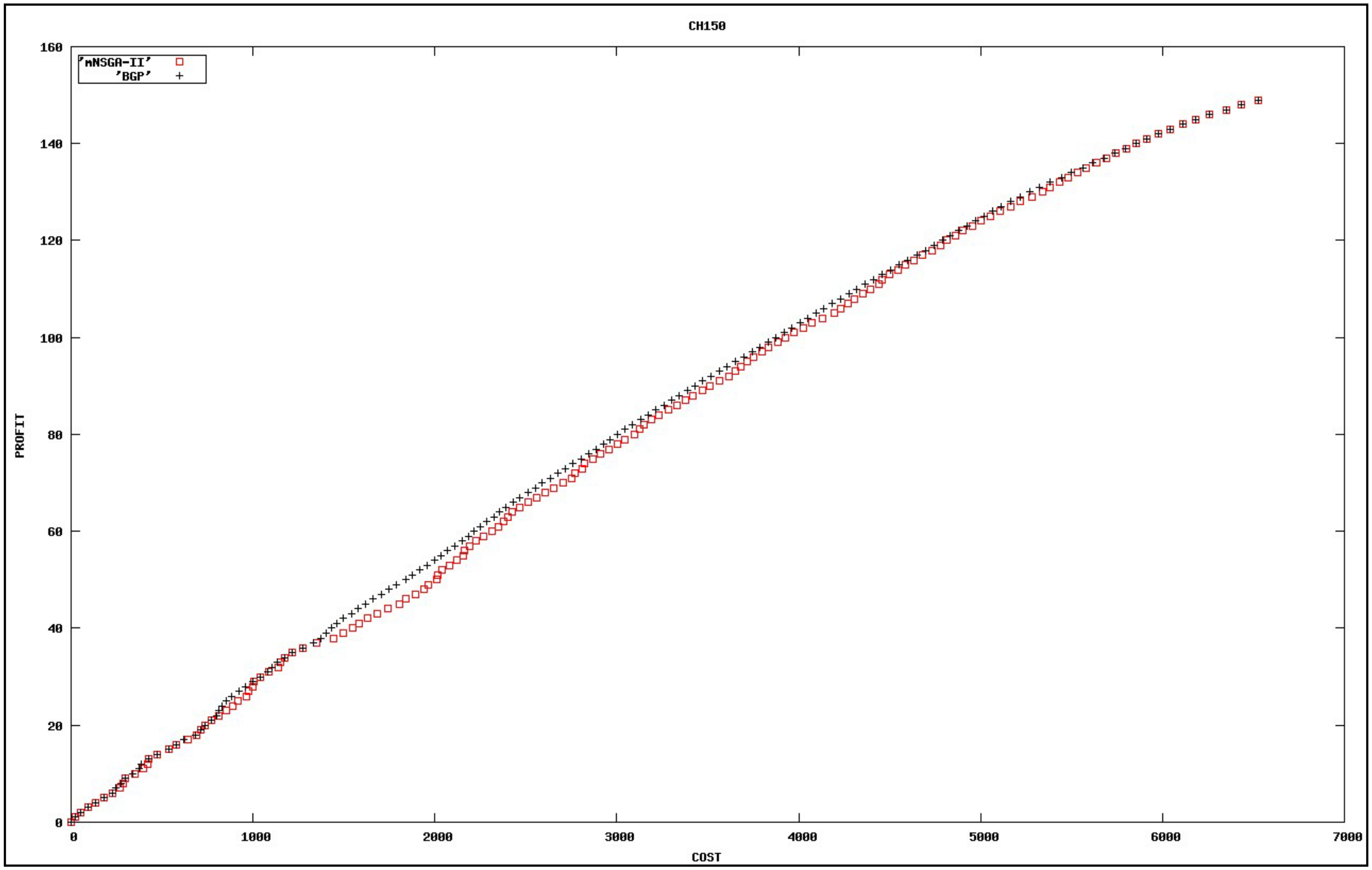


Figure 11 *Pareto front generated by mNSGA-II and solutions provided by BGP for CH150 with Type 1 profits.*

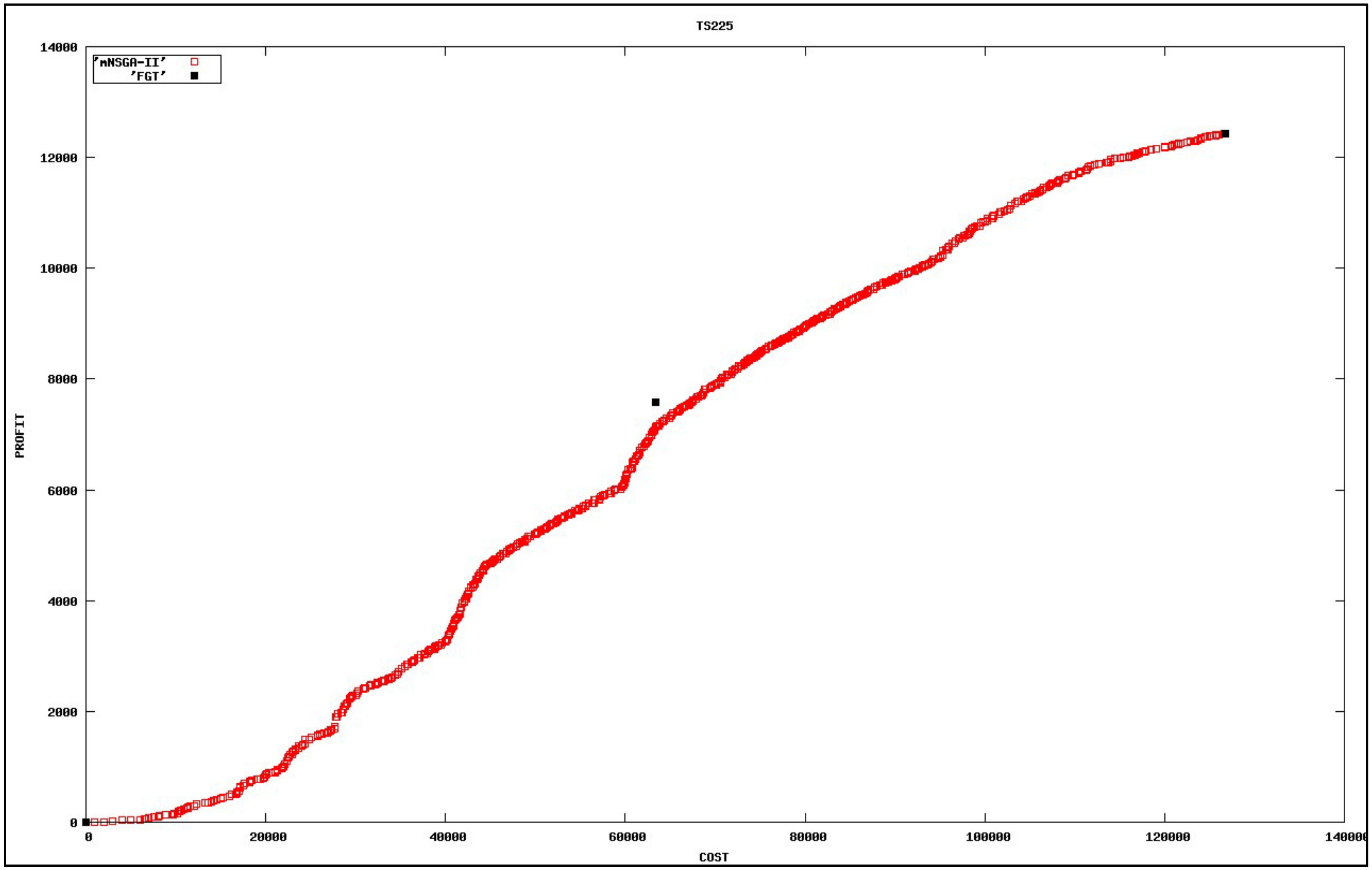


Figure 12 *Pareto front generated by mNSGA-II and solutions provided by BGP for TS225 with Type 3 profits.*

because as problem size increases in TSPP, the search space for mNSGA-II increases exponentially. Hence, either population size or number of generations performed should be increased in order to partially avoid this drawback.

CHAPTER 5

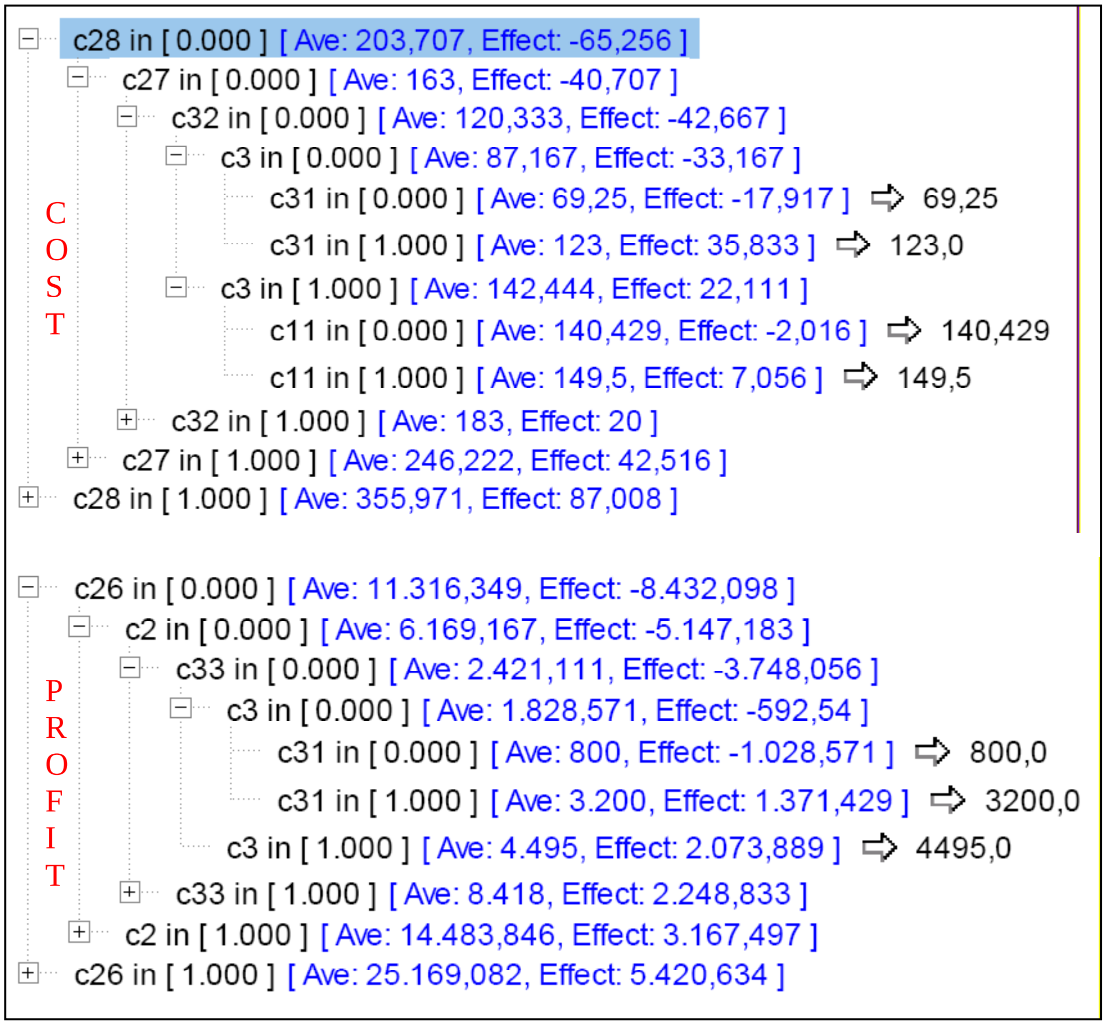
A FRAMEWORK FOR POST-OPTIMALITY ANALYSIS OF PARETO FRONT

After finding Pareto front for the problem, DM has to choose one of these nondominated solutions. The aim of this chapter is to support this decision making process and guide the DM by extracting supportive information from found Pareto front and corresponding decision space vectors (i.e., efficient solution vectors). In this chapter we demonstrate post-optimality analysis performed by a data mining software, SPSS Clementine, on the Pareto front found for the VRP instance EIL33 by mNSGA-II.

Which cities have the largest marginal effect on pareto front?

The first analysis we make is to find cities with largest marginal effect on objectives. We start with inputting the Pareto front to the software. It contains subset of cities visited, cost, and profit values for each efficient solution. Pareto front for EIL33 is given in Appendix C. After feeding data into program, we choose one of the objectives that analysis will be conducted for. The subset of cities for each solution (decision space vector) is examined and the cities with highest marginal effect on chosen objective are reported. Results for the test instance EIL33 are shown in Figure 13. For instance, considering cost objective, the average cost is 203 for solutions excluding city 28 while it is 355 units for solutions including city 28. Average cost is 163 units for solutions excluding both cities 28 and 27. The process is shown also for profit objective. For this example we limited the depth of tree to 5 levels, however, the software can provide a depth where average of the considered objective hits to its minimum.

In fact, entire process is similar to conducting a sensitivity analysis in the LP (Linear



```
c28 in [ 0.000 ] [ Ave: 203,707, Effect: -65,256 ]
    c27 in [ 0.000 ] [ Ave: 163, Effect: -40,707 ]
        c32 in [ 0.000 ] [ Ave: 120,333, Effect: -42,667 ]
            c3 in [ 0.000 ] [ Ave: 87,167, Effect: -33,167 ]
                c31 in [ 0.000 ] [ Ave: 69,25, Effect: -17,917 ]  ⇨  69,25
                c31 in [ 1.000 ] [ Ave: 123, Effect: 35,833 ]  ⇨  123,0
            c3 in [ 1.000 ] [ Ave: 142,444, Effect: 22,111 ]
                c11 in [ 0.000 ] [ Ave: 140,429, Effect: -2,016 ]  ⇨  140,429
                c11 in [ 1.000 ] [ Ave: 149,5, Effect: 7,056 ]  ⇨  149,5
        c32 in [ 1.000 ] [ Ave: 183, Effect: 20 ]
    c27 in [ 1.000 ] [ Ave: 246,222, Effect: 42,516 ]
c28 in [ 1.000 ] [ Ave: 355,971, Effect: 87,008 ]

c26 in [ 0.000 ] [ Ave: 11.316,349, Effect: -8.432,098 ]
    c2 in [ 0.000 ] [ Ave: 6.169,167, Effect: -5.147,183 ]
        c33 in [ 0.000 ] [ Ave: 2.421,111, Effect: -3.748,056 ]
            c3 in [ 0.000 ] [ Ave: 1.828,571, Effect: -592,54 ]
                c31 in [ 0.000 ] [ Ave: 800, Effect: -1.028,571 ]  ⇨  800,0
                c31 in [ 1.000 ] [ Ave: 3.200, Effect: 1.371,429 ]  ⇨  3200,0
            c3 in [ 1.000 ] [ Ave: 4.495, Effect: 2.073,889 ]  ⇨  4495,0
        c33 in [ 1.000 ] [ Ave: 8.418, Effect: 2.248,833 ]
    c2 in [ 1.000 ] [ Ave: 14.483,846, Effect: 3.167,497 ]
c26 in [ 1.000 ] [ Ave: 25.169,082, Effect: 5.420,634 ]
```

Figure 13 *Post-optimality analysis for VRP instance EIL33.*

Programming). Here, the DM can clearly observe cities with drastic marginal effects on the considered objective, and can focus on specific regions of the Pareto front generated within a few steps by including or excluding some of these cities.

How are cities distributed on pareto front?

We are also able to find the range and concentration in each objective for each city. For example, Figure 14 shows histograms cities 6 and 17. X-coordinate shows values for considered objective and y-coordinate shows frequency of occurrence of considered city for each level of considered objective.

It is clearly observed that city 6 is included in various parts of Pareto front and has a large range of objective values. This means that city 6 is included both in short and long tours on Pareto front. However, city 17 is included only in long tours where large costs are accepted for larger profits. We can conclude that city 17 is far away from depot city and the only reason for including this city in a tour is the desire for larger profits. It is also reasonable to point out that there is nothing more to be decided for city 6, because city 6 is already included most of the tours. However, this is not the case for city 17.

Figure 15 shows another point of view for discussion on distribution of a city over Pareto front. It shows range of pareto front and the number of solutions including cities 6 and 7 and number of solutions excluding cities 6 and 7. In histograms, cities are considered independently, that is we do not mean solutions where both cities are included. Blue (red) colored bars show the number of solutions where the predetermined city is excluded (included).

How is a pair of cities distributed on pareto front?

We are also able to investigate the Pareto front considering both cities together. Figure 16 shows frequency diagrams considering both city 6 and city 7. In Figure 16, blue colored bars of graphic on the top left corner show the number and dispersion of

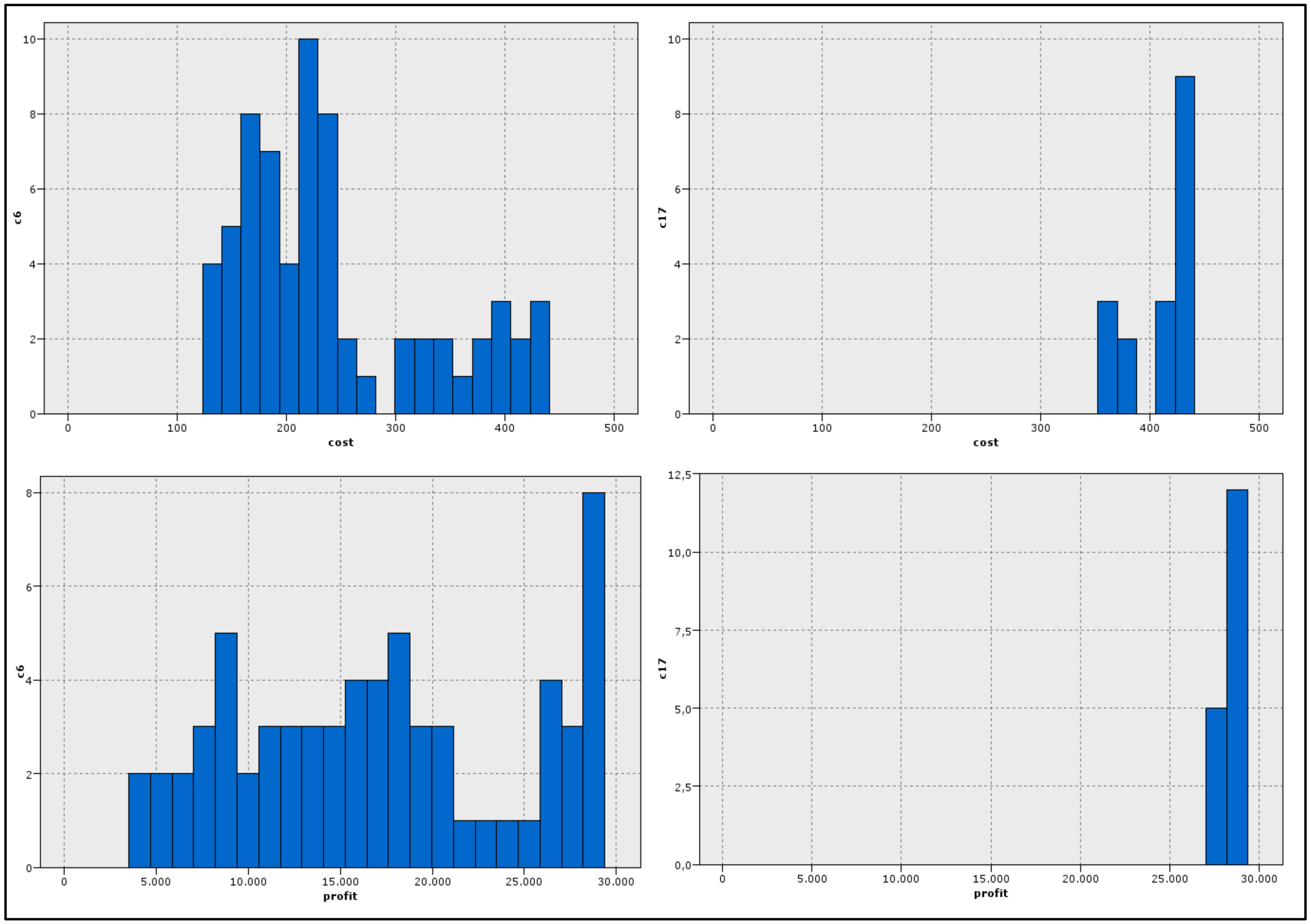


Figure 14 *Histograms for cities 6 and 17 showing distribution of objective function values for solutions they are included.*

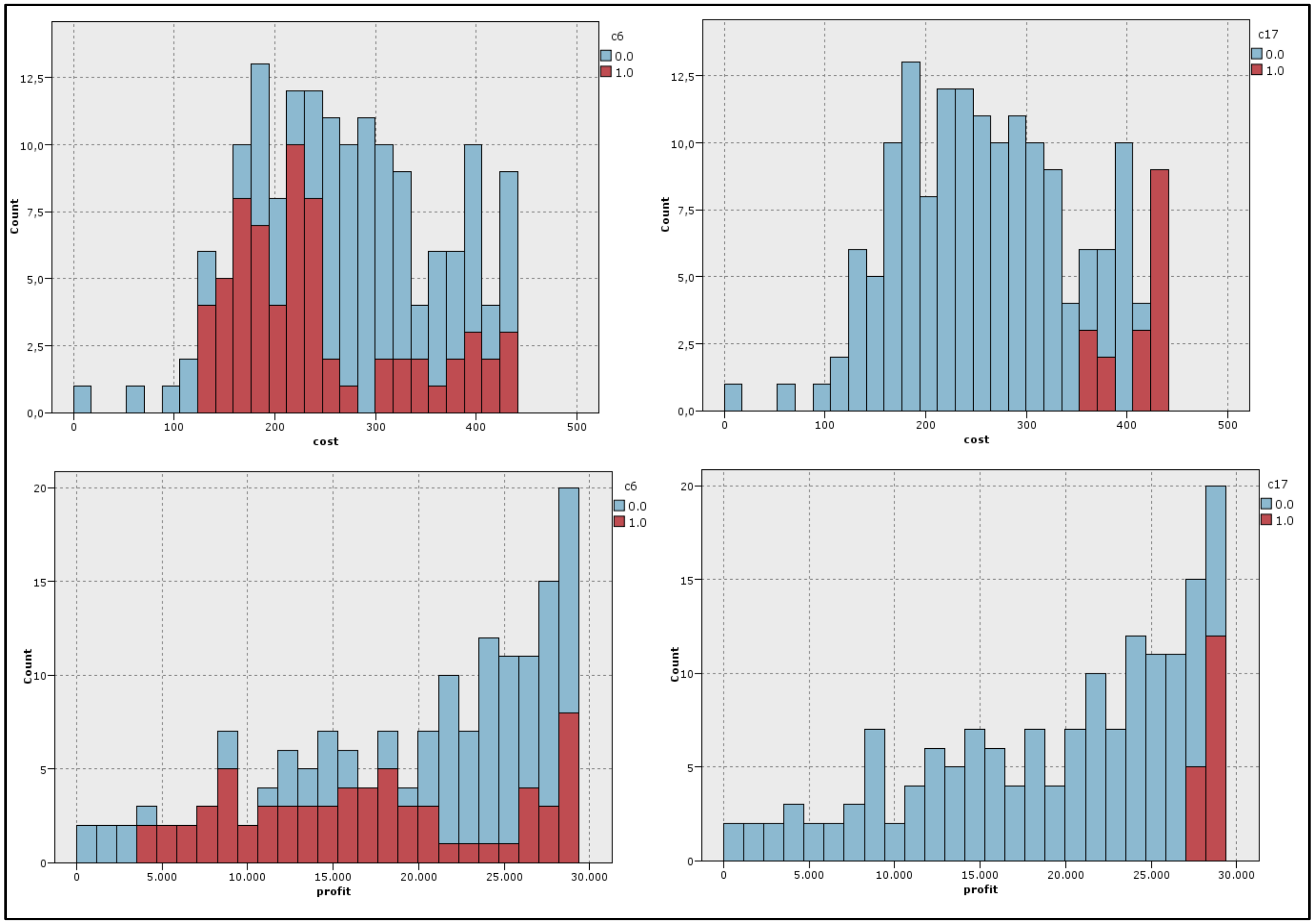


Figure 15 *Histograms for cities 6 and 17 showing objective function values where the two cities are included and excluded.*

solutions excluding both cities 6 and 17 in cost objective. Red colored bars of this graphic shows solutions excluding city 17 and including city 6. On the other hand, blue colored bars of graphic on the top right corner show solutions including city 17 and excluding 6. Red colored bars of this graphic shows distribution of solutions including both cities 6 and 17 in cost objective. Two graphics on the bottom of Figure 16 show similar results in profit objective.

How are cities correlated based on their usage on pareto front?

The last result we want to demonstrate is an overall graph showing correlation between cities considering Pareto front. To assist this aim, Clementine can generate a network of relationship intensity between a set of cities. As an example analysis, we consider cities 5, 6, 13, 17, and 31. Figure 17 shows co-occurrence frequency webfor considered 5 cities. Note that there is no limit on the number of cities that can be considered. In Figure 17, for each city considered there are two nodes on the web. The "0.000000" node represents the case for solutions excluding considered city and "1. 000000" node represents the case for solutions including considered city. The arc between two nodes on this net shows the co-occurrence of these two cases. Bold arcs represent the "strong" relationship between two cases, thin arcs represent "medium" relationship, and dashed arcs represents "weak" relationship between two cases. This classification is based on the percentage of solutions where considered two cases co-occur. Then, a DM can read from Figure 17 that the relationship type where both city 17 and city 6 are excluded from solutions on Pareto front is a strong one. Also one can conclude that relationship is always weak between the case that city 5 is included in the solutions and all other cases. That is, city 5 is most of the time excluded from tours. The same situation is valid for the case where city 17 is included in the solutions. Hence we can conclude that also city 17 is rarely included in tours. It is shown with a strong relationship on the co-occurrence frequency web for the links connecting the case where city 17 is excluded from solutions and all other cases.

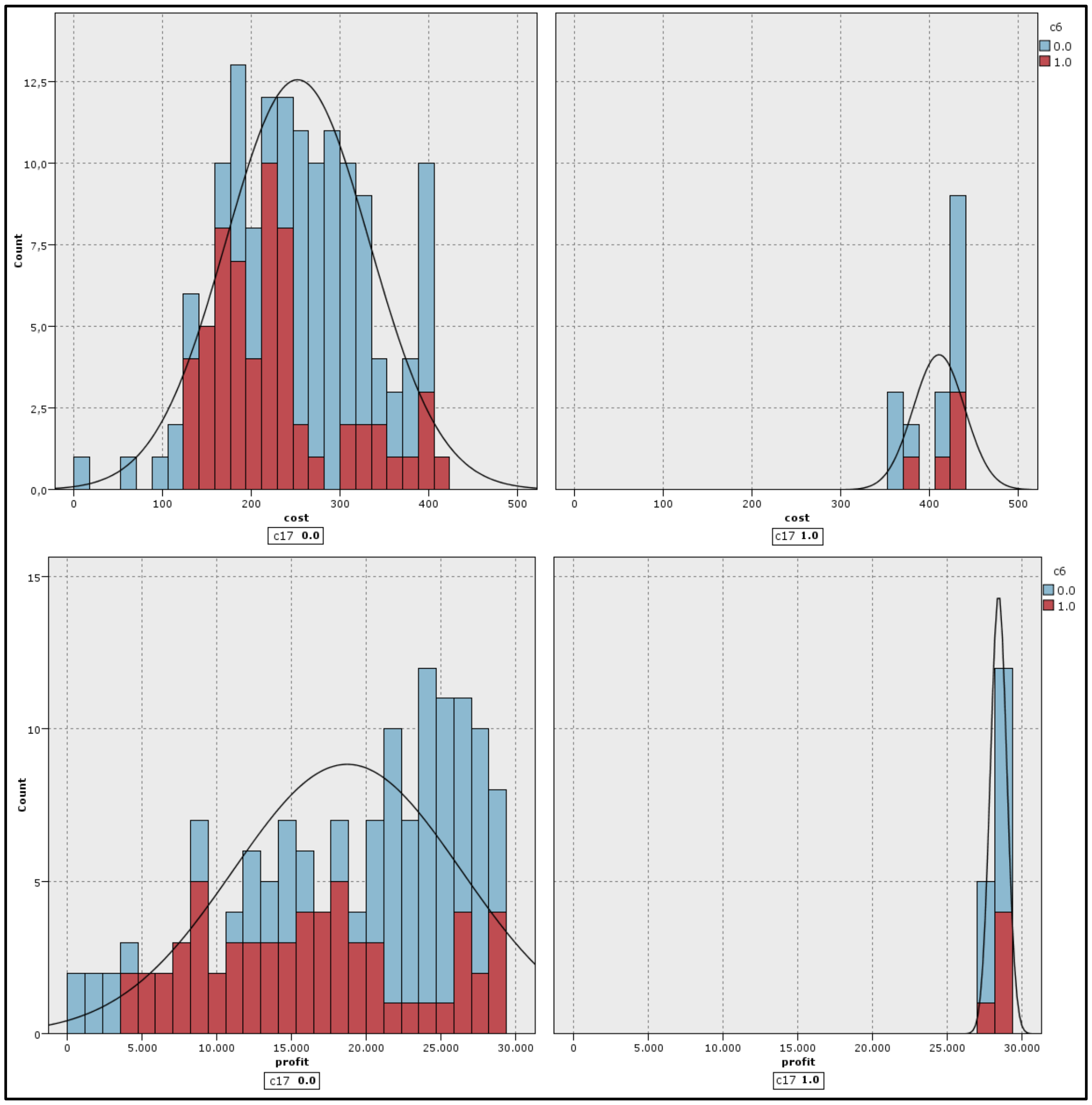


Figure 16 *Frequency histograms for cost and profit objectives when considering cities 6 and 17 together.*

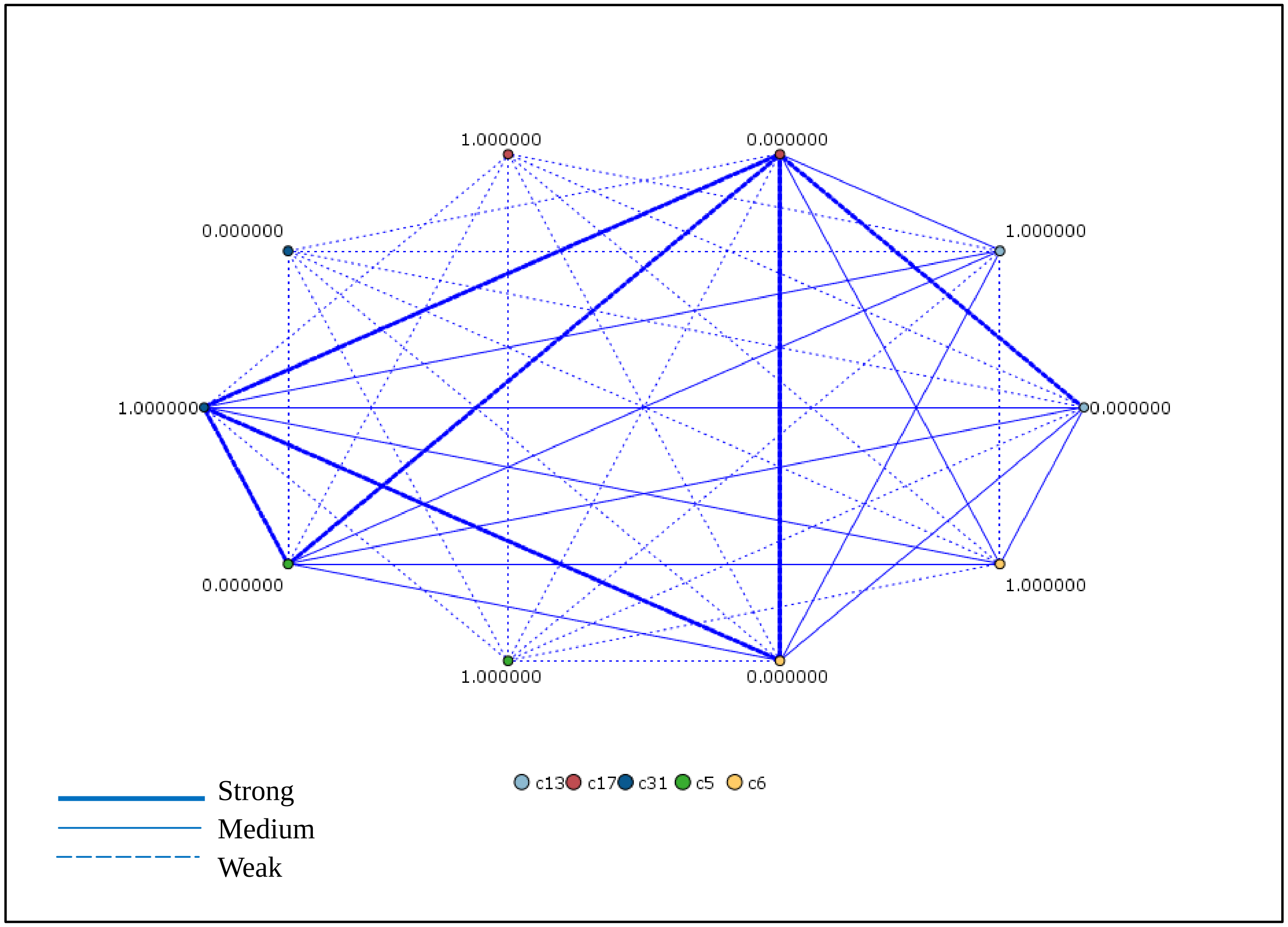
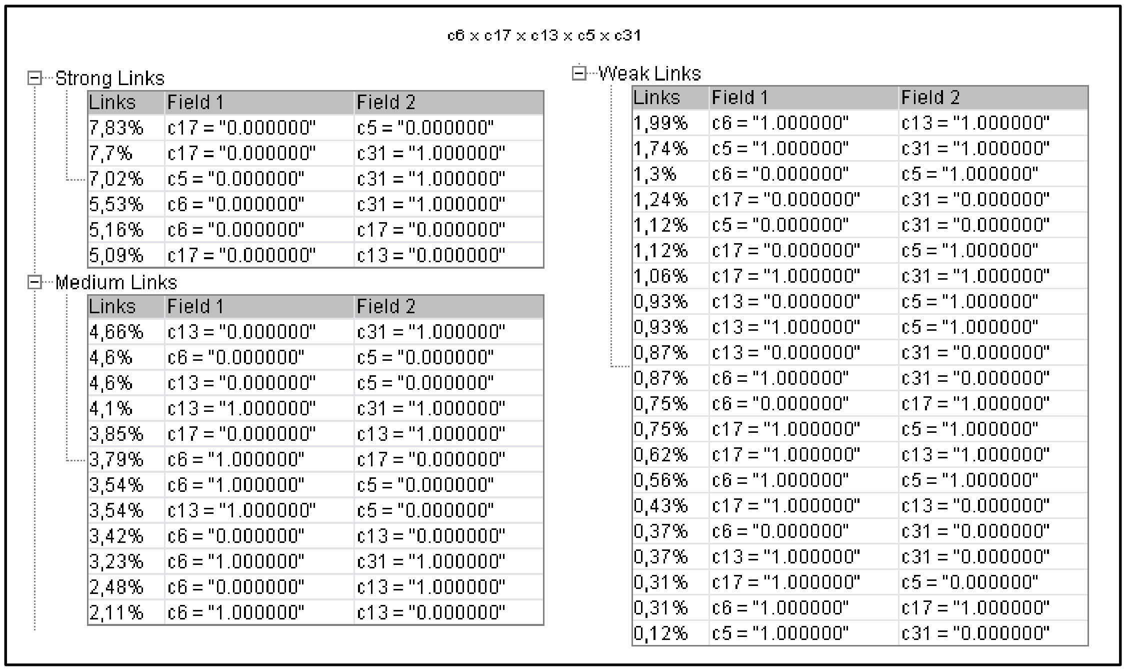


Figure 17 *Co-occurrence frequency web.*


c6 x c17 x c13 x c5 x c31

Strong Links

Links	Field 1	Field 2
7,83%	c17 = "0.000000"	c5 = "0.000000"
7,7%	c17 = "0.000000"	c31 = "1.000000"
7,02%	c5 = "0.000000"	c31 = "1.000000"
5,53%	c6 = "0.000000"	c31 = "1.000000"
5,16%	c6 = "0.000000"	c17 = "0.000000"
5,09%	c17 = "0.000000"	c13 = "0.000000"

Medium Links

Links	Field 1	Field 2
4,66%	c13 = "0.000000"	c31 = "1.000000"
4,6%	c6 = "0.000000"	c5 = "0.000000"
4,6%	c13 = "0.000000"	c5 = "0.000000"
4,1%	c13 = "1.000000"	c31 = "1.000000"
3,85%	c17 = "0.000000"	c13 = "1.000000"
3,79%	c6 = "1.000000"	c17 = "0.000000"
3,54%	c6 = "1.000000"	c5 = "0.000000"
3,54%	c13 = "1.000000"	c5 = "0.000000"
3,42%	c6 = "0.000000"	c13 = "0.000000"
3,23%	c6 = "1.000000"	c31 = "1.000000"
2,48%	c6 = "0.000000"	c13 = "1.000000"
2,11%	c6 = "1.000000"	c13 = "0.000000"

Weak Links

Links	Field 1	Field 2
1,99%	c6 = "1.000000"	c13 = "1.000000"
1,74%	c5 = "1.000000"	c31 = "1.000000"
1,3%	c6 = "0.000000"	c5 = "1.000000"
1,24%	c17 = "0.000000"	c31 = "0.000000"
1,12%	c5 = "0.000000"	c31 = "0.000000"
1,12%	c17 = "0.000000"	c5 = "1.000000"
1,06%	c17 = "1.000000"	c31 = "1.000000"
0,93%	c13 = "0.000000"	c5 = "1.000000"
0,93%	c13 = "1.000000"	c5 = "1.000000"
0,87%	c13 = "0.000000"	c31 = "0.000000"
0,87%	c6 = "1.000000"	c31 = "0.000000"
0,75%	c6 = "0.000000"	c17 = "1.000000"
0,75%	c17 = "1.000000"	c5 = "1.000000"
0,62%	c17 = "1.000000"	c13 = "1.000000"
0,56%	c6 = "1.000000"	c5 = "1.000000"
0,43%	c17 = "1.000000"	c13 = "0.000000"
0,37%	c6 = "0.000000"	c31 = "0.000000"
0,37%	c13 = "1.000000"	c31 = "0.000000"
0,31%	c17 = "1.000000"	c5 = "0.000000"
0,31%	c6 = "1.000000"	c17 = "1.000000"
0,12%	c5 = "1.000000"	c31 = "0.000000"

Figure 18 *Overall percentage of co-occurence frequencies for pair of cities.*

Moreover, this situation can be clearly observed in Figures 14 and 15 where we provide histograms for frequency of solutions including city 17. Figure 18 shows percentages of co-occurrences for both links that are shown and not shown in Figure 17. In Figure 18 we observe that 7.83% of solutions on Pareto front exclude both cities 17 and 5. Percentage of nondominated solutions that exclude city 17 and include city 31 is 7.7. On the other hand, the weakest link is the case where city 5 is included and city 31 is excluded from the tours constructed with a frequency of 0.12%. A DM can use these results for making robust decisions. For instance, strong links point out cities that are included or excluded in most of tours, hence these cities could be removed from or included into tours only in major distortions of considered scenarios. The Clementine models used to generate results in this chapter are summarized in Appendix D.

We believe that data mining can be a powerful tool for multiobjective decision making where preferences of a decision maker or decision makers are imprecise, conflicting, and very difficult to obtain and incorporate into developed algorithms. Post-optimality analysis that can be performed on the set of nondominated solutions by the help of a data mining software like SPSS Clementine can demonstrate the links from decision space into objective space and provide a better understanding of the problem under consideration. With a better understanding of the problem and guidelines provided, a DM could easily select his/her most preferred solution from the set of nondominated solutions generated with mNSGA-II.

CHAPTER 6

CONCLUSION

TSPP is a well-known hard multiobjective optimization problem. Hence, methods that can tackle the multiobjective nature of the problem effectively are compulsory. However, most of the algorithms proposed in literature reduce the problem to a single objective problem through scalarization of objectives. Few studies deal with TSPP using a multiobjective approach, however, these studies lack to effectively deal with problem.

In this study we review a well-known Multiobjective Genetic Algorithm, NSGA-II, for TSPP and propose several improvements on this algorithm, which we call mNSGA-II after the implementation of proposed improvements. We use Lin-Kernighan Heuristic provided with CONCORDE TSP solver to solve TSP instances generated during execution of mNSGA-II. We generate approximate Pareto fronts for 61 test problems in the TSP and VRP literature. The problem size for test problems varies between 22 and 400 cities.

Comparison with existing results in the literature proves that the pareto fronts generated by mNSGA-II algorithm are very close to Pareto-optimal fronts for problems including less than 150 cities and shows at most 2% deviation from Pareto-optimal front for larger problem sizes, except for a few test problems. We observe that computation time for our algorithm does not increase as rapid as computation times of other algorithms provided in the literature (e.g., algorithm of Fischetti et al., 1998, and algorithm of Berube et al., 2008). mNSGA-II is able to solve our largest test problem instance, the problem with 400 cities, in 38 hours, whereas the best known algorithm in literature for TSPP is unable to solve problems including more than 150 cities in a time limit of 72 hours.

Another strength of mNSGA-II is that its computational performance is independent from profit structure of test problems, based on results we have obtained for test problems we have considered. Berube et al. (2008) report that their algorithm is unable to solve problems with more than 100 cities in a time limit of 72 hours for a specific type of profit. For instance, for this specific profit structure our algorithm solves PR76 test problem which includes 76 cities in 50 minutes, whereas Berube et al. (2008) report 48 hours of CPU time. We solve the RD400 TSP instance which includes 400 cities in 38 hours using the same profit structure.

We also provide a data mining based approach for post-optimality analysis of generated pareto fronts. By conducting an analysis on decision space properties of a generated pareto front, DM is enabled to observe which cities are included in which parts of pareto front and the relationship among a given set of cities. We believe that this information is helpful to a DM for making a selection among the generated pareto front.

In conclusion, proposed algorithm, mNSGA-II, is shown to be a robust and effective heuristic for TSPP. Furthermore, a post-optimality analysis framework is described in this study. As future research directions, one could try to observe performance of proposed heuristic for larger problem sizes and study the proposed framework for post-optimality in a more detailed extent.

REFERENCES

1 Alves M. J. and Almeida M., 2007. MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem. *Computers & OR* 34, 3458-3470.

2 Awerbuch B., Azar Y., Blum A., and Vempala S., 1998. New Approximation Guarantees For Minimum-Weight K-Trees And Prize-Collecting Salesmen. *SIAM Journal on Computing* 28(1) 254-262.

3 Balas E., 1989. The prize collecting traveling salesman problem. *Networks* 19(6) 621–636.

4 Balas E., 1995. The prize collecting traveling salesman problem II: Polyhedral results. *Networks* 25(4) 199–216.

5 Balas E., 1999. New Classes of Efficiently Solvable Generalized Traveling Salesman Problems. *Annals of Operations Research* 86 529–558.

6 Berube J.F., Gendreau M., Potvin J.Y., 2008. An exact Epsilon-constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European Journal of Operational Research* (article in press).

7 Beume N., Naujoks B., and Emmerich M., 2007. SMS-EMOA: Multiobjective Selection based on dominated hypervolume. *European Journal of Operational Research* 181 1653-1669.

8 Chao I.M., Golden B. L., Wasil E. A., 1996. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research* 88(3) 475–489.

9 Chinchuluun A. and Pardalos P. M., 2007. A survey of recent developments in multiobjective optimization. *Annals of Operations Research* 154 29–50.

10 Coello Coello A., 2006. 20 Years of Evolutionary Multi-Objective Optimization: What Has Been Done and What Remains To Be Done. CINVESTAV-IPN

Evolutionary Computation Group, Departamento de Ingenieria Electrica, Seccion de Computacion Av. Instituto Polit´ecnico Nacional, MEXICO.

11 Georgia Institute of Technology, Concorde TSP Solver, 2008. http://www.tsp.gatech.edu/concorde.html, Last Accessed Date: 01 July 2008.

12 Deb K., Agrawal S., Pratap S., and Meyarivan T., 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. Kanpur Genetic Algorithms Laboratory (KanGAL), Indian Institute of Technology, Kanpur, INDIA.

13 Deb K., 2001. Multiobjective Optimization using Evolutionary Algorithms. *John Wiley & Sons LTD*.

14 Deb K., Mohan M., and Mishra S., 2005. Evaluating the ε–domination Based Multi-objective Evolutionary Algorithm for a Quick Computation of Pareto Optimal Solutions. *Evolutionary Computation* 13(4) 501-525.

15 Dell'Amico M., Maffioli F., Varbrand P., 1995. On Prize Collecting Tours and the Asymmetric TSP. *International Transactions of Operational Research* 2(3) 297-308.

16 Dell'Amico M., Maffioli F., Sciomachen A., 1998. A Lagrangian Heuristic for the Prize Collecting Traveling Salesman Problem. *Annals of Operations Research* 81 289-305.

17 Fischetti M., González J.J.S., Toth P., 1998. Solving the Orienteering Problem through Branch-and-Cut. *INFORMS Journal on Computing* 10 (2) 133-148.

18 Gendreau M., Laporte G., and Semet F., 1998a. A Branch-and-Cut Algorithm for the Undirected Selective Traveling Salesman Problem. *Networks* 32 263–273.

19 Gendreau M., Laporte G., and Semet F., 1998b. A Tabu Search Heuristic for the Undirected Selective Traveling Salesman Problem. *European Journal of Operational Research* 106 539–545.

20 Keller C.P., Goodchild M.F., 1998. The Multiobjective Vending Problem: A Generalization of the Travelling Salesman Problem. *Environmental Planning B: Planning Design* 15 447-460.

21 Laporte G. and Martello S., 1990. The Selective Traveling Salesman Problem.

Discrete Applied Mathematics 26 193–207.

22 Liang Y. and Smith A. E., 2001. An ant colony approach to the orienteering problem. Technical report. Department of Industrial and Systems Engineering, Auburn University, Auburn, AL, USA.

23 Millar H. H., Kiragu M., 1997. A Time-Based Formulation and Upper Bounding Scheme for the SelectiveTravelling Salesperson Problem. *The Journal of the Operational Research Society* 48(5) 511-518.

24 Rachmawati L. and Srinivasan D., 2006. Preference Incorporation in Multi-objective Evolutionary Algorithms: A Survey. *Proceedings of 2006 IEEE Congress on Evolutionary Computation,* Vancouver, CANADA.

25 Ramesh R., Yoon Y. S., and Karwan M. H., 1992. An Optimal Algorithm for the Orienteering Tour Problem. *ORSA Journal on Computing* 4(2) 155-165.

26 Soylu B. and Köksalan M., 2007. A Favorable Weight Based Evolutionary Algorithm for Multiple Criteria Problems. Technical Report. Department of Industrial Engineering, Middle East Technical University, Ankara, TURKEY.

27 Şimşek Ö., 2007. The biobjective Traveling Salesman Problem with Profit, Unpublished M.Sc. Thesis. Department of Industrial Engineering, Middle East Technical University, Ankara, Turkey.

28 Tasgetiren M. F. and Smith A. E., 2000, A Genetic Algorithm for the Orienteering Problem. *Proceedings of 2000 IEEE Congress on Evolutionary Computation,* San Diego, CA, USA.

29 Tsiligirides T., 1984. Heuristic Methods Applied to Orienteering. *The Journal of the Operational Research Society* 35(9) 797- 809.

30 Ruprecht Karls Universitat Heidelberg, TSPLIB, 2008. http://www.iwr.uni-heidelberg.de/groups/comopt/software/ TSPLIB95, Last Accessed Date: 01 July 2008.

31 Zitzler E., Laumanns M., and Thiele L., 2001. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. Technical Report. Evolutionary Methods for Design, Optimization and Control, Barcelona, SPAIN.

APPENDIX A

THE PSEUDO CODE FOR NSGA-II

```
Define:
    - P(i) : parent population in ith generation (of size popsize).
    - C(i) : child population in ith generation (of size popsize).
    - M(i) : mixed population in ith generation (of size 2*popsize).

Initialize
    - Randomly genarate P(1) (always include depot city and extreme solutions)
    - Evaluate P(1)
        o Assign cost using CONCORDE
        o Assign profit
    - Assign crowding distance and rank
        o Assign rank
        o Assign crowding distance
            ▪ Assign infinity to extremes in the population
            ▪ For solution j (where j is not an extreme solution)
                • Crow.dist(j) =
                  $\frac{\sum_{obj\ (i)} \frac{difference\ of\ two\ neighboring\ solutions\ in\ objective\ i}{difference\ of\ the\ extreme\ solutions\ in\ objective\ i}}{number\ of\ objectives}$
    - Return P(1) as P(2)

For ( i=2 ; i<=generation number limit ; i++)
    - Selection
        o Randomize P(i) and copy to POOL_1
        o Randomize P(i) and copy to POOL_2
        o For ( j=0 ; j<popsize ; j=j+4)
            ▪ Choose indivdual(j) and indivdual(j+1) from POOL_1
                • Tournament select parent 1 from indivdual(j) and
                  indivdual(j+1)
                    o If indivdual(j) dominates indivdual(j+1)
                        ▪ Return indivdual(j)
                    o If indivdual(j+1) dominates indivdual(j)
                        ▪ Return individual(j+1)
                    o If they are nondominated
                        ▪ If crowding_distance(j) >
                          crowding_distance(j+1)
                            • Return individual(j)
                        ▪ If crowding_distance(j) <
                          crowding_distance(j+1)
                            • Return individual(j+1)
                        ▪ If crowding_distance(j) =
                          crowding_distance(j+1)
                            • Randomly select one of the
                              individuals
            ▪ Choose indivdual(j+2) and indivdual(j+3) from POOL_1
                • Tournament select parent 2 from indivdual(j+2) and
                  indivdual(j+3)
            ▪ Choose indivdual(j) and indivdual(j+1) from POOL_2
                • Tournament select parent 3 from indivdual(j) and
                  indivdual(j+1)
            ▪ Choose indivdual(j+2) and indivdual(j+3) from POOL_2
                • Tournament select parent 4 from indivdual(j+2) and
                  indivdual(j+3)
            ▪ Perform crossover
                • For parent 1 and parent 2
                    o With crossover_probability cross two
                      parents
                        ▪ Perform two-point binary crossover
                            • Return two childs
                    o Otherwise return two parents as child
                • For parent 3 and parent 4
                    o With crossover_probability cross two
                      parents
                    o Otherwise return two parents as child
        o Return child population C(i)
    -   Mutation (bitwise mutation)
        o For  all childeren
            ▪ For all genes
                • Mutate gene with probability mutation_probability
                    o If gene=1, then gene=0
                    o Else gene=1
```

    - Return mutated C(i)
- **Evaluate child population C(i)**
- **Merge P(i) and C(i) into mixed population M(i)**
    - Create a mixed population of size 2*popsize
        - Copy P(i) and C(i) into M(i)
        - Return M(i)
- **Fill Nondominated sort**
    - Divide M(i) into fronts
        - Assign rank 1 to the first front
        - Assign rank 2 to the second front, and so on.
    - Create a temporary population TEMP of size popsize
        - If size_of_first_front <= popsize
            - Copy individuals in first front into TEMP
            - If size_of_second_front <= (popsize - first_front)
                - Copy individuals in second front into TEMP
                - If size_of_third_front <= (popsize – (first+second fronts))
                    - ...
            - Assign crowding distance to TEMP
        - If size_of_first_front > popsize
            - Assign crowding distance to the first front
                - Fill TEMP in non increasing order of crowding distance
    - Return TEMP as P(i+1)

APPENDIX B

THE PSEUDO CODE FOR mNSGA-II

```
Define:
    - P(i,j) : jth parent population in ith generation (of size popsize).
    - C(i,j) : jth child population in ith generation (of size popsize).
    - M(i,j) : jth mixed population in ith generation (of size 2*popsize).
    - E1(i) :  First external population in generation i, Elite Pop 1, (of size
      infinity).
    - E2(i) :  Second external population in generation i, Elite Pop 2, (of size
      infinity).

Initialize
    - Randomly genarate P(1,1)
    - Evaluate P(1,1)
        o Assign cost using CONCORDE
        o Assign profit
    - Assign crowding distance and rank
        o Assign rank
        o Assign crowding distance
            ▪ Assign infinity to extremes in the population
            ▪ For solution j (where j is not an extreme solution)
                • Crow.dist(j) =
                  $\frac{\sum_{obj\ (i)} \frac{difference\ \ of\ two\ neighboring\ \ solutions\ \ in\ objective\ \ i}{difference\ \ of\ the\ extreme\ \ solutions\ \ in\ objective\ \ i}}{number\ \ of\ objectives}$
    - E1(1)=NULL
    - P(2,1)=P(1,1)

For ( i=2 ; i<=generation number limit ; i++)
    - SELECTION
        o Randomize P(i) and copy to POOL_1
        o Randomize P(i) and copy to POOL_2
        o For ( j=0 ; j<popsize ; j=j+4)
            ▪ Choose indivdual(j) and indivdual(j+1) from POOL_1
            • Tournament select parent 1 from indivdual(j) and
              indivdual(j+1)
                o If indivdual(j) dominates indivdual(j+1)
                    ▪ Return indivdual(j)
                o If indivdual(j+1) dominates indivdual(j)
                    ▪ Return individual(j+1)
                o If they are nondominated
                    ▪ If crowding_distance(j) >
                      crowding_distance(j+1)
                        • Return individual(j)
                    ▪ If crowding_distance(j) <
                      crowding_distance(j+1)
                        • Return individual(j+1)
                    ▪ If crowding_distance(j) =
                      crowding_distance(j+1)
                        • Randomly select one of the
                          individuals
        ▪ Choose indivdual(j+2) and indivdual(j+3) from POOL_1
            • Tournament select parent 2 from indivdual(j+2) and
              indivdual(j+3)
        ▪ Choose indivdual(j) and indivdual(j+1) from POOL_2
            • Tournament select parent 3 from indivdual(j) and
              indivdual(j+1)
        ▪ Choose indivdual(j+2) and indivdual(j+3) from POOL_2
            • Tournament select parent 4 from indivdual(j+2) and
              indivdual(j+3)
        ▪ Perform crossover
            • For parent 1 and parent 2
                o With crossover_probability cross two
                  parents
                    ▪ Perform two-point binary crossover
                        • Return two childs
                o Otherwise return two parents as child
            • For parent 3 and parent 4
                o With crossover_probability cross two
                  parents
                o Otherwise return two parents as child
    o Return child population C(i)
```

```
- Mutation (bitwise mutation)
    o For  all childeren
        ▪ For all genes
            • Mutate gene with probability mutation_probability
                o If gene=1, then gene=0
                o Else gene=1
    o Return mutated C(i)
- Evaluate child population C(i,1)
- Merge P(i,1) and  C(i,1) into mixed population M(i,1)
    o Create a mixed population of size 2*popsize
        ▪ Copy P(i,1) and C(i,1) into M(i,1)
        ▪ Return M(i,1)
- Fill Nondominated sort
    o Divide M(i,1) into fronts
        ▪ Assign rank 1 to the first front
        ▪ Assign rank 2 to the second front, and so on.
    o Create a temporary population TEMP of size popsize
        ▪ If size_of_first_front <= popsize
            • Copy individuals in first front into TEMP
            • If size_of_second_front <= (popsize - first_front)
                o Copy individuals in second front into TEMP
                o If size_of_third_front <= (popsize –
                  (first+second fronts))
                    ▪ ...
            • Assign crowding distance to TEMP

        ▪ If size_of_first_front > popsize
            • Assign crowding distance to the first front
                o Fill TEMP in non increasing order of
                  crowding distance
    o Return TEMP as P(i,2)
    o Copy all rank 1 individuals in M(i,1) but not in P(i,2) into E2(i)
    o Return  P(i,2)
- Use waste individuals
    o While E1(i-1) is not empty; Copy jth  popsize individuals in E1(i-1)
      into C(i,j+1)
        ▪ Merge P(i,j+1) and  C(i,j+1) into mixed population
          M(i,j+1)
        ▪ Fill Nondominated sort
```

```
                • Fill nondominated sort P(i,j+2)
                • Append all rank 1 individuals in M(i,j+1) but not in
                  P(i,j+2) into E2(i)
            ▪ Set j=j+1;


- **Finalize**
    o Copy E2(i) into E1(i)
    o Return E1(i)
    o Return  P(i,j+1) as P(i+1,1)
```

APPENDIX C

PARETO FRONT GENERATED BY mNSGA-II FOR EIL33 VRP INSTANCE

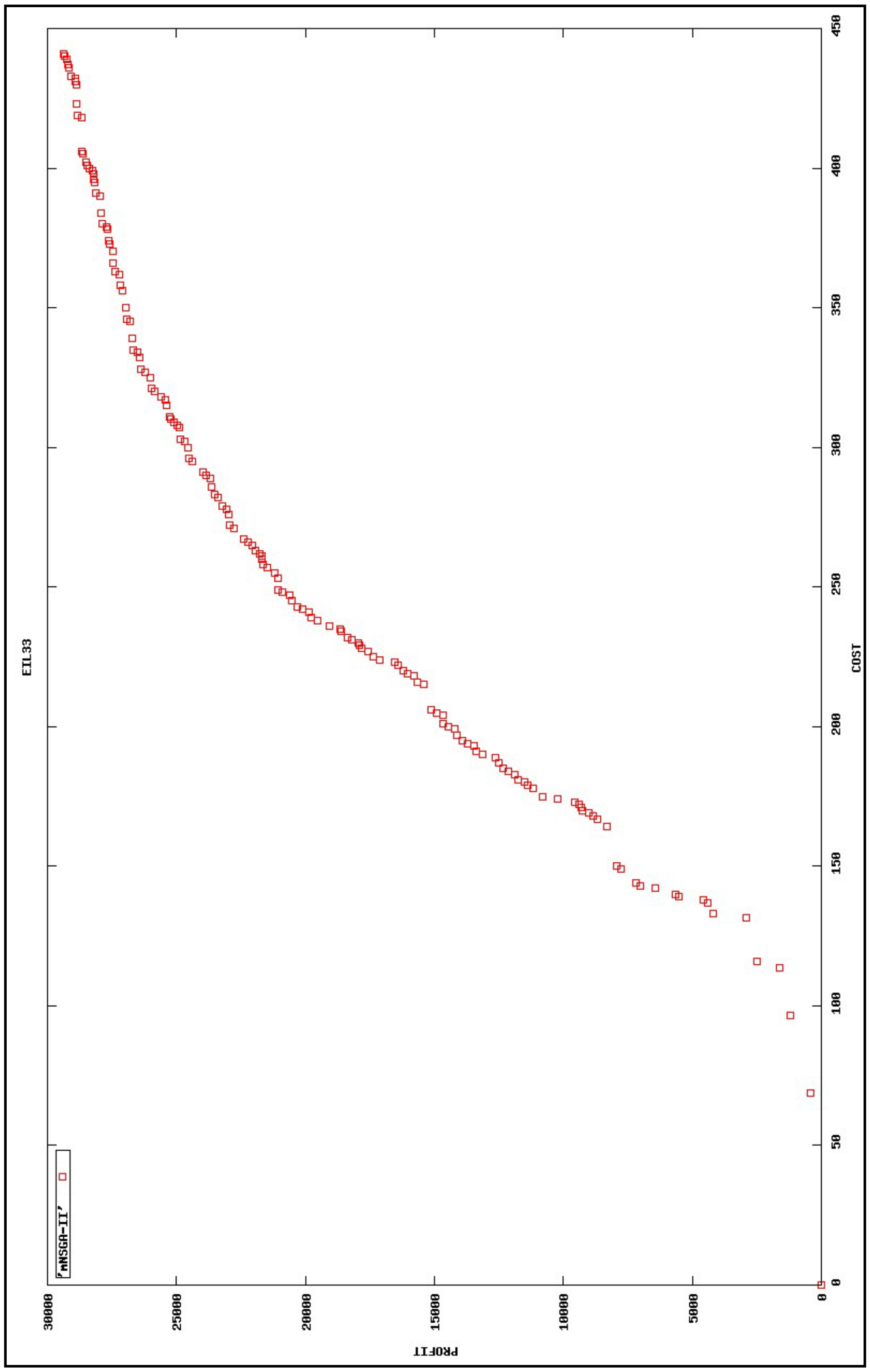


Figure 19 *Pareto front for EIL33.*

Table 13 *Pareto front generated for EIL33.*

COST	PROFIT	VISITED CITIES
0	0	1
69	400	1-4
97	1.200	1-5
114	1.600	1-4-5
116	2.500	1-31
131	2.900	1-4-31
133	4.200	1-31-32
137	4.420	1-3-4-6-7-8-12
138	4.570	1-3-4-6-7-8-12-13
139	5.520	1-3-4-6-7-8-12-33
140	5.670	1-3-4-6-7-8-12-13-33
142	6.420	1-3-4-6-7-8-9-12-33
143	7.020	1-3-4-6-7-8-9-10-12-33
144	7.170	1-3-4-6-7-8-9-10-12-13-33
149	7.770	1-3-4-6-7-8-9-10-11-12-33
150	7.920	1-3-4-6-7-8-9-10-11-12-13-33
164	8.320	1-4-6-7-8-9-10-12-32-33
167	8.680	1-3-4-7-8-9-10-12-32-33
168	8.870	1-3-4-6-7-8-9-10-12-13-32-33
169	9.020	1-2-4-6-7-8-9-10-12-32-33
170	9.270	1-2-4-6-7-8-9-10-12-14-32-33
171	9.320	1-4-6-7-8-12-31-32-33
172	9.380	1-2-3-4-7-8-9-10-12-32-33
173	9.570	1-2-3-4-6-7-8-9-10-12-13-32-33
174	10.220	1-4-6-7-8-9-12-31-32-33
175	10.820	1-4-6-7-8-9-10-12-31-32-33
178	11.180	1-3-4-7-8-9-10-12-31-32-33
179	11.370	1-3-4-6-7-8-9-10-12-13-31-32-33
180	11.520	1-2-4-6-7-8-9-10-12-31-32-33
181	11.770	1-2-4-6-7-8-9-10-12-14-31-32-33
183	11.880	1-2-3-4-7-8-9-10-12-31-32-33
184	12.130	1-2-3-4-7-8-9-10-12-14-31-32-33
185	12.320	1-2-3-4-6-7-8-9-10-12-13-14-31-32-33
187	12.520	1-2-4-6-7-8-9-10-11-12-14-31-32-33
189	12.630	1-2-3-4-7-8-9-10-11-12-31-32-33
190	13.120	1-2-4-6-7-8-9-10-12-15-31-32-33
191	13.370	1-2-4-6-7-8-9-10-12-14-15-31-32-33
193	13.480	1-2-3-4-7-8-9-10-12-15-31-32-33
194	13.730	1-2-3-4-7-8-9-10-12-14-15-31-32-33
195	13.920	1-2-3-4-6-7-8-9-10-12-13-14-15-31-32-33
197	14.120	1-2-4-6-7-8-9-10-11-12-14-15-31-32-33
199	14.230	1-2-3-4-7-8-9-10-11-12-15-31-32-33
200	14.480	1-2-3-4-7-8-9-10-11-12-14-15-31-32-33
201	14.670	1-2-3-4-6-7-8-9-10-11-12-13-14-15-31-32-33
204	14.680	1-2-3-4-7-8-9-10-11-12-15-16-31-32-33
205	14.930	1-2-3-4-7-8-9-10-11-12-14-15-16-31-32-33
206	15.120	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-31-32-33
215	15.420	1-2-4-6-7-8-9-10-12-15-27-31-33
216	15.670	1-2-4-6-7-8-9-10-12-14-15-27-31-33
218	15.780	1-2-3-4-7-8-9-10-12-15-27-31-33
219	16.030	1-2-3-4-7-8-9-10-12-14-15-27-31-33
220	16.220	1-2-3-4-6-7-8-9-10-12-13-14-15-27-31-33
222	16.420	1-2-4-6-7-8-9-10-11-12-14-15-27-31-33
223	16.520	1-2-4-6-7-8-9-12-15-27-31-32-33
224	17.120	1-2-4-6-7-8-9-10-12-15-27-31-32-33
225	17.370	1-2-4-6-7-8-9-10-12-14-15-27-31-32-33
227	17.570	1-2-4-6-7-8-9-10-12-15-16-27-31-32-33
228	17.820	1-2-4-6-7-8-9-10-12-14-15-16-27-31-32-33
229	17.920	1-2-3-4-6-7-8-9-10-12-13-14-15-27-31-32-33
230	17.930	1-2-3-4-7-8-9-10-12-15-16-27-31-32-33
231	18.180	1-2-3-4-7-8-9-10-12-14-15-16-27-31-32-33

Table 14 *Cont'd.*

COST	PROFIT	VISITED CITIES
232	18.370	1-2-3-4-6-7-8-9-10-12-13-14-15-16-27-31-32-33
234	18.620	1-4-6-7-8-9-10-12-14-15-18-26-27-31-32-33
235	18.670	1-2-3-4-6-7-8-9-10-11-12-13-14-15-27-31-32-33
236	19.070	1-4-6-7-8-9-10-12-14-15-16-18-26-27-31-32-33
238	19.520	1-2-4-6-7-8-9-10-12-15-16-18-26-27-31-32-33
239	19.770	1-2-4-6-7-8-9-10-12-14-15-16-18-26-27-31-32-33
241	19.880	1-2-3-4-7-8-9-10-12-15-16-18-26-27-31-32-33
242	20.130	1-2-3-4-7-8-9-10-12-14-15-16-18-26-27-31-32-33
243	20.320	1-2-3-4-6-7-8-9-10-12-13-14-15-16-18-26-27-31-32-33
245	20.520	1-2-4-6-7-8-9-10-11-12-14-15-16-18-26-27-31-32-33
247	20.630	1-2-3-4-7-8-9-10-11-12-15-16-18-26-27-31-32-33
248	20.880	1-2-3-4-7-8-9-10-11-12-14-15-16-18-26-27-31-32-33
249	21.070	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-18-26-27-31-32-33
253	21.080	1-2-3-4-7-8-9-10-11-12-13-15-16-19-20-26-27-31-32-33
255	21.180	1-2-3-4-7-8-9-10-11-12-13-15-18-19-20-26-27-31-32-33
257	21.480	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-26-27-31-32-33
258	21.630	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-26-27-31-32-33
260	21.670	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-18-26-27-28-31-32-33
261	21.680	1-3-4-7-8-9-10-11-12-13-15-16-19-20-23-26-27-31-32-33
262	21.780	1-2-3-4-7-8-9-10-11-12-15-19-20-23-26-27-31-32-33
263	21.930	1-2-3-4-7-8-9-10-11-12-13-15-19-20-23-26-27-31-32-33
265	22.080	1-3-4-7-8-9-10-11-12-15-16-18-19-20-23-26-27-31-32-33
266	22.230	1-2-3-4-7-8-9-10-11-12-15-16-19-20-23-26-27-31-32-33
266	22.230	1-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-31-32-33
267	22.380	1-2-3-4-7-8-9-10-11-12-13-15-16-19-20-23-26-27-31-32-33
271	22.780	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-23-26-27-31-32-33
272	22.930	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-31-32-33
276	22.970	1-2-3-4-6-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-31-32-33
278	23.080	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-22-23-26-27-31-32-33
279	23.230	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-22-23-26-27-31-32-33
282	23.380	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-23-26-27-28-31-32-33
283	23.530	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-28-31-32-33
286	23.630	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-26-27-31-32-33
289	23.680	1-3-4-7-8-9-10-11-12-15-16-18-19-20-23-26-27-28-29-31-32-33
289	23.680	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-22-23-26-27-28-31-32-33
289	23.680	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-25-26-27-31-32-33
290	23.830	1-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-28-29-31-32-33
290	23.830	1-2-3-4-7-8-9-10-11-12-15-16-19-20-23-26-27-28-29-31-32-33
290	23.830	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-22-23-26-27-28-31-32-33
291	23.980	1-2-3-4-7-8-9-10-11-12-13-15-16-19-20-23-26-27-28-29-31-32-33
295	24.380	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-23-26-27-28-29-31-32-33
296	24.530	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-28-29-31-32-33
300	24.570	1-2-3-4-6-7-8-9-10-11-12-13-15-16-18-19-20-23-26-27-28-29-31-32-33
302	24.680	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-22-23-26-27-28-29-31-32-33
303	24.830	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-22-23-26-27-28-29-31-32-33
307	24.870	1-2-3-4-6-7-8-9-10-11-12-13-15-16-18-19-20-22-23-26-27-28-29-31-32-33
308	24.980	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-24-25-26-27-28-31-32-33
309	25.080	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-21-22-23-26-27-28-29-31-32-33
310	25.230	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-26-27-28-29-31-32-33
311	25.280	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-25-26-27-28-29-31-32-33
315	25.380	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-21-23-24-25-26-27-28-31-32-33
317	25.430	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-22-23-25-26-27-28-29-31-32-33
318	25.580	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-22-23-25-26-27-28-29-31-32-33
320	25.830	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-23-24-25-26-27-28-29-31-32-33
321	25.980	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-23-24-25-26-27-28-29-31-32-33
325	26.020	1-2-3-4-6-7-8-9-10-11-12-13-15-16-18-19-20-23-24-25-26-27-28-29-31-32-33
327	26.230	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-21-23-24-25-26-27-28-29-31-32-33
328	26.380	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-21-23-24-25-26-27-28-29-31-32-33
332	26.420	1-2-3-4-6-7-8-9-10-11-12-13-15-16-18-19-20-21-23-24-25-26-27-28-29-31-32-33
334	26.530	1-2-3-4-7-8-9-10-11-12-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
335	26.680	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
339	26.720	1-2-3-4-6-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
345	26.780	1-2-3-4-7-8-9-10-11-12-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33

Table 15 *Cont'd.*

COST	PROFIT	VISITED CITIES
346	26.930	1-2-3-4-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
350	26.970	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
356	27.080	1-2-3-4-7-8-9-10-11-12-13-15-16-17-18-19-20-21-23-24-25-26-27-28-29-31-32-33
358	27.180	1-2-3-4-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
362	27.230	1-2-3-4-7-8-9-10-11-12-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
363	27.380	1-2-3-4-7-8-9-10-11-12-13-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
366	27.470	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
370	27.480	1-2-3-5-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
373	27.580	1-2-3-4-5-7-8-9-10-11-12-13-15-16-18-19-20-21-23-24-25-26-27-28-29-31-32-33
374	27.630	1-2-3-4-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
378	27.670	1-2-3-4-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
379	27.730	1-2-3-4-5-7-8-9-10-11-12-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
380	27.880	1-2-3-4-5-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
384	27.920	1-2-3-4-5-6-7-8-9-10-11-12-13-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
390	27.980	1-2-3-4-5-7-8-9-10-11-12-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
391	28.130	1-2-3-4-5-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
395	28.180	1-2-3-4-5-7-8-9-10-11-12-14-15-16-18-19-20-21-23-24-25-26-27-28-29-30-31-32-33
396	28.220	1-2-3-4-5-6-7-8-9-10-11-12-14-15-16-18-19-20-21-23-24-25-26-27-28-29-30-31-32-33
398	28.230	1-2-3-4-5-7-8-9-10-11-12-13-14-15-16-18-19-20-22-23-24-25-26-27-28-29-30-31-32-33
399	28.270	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-22-23-24-25-26-27-28-29-30-31-32-33
400	28.400	1-2-3-4-5-8-9-10-11-12-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
401	28.480	1-2-3-4-5-7-8-9-10-11-12-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
402	28.520	1-2-3-4-5-6-7-8-9-10-11-12-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
405	28.630	1-2-3-4-5-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
406	28.670	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
418	28.680	1-2-3-4-5-7-8-9-10-11-12-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
419	28.830	1-2-3-4-5-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
423	28.870	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-31-32-33
430	28.880	1-2-3-4-5-7-8-9-10-11-12-14-15-16-17-18-19-20-21-23-24-25-26-27-28-29-30-31-32-33
431	28.920	1-2-3-4-5-6-7-8-9-10-11-12-14-15-16-17-18-19-20-21-23-24-25-26-27-28-29-30-31-32-33
432	28.930	1-2-3-4-5-7-8-9-10-11-12-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
433	29.080	1-2-3-4-5-7-8-9-10-11-12-13-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
436	29.180	1-2-3-4-5-7-8-9-10-11-12-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
437	29.220	1-2-3-4-5-6-7-8-9-10-11-12-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
439	29.250	1-2-3-4-5-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
440	29.330	1-2-3-4-5-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33
441	29.370	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33

APPENDIX D: CLEMENTINE MODELS USED FOR POST-OPTIMALITY ANALYSIS

Models Used: C&R Tree

Graphs Used: Histogram, Collection, and Web.

Figure 20 is an instance from the constructed model. As it can be seen from the figure, only one objective can be considered at a time in all analysis.

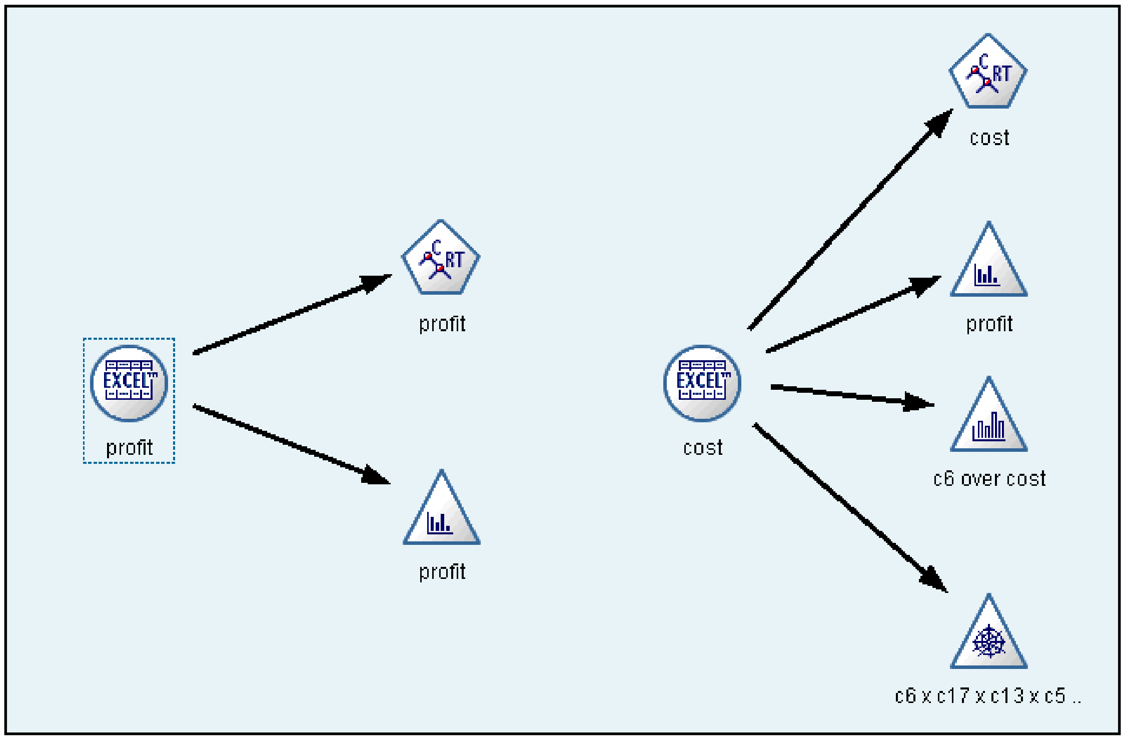


Figure 20 *An instance from SPSS Clementine.*