Traffic Engineering in MPLS Networks with Multiple Objectives: Modeling and Optimization

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades einer Doktorin
der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Selin Kardelen Cerav-Erbas
Master of Science in Industrial and Systems Engineering
aus Manisa, Türkei

Berichter: Universitätsprofessor Dr. Rudolf Mathar
Universitätsprofessor Dr. Otto Spaniol

Tag der mündlichen Prüfung: 21 Juli 2004

Acknowledgments

There is a list of people without whom this study would not have come to an end. Their technical and psychological support has been a great motivation for me to complete this thesis.

Initially, I would like to thank my supervisor, Prof. Dr. Rudolf Mathar, for giving me the opportunity to do research in his department. As an expert, his technical and strategic advices have provided me clear solutions to the difficult questions encountered during the last 3 years.

I would like to thank also Prof. Dr. Otto Spaniol for his help and support to my research. The environment which he has provided among the group of "Graduate College" has been very friendly and enlightening.

My husband, Çağkan, has been very patient with me during the last 3 years. His both technical and moral support has made me stay calm and positive during difficult times.

Priv.-Doz. Dr. Yubao Guo and Prof. Dr. Dr. h.c. Hubertus Jongen, provided me great help to get prepared for the exam.

My sisters, Özlem and Yasemin, and my mother, Ayten, have been very supportive, although they were very far away from Germany. "Teşekkürler!"

I would like to thank all of my colleagues in the department in Aachen: Eric Beutner, Daniel Catrein, Anke Feiten, Wolfgang Herff, Lorenz Imhof, Birgit Kaiser, Michael Meier, Natalie and Michael Naehrig, Ole Nordhoff, Michael Reyer, Michael Schmeink, Völker Schmitz, Reinhard Schwab, Andreas Tillmans, and Iskender Yasar. Their help

and friendship have made me feel like at home in Germany.

Lastly, I would thank my new colleagues in Louvain-la-Neuve. Prof. Bernard Fortz and Hakan Ümit have been very understanding when I needed some extra time for my PhD work. I am also grateful to Roman Kasperzec and Mathias Lorenz especially for their help in the translation of the abstract.

Abstract

With the increasing usage of QoS-based applications, traffic engineering in communication networks has become more crucial. For the optimal utilization of network resources, service providers need to consider multiple criteria that can be customer- or traffic-oriented. Multiprotocol Label Switching (MPLS) has been developed to apply more convenient traffic engineering in autonomous systems. This thesis focuses on the multiobjective optimization of Label Switched Path design problem in MPLS networks. Minimal routing cost, optimal load-balance in the network, and minimal splitting of traffic form the objectives. The problem is formulated as a zero-one mixed integer program and aims at exploring the trade-offs among the objectives. The integer constraints make the problem NP-hard. In the thesis first both the exact and heuristic multiobjective optimization approaches are discussed, and then a heuristic framework based on simulated annealing is developed. Various search strategies within the framework are investigated and experimental studies are carried out for a performance comparison.

Zusammenfassung

Mit zunehmenden Einsatz von QoS-Anwendungen wird Traffic Engineering in Kommunikationsnetzen immer wichtiger. Für eine optimale Ressourcenverwendung im Netz müssen Service Provider mehrere Kriterien beachten, die kunden- oder verkehrsorientiert sein können. Multiprotocol Label Switching (MPLS) ist entwickelt worden, um Traffic Engineering in autonomen Systemen bequemer zu gestalten. Die vorliegende Dissertation konzentriert sich auf die Mehrzieloptimierung des Label Switched Path Design-Problems in MPLS Netzen. Minimale Routingkosten, optimale Eingabebalance im Netz und minimale Verkehrsaufspaltung sind die verfolgten Zielfunktionen. Die Problemformulierung führt zu einem binären gemischt-ganzzahligen Programm, mit dem die Wechselwirkung zwischen den Zielfunktionen untersucht werden. Wegen der Ganzzahligkeit erweist sich das Problem als NP-hart. In der vorliegenden Dissertation werden zunächst Ansätze zur exakten Lösung und Heuristiken der Mehrzieloptimierung untersucht. Anschliessend werden auf Simulated Annealing basierende heuristische Ansätze entwickelt. Hiermit werden verschiedene Suchalgorithmen bezüglich ihrer Leistungsfähigkeit experimentell verglichen.

Contents

1 Introduction

During the last ten years, the importance of the Internet has increased dramatically due to the requirements of the global world. By the use of Internet-based applications, people around the world can exchange data, transfer money and bonds, and are informed by the latest news from the world.

With the introduction of QoS (Quality of Service)-based applications to the Internet, many novel networking technologies have been developed. Traffic engineering has become crucial within these technologies to meet the increasing demand and variety in customers' requests. Network optimization has become a great challenge for Internet Service Providers (ISPs) due to the strict constraints imposed by the limited resource capacities and customers' demands.

Under those issues, most of the optimization problems within traffic engineering are multiobjective in nature, as it is in real-life. The objectives aiming at the customers' satisfaction conflict mostly with the objectives related to the utilization performances of the network. The customers require faster and more reliable connections, whereas the network managers prefer a more stabilized and more balanced network. Being aware of the trade-offs existing among multiple objectives results in a more extensive management of the network. Thus, multiobjective optimization (MOP) of networks may bring additional competence to ISPs in the highly competitive sector.

From a theoretical point of view, an important issue is the complexity of the problems arising in the large systems. The optimization problems within a network can be too difficult (NP-hard), and they may require very long run times to obtain an optimal solution. Theoretically, it has been shown for NP-hard problems that the run-times of exact solution approaches increment dramatically with the size of the problem. Under tight time constraints, it is more practical to develop specific algorithms which try to locate solutions close to the optima in much shorter times. These approximate algorithms are called heuristics in general, and their development has been a very popular research topic in the past years.

The contributions of this thesis are around the Multiprotocol Label Switching (MPLS) networks which have been developed to support QoS routing and to allow more comfortable traffic engineering within an autonomous system. MPLS networks are based on a label forwarding paradigm, which allows explicit routing additional to classification and prioritization of the traffic. The multiobjective optimization problem studied in this thesis in detail is formulated as a mixed zero-one integer pro-

gramming problem. Three conflicting objectives are taken into consideration, namely minimal total routing costs, optimal load balancing, and minimal traffic splitting. The contributions of this thesis are related to this MOP problem:

- A robust mathematical model is developed, by which networks with various characteristics can be represented.
- Discussions about the meaning and importance of load balancing are given.
- The available exact and approximate solution approaches available in the literature for MOP are investigated.
- Theoretical analyses are carried out to explore the attributes (complexity of the problem, characteristics of the trade-off curves) of the multiobjective traffic engineering problem.
- Based on the careful investigation of the available heuristic algorithms in the literature, an approximate algorithm framework utilizing linear programming (LP) and simulated annealing is developed.
- The performance consequences of various search strategies within the framework are compared using four instances of the problem with different sizes.

The outline of the thesis is as follows:

The next chapter introduces the basic terminology within MOP and discusses some of the exact solution approaches in detail.

In Chapter 3, the attributes of MPLS networks are shortly explained, the multiobjective traffic engineering problem and its mathematical model are introduced. This chapter also includes some theoretical analyses about the nature and the complexity of the problem. At the end of the chapter, an exact solution approach is applied in a small case study.

Chapter 4 investigates the available heuristics which have been developed in the literature for MOP. In the last part of the chapter, the performance assessment metrics used in the thesis to evaluate the heuristics are introduced.

In Chapter 5, the heuristic framework proposed for the multiobjective traffic engineering problem is explained in detail. The algorithm variants within this framework are introduced. Experiments are carried out in the final part.

The thesis is concluded in Chapter 6.

2 Multiobjective Optimization and Exact Solution Approaches

Although most of the studies in operations research focus on optimization problems with a single objective, most real-world engineering optimization problems combine multiple objectives. In multiobjective problems, it rarely happens that all of the objectives can be optimized simultaneously; it is generally the case that the objectives conflict with each other. MOP problems occur in various engineering systems apart from telecommunications: for instance, embedded system design [19], [23], [87], production system management [39], [75], applications of civil and construction engineering [12], [54].

In single objective optimization (SOP) problems, the feasible set is totally ordered, i.e., we can rank all feasible solutions with regard to some objective function. Nevertheless, in a problem with multiple objectives it is not possible to obtain a total ordering of the feasible solutions (i.e., the feasible set is partially ordered). When two solutions are compared, we may observe that one solution performs better in terms of one objective, while the other solution scores better on another objective. So, there exist trade-offs among the objectives and one is interested in locating the set of best alternative solutions, the so-called *Pareto optimal* solutions. The decision maker can rank this set of solutions according to his preferences. This chapter discusses the relationship of MOP with decision theory, additional to the introduction of key concepts and terminology within the MOP.

2.1 Search versus Decision Making

MOP entails two conceptually distinct tasks: *search* and *decision making* [35]. Search is related to the processes where the feasible solutions are visited in order to find the Pareto optimal solutions. Decision making refers to the ranking process of alternative solutions. A rational human decision maker determines preferences among the conflicting objectives.

The methodologies in MOP can be classified into three primary groups, depending on how the search and decision making tasks are handled. Horn [35] discusses the state-of-art approaches for each of these methodologies in detail.

- **Decision making before search:** In this approach, the objectives are aggregated into a single objective function where the preference information of the decision maker is represented. The aggregation can be carried out in two ways: scalar combination or lexicographical ordering (ranking according to the importance) of the objectives. The aggregation of the objectives into a single objective function requires domain-specific knowledge about the ranges and the behavior of the functions. However, this kind of deep knowledge about the functions is usually not available, since the functions and/or feasible set may be too complex. However, this methodology has the advantage that the SOP strategies can be applied to the problem with the aggregated objective.

- **Search before decision making:** The feasible set is searched to find a set of *best alternative*s, without giving any information about preferences. Decision making considers only the reduced set of alternatives. For most of the real-life problems, gaining fundamental knowledge about the problem and alternative solutions can be very helpful in realizing the conflicts that are inherent in the problem. Performing the search before decision making makes this favorable circumstances possible, however the search process becomes more difficult with the exclusion of the preferences of the decision maker.

- **Integrating search and decision making:** This approach includes the interactive methods where the preferences of the decision maker are used during the search process. At each iteration, the result of the search is evaluated by the decision maker in order to update the preferences. The search space is then reduced and the direction of the search is restricted to some particular regions according to the preferences of the decision maker. This last methodology integrates the theory of decision making into optimization theory.

The interest of this thesis is in the second category. The study aims at finding the set of all Pareto optimal solutions, or the set of solutions which is a good representation and approximation of the Pareto optimal set. This concept is sometimes called also as *vector optimization* [77], or *Pareto optimization* [45]. In this study we will use the term, multiobjective optimization, which is most commonly used in the literature.

Approximating the set of Pareto optimal solutions instead of locating a single solution allows a decision maker to see the trade-offs among the objectives. As it is stated in [19], knowing the shape of the trade-off curve can be very important to the decision maker. Figure 2.1 illustrates an example trade-off curve for a system design problem with two objectives: minimization of the investment cost and minimization of the total time necessary for the task completion. An approach aiming at a compromise solution may end up with the solution A corresponding to a system design with 40 units of total cost and 80 units of system time. However, the system designer may probably prefer the solution B, since he/she saves in time of 30 units for a cost

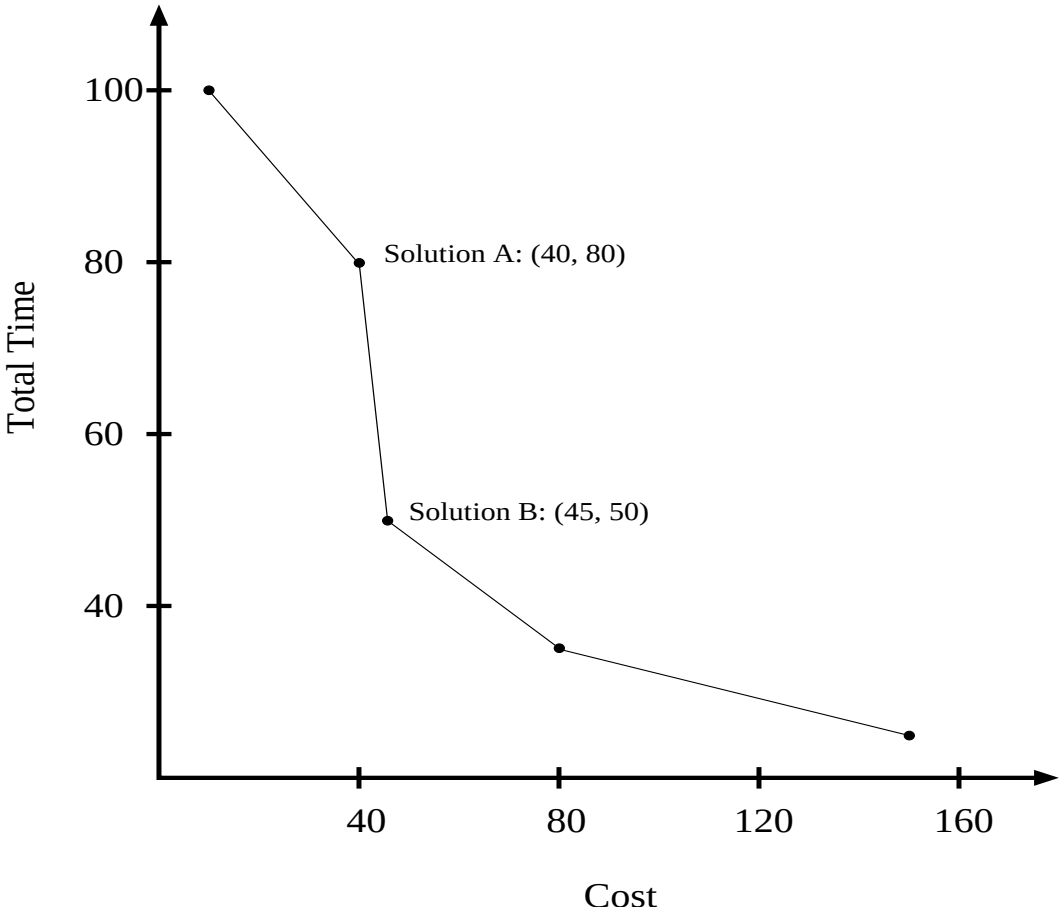

Figure 2.1: The cognition of the trade-off curve can be important to the decision maker.

penalty of only 5 units. Hence, locating the trade-off curve helps the decision maker to realize the conflicts among the objectives.

Before proceeding further, some definitions are given in order to clarify the key concepts and issues in MOP. The definitions, concepts and terminology presented in this thesis are widely borrowed from the studies in [20], [45], [77] and [87].

2.2 Key Concepts and Terminology

Definition 2.1. *A* **multiobjective optimization** *problem is defined as follows:*

$$\min \ \mathbf{f}(\mathbf{v}) = (f_1(\mathbf{v}), \ldots, f_Q(\mathbf{v}))$$

$$\textit{subject to} \ \ \mathbf{v} \in \mathbf{X}_f$$

where $Q \geq 2$ is the number of objectives, $\mathbf{v}$ and $\mathbf{X}_f$ denote the **decision vector** *and* **feasible set**, *respectively. The image of $\mathbf{X}_f$ under the vector valued function $\mathbf{f} = (f_1, \ldots, f_Q)$ is denoted by $\mathbf{Z}_f = \mathbf{f}(\mathbf{X}_f)$. $\mathbf{z} \in \mathbf{Z}_f$ is called the* **objective vector**.

Without loss of generality, a problem with all of the objectives in minimization form is assumed. Any objective function in maximization form can be converted to minimization by multiplying with -1.

Definition 2.2. *The* **component-wise order relation** $<$ *between two objective vectors is defined as follows: $\mathbf{z}^1 < \mathbf{z}^2$ iff $\mathbf{z}^1_i \leq \mathbf{z}^2_i, i = 1, \ldots, Q$ and $\mathbf{z}^1 \neq \mathbf{z}^2$.*

Definition 2.3. *The* **weak component-wise order relation** $\leq$ *between two objective vectors is defined as follows: $\mathbf{z}^1 \leq \mathbf{z}^2$ iff $\mathbf{z}^1_i \leq \mathbf{z}^2_i, i = 1, \ldots, Q$.*

Definition 2.4. *The* **strict component-wise order relation** $\leq$ *between two objective vectors is defined as follows: $\mathbf{z}^1 \ll \mathbf{z}^2$ iff $\mathbf{z}^1_i < \mathbf{z}^2_i, i = 1, \ldots, Q$.*

Definition 2.5. *For any two decision vectors $\mathbf{v}^1$ and $\mathbf{v}^2$ following relations are defined:*

$$
\begin{aligned}
\mathbf{v}^1 \textit{ dominates } \mathbf{v}^2 &: \\
\mathbf{v}^1 \prec \mathbf{v}^2 \quad &\textit{iff} \quad \mathbf{f}(\mathbf{v}^1) < \mathbf{f}(\mathbf{v}^2) \\
\mathbf{v}^1 \textit{ weakly dominates } \mathbf{v}^2 &: \\
\mathbf{v}^1 \preceq \mathbf{v}^2 \quad &\textit{iff} \quad \mathbf{f}(\mathbf{v}^1) \leq \mathbf{f}(\mathbf{v}^2) \\
\mathbf{v}^1 \textit{ is indifferent to } \mathbf{v}^2 &: \\
\mathbf{v}^1 \sim \mathbf{v}^2 \quad &\textit{iff} \quad \mathbf{f}(\mathbf{v}^1) \not\leq \mathbf{f}(\mathbf{v}^2) \\
&\qquad\qquad \textit{and } \mathbf{f}(\mathbf{v}^2) \not\leq \mathbf{f}(\mathbf{v}^1).
\end{aligned}
$$

Definition 2.6. *A decision vector $\mathbf{v}^* \in \mathbf{X}_f$ is said to be* **Pareto optimal** *iff*

$$\nexists \mathbf{v} \in \mathbf{X}_f \textit{ such that } \mathbf{f}(\mathbf{v}) < \mathbf{f}(\mathbf{v}^*).$$

Stated differently, $\mathbf{v}^* \in \mathbf{X}_f$ is **Pareto optimal** *iff there is no $\mathbf{v} \in \mathbf{X}_f$ such that $f_i(\mathbf{v}) \leq f_i(\mathbf{v}^*)$ for all $i = 1, \ldots, Q$ and $f_j(\mathbf{v}) < f_j(\mathbf{v}^*)$ for at least one j. If $\mathbf{v}^*$ is Pareto optimal, the corresponding objective vector $\mathbf{z}^* = \mathbf{f}(\mathbf{v}^*)$ is called* **efficient**. *The set of all Pareto optimal solutions is called the* **Pareto optimal set** *and denoted by $\mathbf{X}_p$. Similarly, the set of all efficient points is called the* **efficient set** *and denoted by $\mathbf{Z}_p$. The* **Pareto (optimal) front** *is obtained by plotting the set of all efficient vectors in the objective space.*

Definition 2.7. *A decision vector $\mathbf{v}^* \in \mathbf{X}_f$ and its corresponding objective vector $\mathbf{z}^* = \mathbf{f}(\mathbf{v}^*)$ are called* **nondominated** *with regard to a set $\mathbf{X}_a \subseteq \mathbf{X}_f$ iff*

$$\nexists \mathbf{v} \in \mathbf{X}_a \textit{ such that } \mathbf{f}(\mathbf{v}) < \mathbf{f}(\mathbf{v}^*).$$

If $\mathbf{X}_a$ *is equal to* $\mathbf{X}_f$*, then* $\mathbf{v}^*$ *and* $\mathbf{z}^*$ *are Pareto optimal and efficient, correspondingly. The plot obtained only by the nondominated objective vectors build the* **nondominated front** *with regard to* $\mathbf{X}_a$*.*

Remark 2.8. *Since this is a topic which is newly developed, there exist slight differences in the use of above concepts in various studies. Similarly, the terminology for the methods of MOP is not unique in the literature.*

Although the studies on single objective programming usually focus on the decision space, in multiobjective programming mostly the objective space is studied. Regarding Definitions 2.6 and 2.10, while the concept of Pareto optimality corresponds to the decision space, efficiency is related to the objective space. In this study, the concept of dominance is used in both spaces. So, Definition 2.5 applies for the objective space as well.

The subset corresponding to Definition 2.7 can be the set of feasible solutions encountered during a heuristic algorithm. So, the output of the algorithm will be the nondominated solutions with regard to the set of all visited solutions.

The next discussion helps us to detect the efficient points graphically [78]. Let $\mathbb{R}^Q_-$ be the nonpositive orthant;

$$\mathbb{R}^Q_- = \{\mathbf{z} \in \mathbb{R}^Q \mid z_i \leq 0\}.$$

To determine easily whether an objective vector is efficient or not efficient, $\mathbb{R}^Q_-$ is translated so that the origin coincides with the vector at hand. In fact, the whole dominance relation of a particular point with other vectors in the feasible set can be determined with the translated orthants. Following, the translation is achieved with the Minkowski set addition, $\oplus$.

Corollary 2.9 ([78]). *An objective vector* $\mathbf{z}^*$ *is* **efficient** *iff* $\mathbf{Z}_f \cap (\{\mathbf{z}^*\} \oplus \mathbb{R}^Q_-) = \{\mathbf{z}^*\}$*.*

The corollary implies that $\mathbf{z}^*$ is efficient if and only if $\mathbf{z}^*$ is the only point located in the intersection of the nonpositive orthant translated to $\mathbf{z}^*$ and $\mathbf{Z}_f$. The nonpositive orthant translated to $\mathbf{z}^*$ includes the points which dominates $\mathbf{z}^*$. In Figure 2.2, the point at hand is efficient, since this region (shaded lightly) does not include any other feasible point. The vectors which are dominated by $\mathbf{z}^*$ are located in the nonnegative orthant (shaded darkly) translated to $\mathbf{z}^*$. The points residing in unshaded quadrants are indifferent to $\mathbf{z}^*$.

The analytical study in this chapter requires the introduction of another concept in MOP, namely *weak Pareto optimality.*

Definition 2.10. *A decision vector* $\mathbf{v}^* \in \mathbf{X}_f$ *is called* **weakly Pareto optimal** *if there is no* $\mathbf{v} \in \mathbf{X}_f$ *such that* $\mathbf{f}(\mathbf{v}) < \mathbf{f}(\mathbf{v}^*)$ *(i.e.,* $f_i(\mathbf{v}) < f_i(\mathbf{v}^*)$ *for all* $i = 1, \ldots, Q$*).*

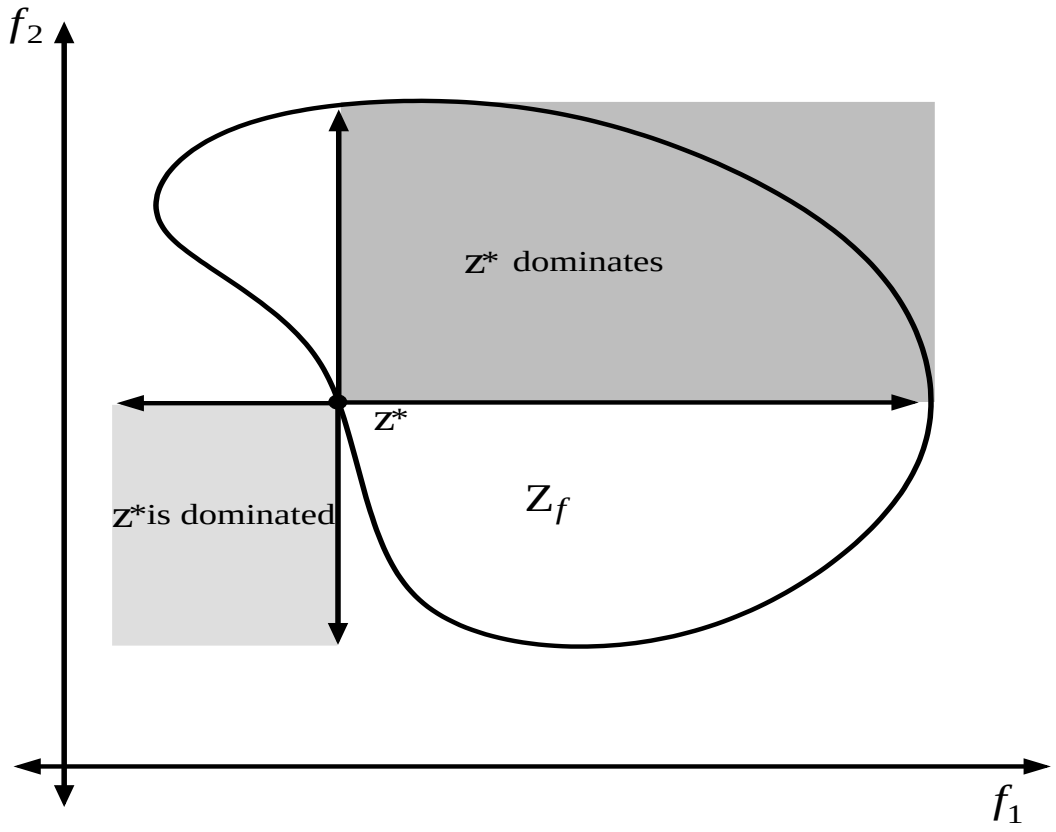

Figure 2.2: The efficient points can be detected graphically.

If $\mathbf{v}^*$ *is weakly Pareto optimal* $\mathbf{z}^* = \mathbf{f}(\mathbf{v}^*)$ *is called* **weakly efficient**[1]*. The set of all weakly Pareto optimal solutions is called the* **weakly Pareto optimal set** *and denoted with* $\mathbf{X}_w$*. The set of all weakly efficient points is called the* **weakly efficient set** *and denoted by* $\mathbf{Z}_w$*.*

From Definition 2.6 and 2.10, it is concluded that

$$\mathbf{X}_p \subseteq \mathbf{X}_w,$$

and correspondingly

$$\mathbf{Z}_p \subseteq \mathbf{Z}_w.$$

To illustrate the concepts explained above, consider the following Example 2.11.

[1]Please note that weak efficiency and weak dominance are not same.

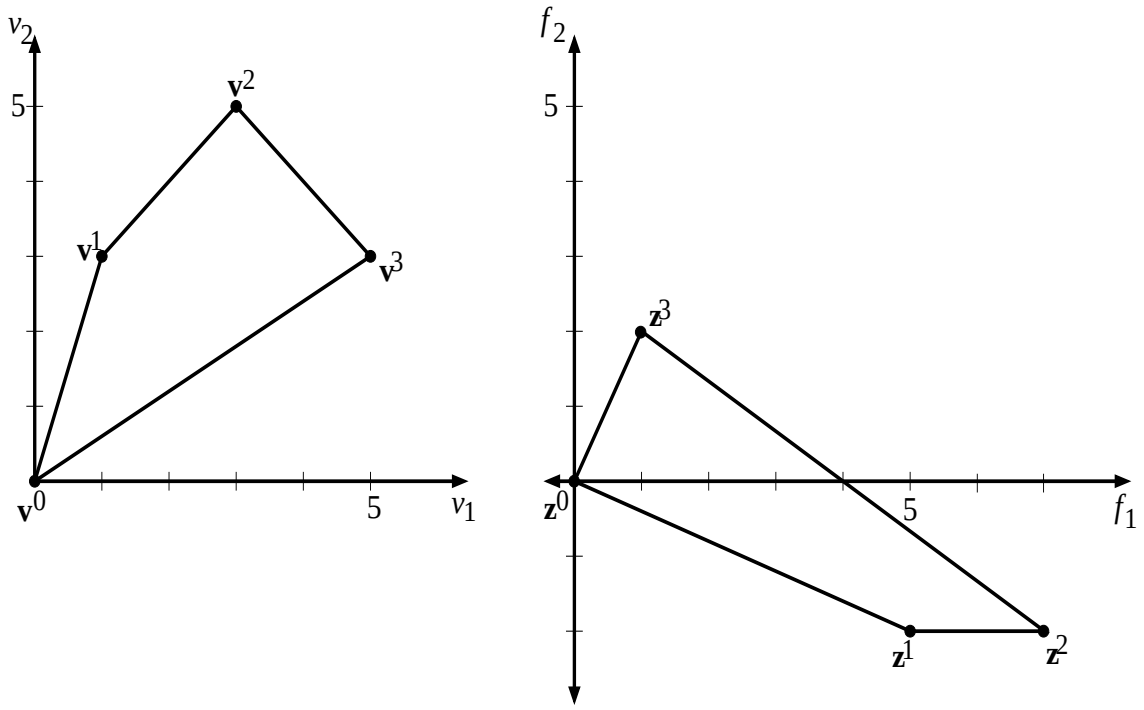

Figure 2.3: Example 2.11 illustrates the definitions related with multiobjective optimization.

Example 2.11. Consider the following problem with two objectives:

$$\begin{aligned} \min \quad & (f_1(\mathbf{v}) = 2v_2 - v_1, f_2(\mathbf{v}) = v_1 - v_2) \\ \text{subject to} \quad & v_2 - 3v_1 \leq 0, \\ & v_2 - v_1 \leq 2, \\ & 5v_2 - 3v_1 \leq 0, \\ & v_1 + v_2 \leq 8, \\ & v_1, v_2 \geq 0. \end{aligned}$$

The example is depicted in Figure 2.3. Figure 2.3(a) shows the feasible region in the decision space, whereas the image of the feasible region in the objective space is plotted in Figure 2.3(b). There are four extreme points, $\mathbf{v}^0 - \mathbf{v}^4$ and the following relation exist between these points:

$$\begin{gathered} \mathbf{v}^0 \prec \mathbf{v}^3 \text{ and } \mathbf{v}^1 \sim \mathbf{v}^3, \\ \mathbf{v}^1 \prec \mathbf{v}^2 \text{ and } \mathbf{v}^0 \sim \mathbf{v}^2, \\ \mathbf{v}^0 \sim \mathbf{v}^1. \end{gathered}$$

According to the relations we conclude that the Pareto optimal set consists of $\mathbf{v}^0$, $\mathbf{v}^1$ and the line between these two extreme points. Correspondingly, the efficient set includes $\mathbf{z}^0$, $\mathbf{z}^1$ and the line connecting them. The weakly Pareto optimal set includes additionally $\mathbf{v}^2$ and the line between $\mathbf{v}^1$ and $\mathbf{v}^2$. The weakly efficient set includes $\mathbf{z}^2$ and the line between $\mathbf{z}^1$ and $\mathbf{z}^2$.

2.3 Exact Methods

Most of the exact methods find Pareto optimal solutions by converting the MOP problem into a parameterized SOP problem. During the optimization process, the parameters are systematically changed for different versions of the problem. Multiple single objective searches are carried out in order to generate the Pareto optimal solutions.

The *weighted sum method*, *ϵ-constraint method*, and *lexicographic weighted Chebyshev method* are among the most popular exact methods [20], [77]. All of these methods are based on solving multiple SOP problems. The weighted sum method and the lexicographic weighted Chebyshev method aggregate the objective functions into a single objective by weighting parameters. The ϵ-constraint method is distinct from these two in the way that it transforms multiple objectives into a single objective by restating all but one objectives as constraints in the problem. In this chapter these three methods are discussed in detail.

2.3.1 The Weighted Sum Method

The most traditional approach in MOP is to combine the objective functions into a single function by using a weighted mean. In order to obtain the Pareto optimal set, the parametric SOP problem is solved for different weights (parameters) [20], [77].

In the weighted sum method, the quality of the solutions is distinguished by the nonnegative and strictly positive weights. The strictly positive weights ensure the Pareto optimality of the solution, whereas with nonnegative weights the method may end up with a weakly Pareto optimal solution. The following two theorems [20, Theorem 3.2 and 3.5] enlighten this issue.

Theorem 2.12. *Suppose* $\mathbf{v}^*$ *is an optimal solution of the problem*

$$\begin{aligned} \min \quad & \sum_{i=1}^{Q} \delta_i f_i(\mathbf{v}) \\ \textit{subject to} \quad & \mathbf{v} \in \mathbf{X}_f, \\ & \delta_i \geq 0 \qquad \forall i \in \{1, \ldots, Q\}, \\ & \sum_{i=1}^{Q} \delta_i = 1. \end{aligned}$$

Then $\mathbf{v}^* \in \mathbf{X}_w$.

Theorem 2.13. *Suppose* $\mathbf{v}^*$ *is an optimal solution of the problem*

$$\begin{aligned} \min \quad & \sum_{i=1}^{Q} \delta_i f_i(\mathbf{v}) \\ \textit{subject to} \quad & \mathbf{v} \in \mathbf{X}_f, \\ & \delta_i > 0 \qquad \forall i \in \{1, \ldots, Q\}, \\ & \sum_{i=1}^{Q} \delta_i = 1. \end{aligned}$$

Then $\mathbf{v}^* \in \mathbf{X}_p$.

The main drawback of the weighted sum method is that it cannot generate all the Pareto optimal solutions of problems with non-convex Pareto front. Although Theorem 2.13 gives a sufficient condition for Pareto optimality, it does not provide necessity. When the Pareto optimal front is non-convex, there exist some Pareto optimal solutions which are not optimal for any weighted sum optimization problem [20]. This situation is illustrated graphically.

In Figure 2.4, the image of a feasible set in the objective space is shown. We notice that the problem has a non-convex Pareto front and $\mathbf{z}^1, \mathbf{z}^2$ and $\mathbf{z}^3$ are efficient. $\mathbf{z}^1$ and $\mathbf{z}^3$ correspond to the optimum solutions of the problems with objective functions, min $\delta_1 f_1(\mathbf{v}) + \delta_2 f_2(\mathbf{v})$ and min $\delta_1' f_1(\mathbf{v}) + \delta_2' f_2(\mathbf{v})$, respectively. However, since the point $\mathbf{z}^2$ is on the non-convex part of the Pareto frontier, there is no weight vector for which the optimal solution yields this point.

2.3.2 The ϵ-Constraint Method

One of the common approaches to MOP is the ϵ-constraint method. It is introduced by Haimes et al. [33]. This approach is different from the other exact methods in the way

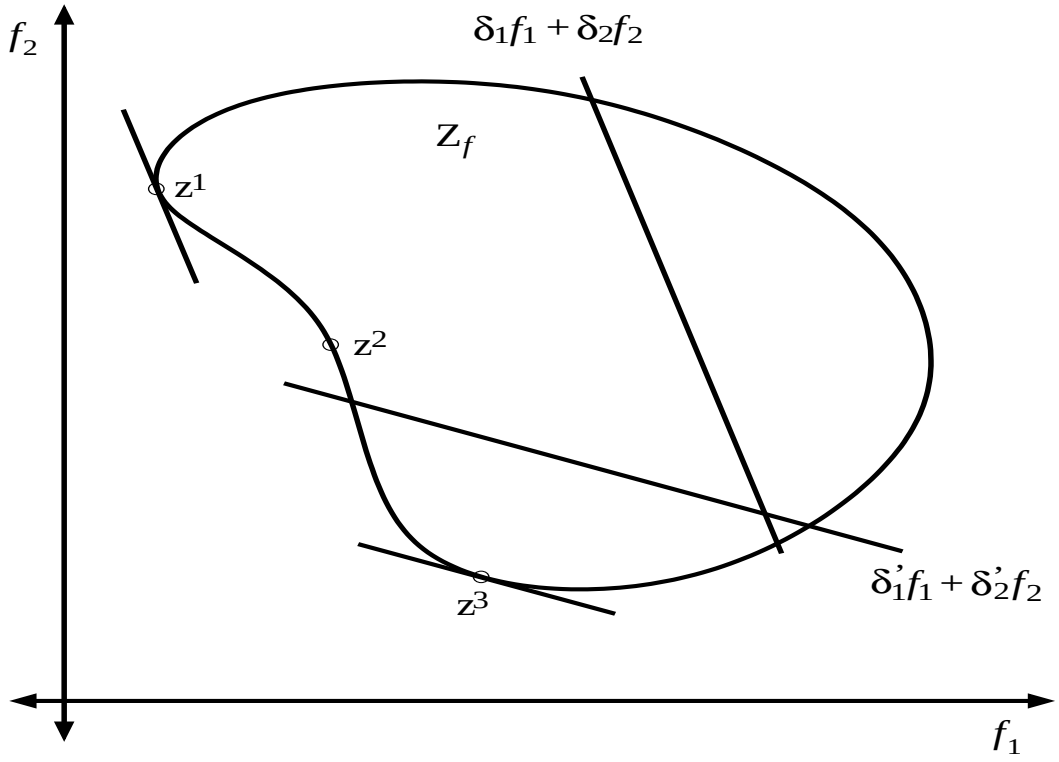

Figure 2.4: The weighted sum method fails to find the points at the non-convex part of the Pareto front.

that it is not based on the aggregation of the component objective functions. Contrary to the other methods, one of the objective functions is selected to be minimized, and the remaining ones are transformed into constraints. For a detailed discussion, the readers are pointed to [20].

For the ϵ-constraint method, the original MOP is translated to the following SOP problem:

$$\begin{aligned} S_k(\epsilon): \quad & \min \ \ f_k(\mathbf{v}) \\ \text{subject to} \quad & f_i(\mathbf{v}) \leq \epsilon_i \quad \forall i \in \{1,\ldots,Q\} \setminus k, \\ & \mathbf{v} \in \mathbf{X}_f. \end{aligned}$$

Here, $\epsilon \in \mathbb{R}^Q$. Note that, the k^{th} component of ϵ is of no importance in this problem.

The illustration of the method for a bi-objective problem is given in Figure 2.5. In order to find the efficient solutions, ϵ is varied by the optimizer at each iteration. In the figure, $\mathbf{z}^a$ is obtained by solving the problem $S_2(\epsilon^a)$, whereas $\mathbf{z}^b$ corresponds to the solution of the problem $S_2(\epsilon^b)$.

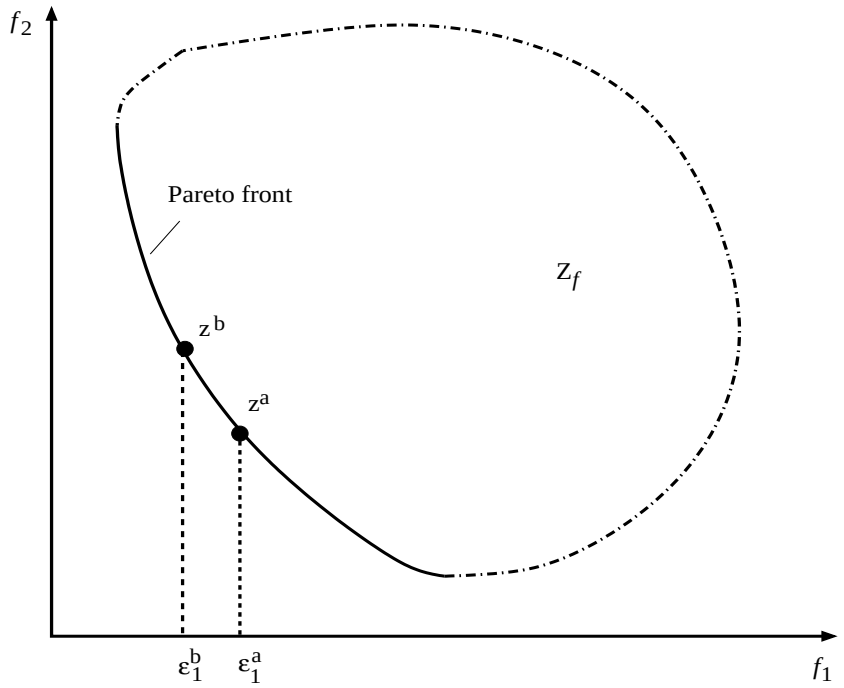

Figure 2.5: The ϵ-constraint method utilizes single objective optimization problems.

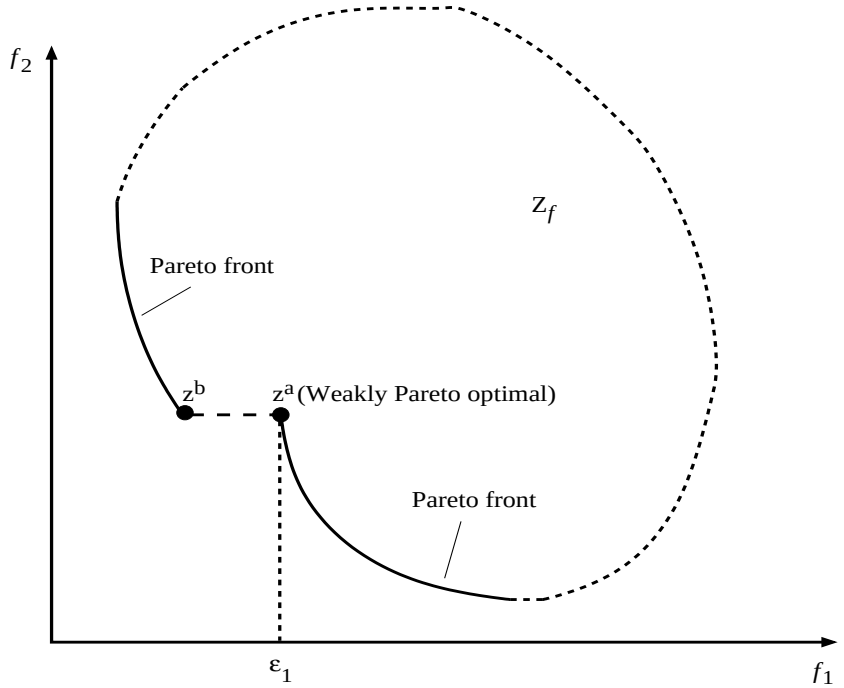

Figure 2.6: The ϵ-constraint method guarantees weak Pareto optimality.

The ϵ-constraint method may fail in finding the efficient points. However, it will guarantee at least weak Pareto optimality of the solutions (see [20, Proposition 4.1]). Figure 2.6 illustrates a case where the ϵ-constraint method fails to locate the Pareto optimal solution. The optimal set of the problem $S_2(\epsilon)$ includes the points lying on the line between $\mathbf{z}^a$ and $\mathbf{z}^b$. On this line, $\mathbf{z}^b$ is the only efficient solution. However, the optimizer may end up with $\mathbf{z}^a$ whose corresponding decision vector is weakly Pareto optimal.

The ϵ-constraint method is superior to the weighted sum method in the way that it does not end up only with the solutions located in the convex parts of the Pareto front. The following theorem from [20] summarizes the conditions necessary for the Pareto optimality.

Theorem 2.14. *The solution $\mathbf{v}^*$ is Pareto optimal iff there exists an $\epsilon^* \in \mathbb{R}^Q$ such that $\mathbf{v}^*$ is an optimal solution of $S_k(\epsilon^*)$ for all $k = 1, \ldots, Q$.*

According to Theorem 2.14 the ϵ-constraint method is able to find the whole Pareto frontier with the appropriate choices of ϵ which actually corresponds to the efficient vectors. Although the ϵ-constraint method ensures only weak Pareto optimality, in the literature there are some approximation algorithms inspired by this method, e.g., constraint-based evolutionary algorithm in [65]. The heuristic framework introduced later in this thesis also utilizes this method.

2.3.3 The Lexicographic Weighted Chebyshev Programming

The lexicographic weighted Chebyshev method was first introduced by Steuer and Choo in [79]. Further information can be found in [77] and [78]. This method is more powerful than the weighted sum method, since it is able to locate the whole Pareto frontier regardless of its shape. It guarantees the Pareto optimality in two steps which in turn names it lexicographic. The first step minimizes the distance to a reference objective vector where the distance is measured by the weighted Chebyshev metric.

The reference vector is given by

$$z_i^{ref} = \min \ \{f_i(\mathbf{v}) | \mathbf{v} \in \mathbf{X}_f\} - \epsilon_i$$

where ϵ_i are small positive values. It is common to choose values for ϵ_i which decrease z_i^{ref} to the nearest integer less than $z_i^* = \min \ \{f_i(\mathbf{v}) | \mathbf{v} \in \mathbf{X}_f\}$.

The weighted Chebyshev metric is then measured as

$$\|\mathbf{z} - \mathbf{z}^{ref}\|_\infty^\delta = \max_{i \in \{1, \cdots, Q\}} \ \delta_i |z_i - z_i^{ref}|$$

where $\delta_i \in \mathbb{R}, 0 < \delta_i < 1 \ \forall i \in \{1, \ldots, Q\}$ and $\sum_{i=1}^{Q} \delta_i = 1$.

The points with the same metric value form the surfaces of a Q-hypercuboid centered at $\mathbf{z}^{ref}$. The vertices of the hypercuboid are determined by the weight vector. The metric is minimized at the point(s) where the hypercuboid touch the border of the feasible region.

Figure 2.7 illustrates this for a bi-objective problem. Two rectangles with different metric values are shown. The points lying on the outer rectangle (shown with dotted lines) have equal metric values. However, the metric is optimized with the inner rectangle touching the border of the feasible region at $\mathbf{z}^a$ which is efficient.

In this small example, the optimal solution set consists of a single point. However, minimizing the weighted Chebyshev metric may end up with alternative solutions in the objective space. An example where the first step fails to locate an efficient vector is shown in Figure 2.8. The image of optimal solutions in the objective space includes the feasible objective vectors that lie between $\mathbf{z}^a$ and $\mathbf{z}^b$. $\mathbf{z}^b$ is the only point which is efficient in this set.

The following theorem specifies the attributes of the solution encountered at the first step of the method:

Theorem 2.15. *A feasible point $\mathbf{v}^*$ is weakly Pareto optimal iff there is a weight vector δ such that $\mathbf{v}^*$ is an optimal solution of the problem*

$$\begin{aligned} P(\delta): \quad & \min \ \alpha \\ \textit{subject to} \quad & \mathbf{v} \in \mathbf{X}_f, \\ & \alpha \geq \delta_i (f_i(\mathbf{v}) - z_i^{ref}) \qquad i = 1, \ldots, Q, \\ & \delta \in \Delta = \{\delta \in \mathbb{R}^Q | \delta_i \in (0,1), \sum_{i=1}^{Q} \delta_i = 1\}. \end{aligned}$$

The proof for the general case is given in [20, Theorem 4.12]. Here, we consider the case with scaled weights where they add up to 1.

Since a Pareto optimal solution is also weakly Pareto optimal, according to Theorem 2.15 it is among the optimal solution set of $P(\delta)$. However, solving $P(\delta)$ can only guarantee the weak Pareto optimality of the solution. So, the second step aims at locating the Pareto optimal solution among the alternative optimal solutions (in the objective space) of $P(\delta)$. Let X_δ be the set of optimal solutions of the problem $P(\delta)$.

Theorem 2.16 (Steuer and Choo, 1983, [79]). *A feasible point $\mathbf{v}^*$ is Pareto optimal, if it is an optimal solution of the problem*

$$\begin{aligned} P(\delta)': \quad & \min \ \sum_{i=1}^{Q} f_i(\mathbf{v}) \\ \textit{subject to} \quad & \mathbf{v} \in X_\delta. \end{aligned}$$

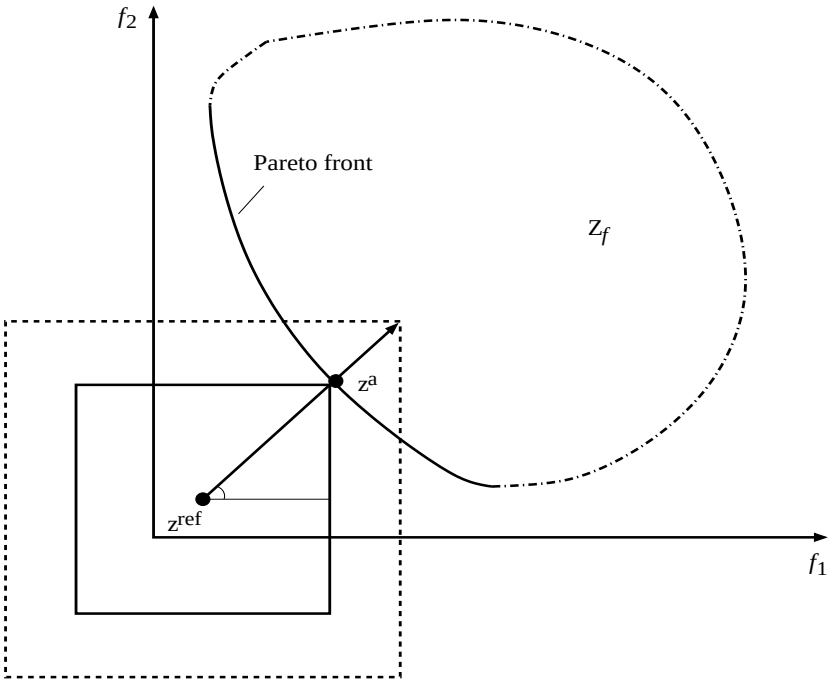

Figure 2.7: The weighted Chebyshev metric is associated with a ray extending from a reference point.

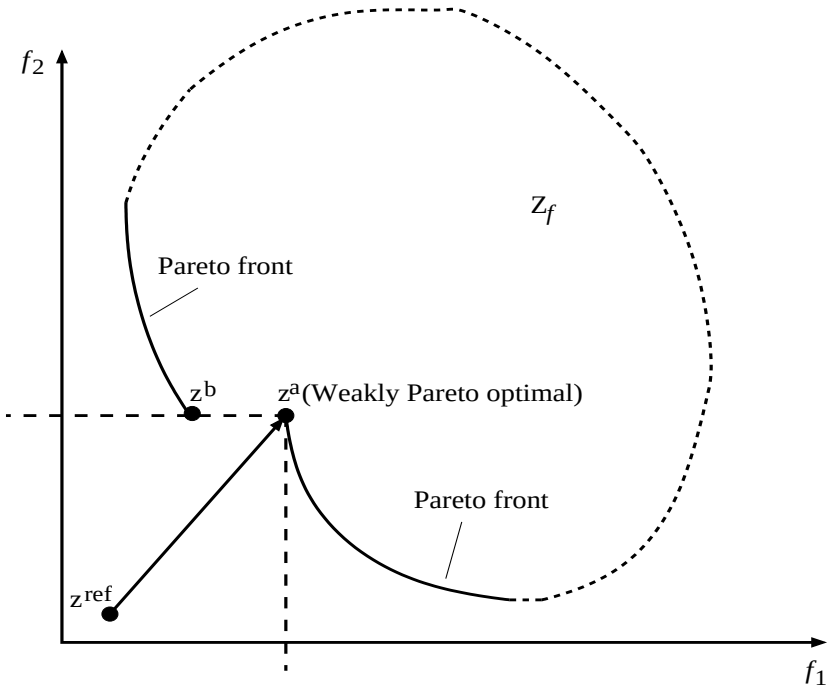

Figure 2.8: The first step of the lexicographic weighted Chebyshev programming guarantees weak Pareto optimality.

Unless the problem has a non-linear structure, two steps can be combined into one, by multiplying the objective function of $P(\delta)'$ with a proper scalar ρ [77], [79]. The reduced problem is then

$$\begin{aligned} & \min \ \ \alpha + \rho(\sum_{i=1}^{Q} f_i(\mathbf{v})) \\ \text{subject to} \quad & \mathbf{v} \in \mathbf{X}_f, \\ & \alpha \geq \delta_i(f_i(\mathbf{v}) - z_i^{ref}) \qquad i = 1, \ldots, Q, \\ & \delta \in \Delta = \{\delta \in \mathbb{R}^Q | \delta_i \in (0,1), \sum_{i=1}^{Q} \delta_i = 1\}. \end{aligned}$$

2.4 Discussion of Exact Solution Approaches

The exact solution methods have been widely used in the literature. The main drawback within the exact approaches is their requirement for several runs of the optimizer with different parameters to obtain the Pareto front. The weighted sum and the lexicographic weighted Chebyshev method work with different weights at each iteration. The same solution can be obtained even with different weight vectors. As a result, the number of SOP problems to be solved is usually larger than the number of Pareto optimal solutions obtained at the end of the optimizing process. Moreover, in order to determine the proper weights, one needs the pre-knowledge about the scale of the objective functions.

The main deficiency of the weighted sum method is its failure to obtain the non-convex parts of the Pareto frontier. In [15], Das and Dennis focus on the drawbacks of the weighted sum method. Using examples, they show that a uniform spread of efficient points often corresponds to a nonuniform distribution of the weights. Hence, a uniformly distributed set of weights may fail to produce solutions which are uniformly distributed over the Pareto frontier.

With respect to others, the ϵ-constraint method has received less attention in the literature. Its primary weakness is that the parameters requested in this method correspond to the true efficient vectors. Additionally, to guarantee Pareto optimality, it requires to solve several SOP problems.

The exact methods are not well suited for difficult MOP problems due to their inefficiencies. However, the theoretical aspects within the exact approaches may help us to develop better approximate methods.

3 An Off-line Multiobjective Model for MPLS Networks

As the use of the Internet spreads in both business and entertainment sectors, the notion of *QoS* has become more critical for network service providers. Today's standard IP (Internet Protocol) networks support only a single service level called *best-effort* service. For best-effort service, the network will try its best to forward the traffic without giving guarantees on the routing performance in terms of loss rate, bandwidth, delay, delay jitter, and so on. All packets are treated equally regardless of their source applications. In the current Internet, some of the applications have *elastic* requirements, e.g., they can tolerate packet losses and/or delays, or they can respond to the congestion by decreasing their transmission rates [74]. Remote terminal (e.g., Telnet), file transfer protocol (e.g., FTP), and electronic mail are among the examples for elastic applications.

Although best-effort service is acceptable for elastic applications, it is not tolerable for real-time and multimedia applications such as Internet telephony and videoconferencing. Real-time applications have more difficult requirements than the elastic applications. Their performance is very sensitive to packet losses, delays and delay jitters throughout the network. Moreover, they can not reduce their transmission rates in case of a congestion.

Under the hard requirements of some specific applications, the notion of QoS has been very popular in the Internet literature. In [14], QoS is defined as "a set of service requirements to be met by the network while transporting a flow", where flow implies a packet stream associated with a specific application. Alternatively, QoS can be defined as the level of service measured by the user(s) of the network. QoS requirements of a specific flow can be specified in terms of packet loss probability, bandwidth, end-to-end delay, reliability, etc. The customers of the network may agree with the service providers on the QoS requirements via *Service Level Agreements* (SLAs).

To support QoS in today's Internet, several new architecture models have been proposed [9], [29], [84]. Traffic engineering has become a key issue within these new architectures, as supporting QoS requires more sophisticated resource management tools. The goal of traffic engineering is the performance evaluation and optimization of operational networks [6]. Its functions include measurement, characterization,

modeling and control of the traffic. One of the most novel technologies introduced lately with the increasing need of convenient traffic engineering is *Multiprotocol Label Switching* (MPLS) [5], [16], [38], [68].

This chapter first introduces the basic concepts within MPLS networks. The introduction is followed with the formulation of the multiobjective traffic engineering problem. Some theoretical analyses are carried out to investigate the basic attributes of the problem. At the end of the chapter, an exact solution approach is applied in a case study.

3.1 How does MPLS work?

MPLS is a high-performance technology for transporting IP packets through a network. The basic idea within MPLS is assigning short and fixed length labels to the packets at the *ingress routers* (the routers where the packets enter the network). Throughout the network the packets are forwarded according to these labels. The labels are removed at the *egress routers* (the routers where the packets exit the network). In MPLS, it is also possible to classify the traffic by the notion of *Forwarding Equivalence Class* (FEC). We clarify the forwarding paradigm of the MPLS networks by explaining its key components:

- **Label:** A short and fixed length physical identifier which is assigned to the packets at their ingress routers according to their FECs. Within the domain of MPLS, the label switching routers (LSRs) use this label as an index to look up their forwarding tables. Thus, the neighbor router, to which the packet is sent, is determined with this label. Since the label is kept shorter than the usual IP header, less overhead for the investigation of the packets exists on the intermediate routers than in IP networks. With the introduction of labels, the routing decisions are carried out at the source routers, rather than being under the control of hop-by-hop behavior of the standard IP networks.
- **Forwarding Equivalence Class (FEC):** A group of IP packets which are to be forwarded in the same manner with the same treatment through the network. The FEC of an IP packet is determined at the ingress router, before it enters the network. This task requires the analysis of the packet header to derive the necessary information to classify the packet. The destination of the packet, its service class, routing options, policy requirements, etc. are used for packet classification. The ingress router assigns a label to the packet according to its FEC and forwards it to one of its neighbors determined by its LSP.
- **Label Switched Path (LSP):** The path which is created by the chain of one or more LSRs. An LSP is followed by the packets from a specific FEC. The LSPs are built by some label distribution protocols.

- **Label Distribution Protocol (LDP):** A specification to build the forwarding tables of the routers throughout the network. A LDP is used to reserve the network resources (bandwidth) necessary to meet the service requirements of the LSP. RSVP-TE (Resource Reservation Protocol-TE) [4] and CR-LDP (Constraint-based Routing Label Distribution Protocol) [40] are among the known label distribution protocols. These protocols build an LSP before the first packet of the specific FEC enters the network. When the request for an LSP arrives at the ingress router, the label distribution protocol supplies the communication between the corresponding routers for an agreement on the labels they use to forward the traffic. It is worth to note here that the label distribution protocols do not compute routes for the traffic.

Traffic engineering in MPLS networks is similar to those arising in Asynchronous Transfer Mode (ATM) networks. To bring guaranteed QoS (lacking in connectionless IP networks), MPLS provides connection-oriented capabilities, as in ATM networks. LSPs coincide with the circuit-switched paths in ATM networks.

Figure 3.1 shows how LSRs forward the packets to its neighbors. The routers only check the labels of the packets to look up the related information in their label forwarding tables. After they replace the current label with a new one, they forward the packet on hand to the next router in the LSP. As it is explained before, the forwarding tables are created by the use of some label distribution protocol.

The following features especially bring flexibilities to MPLS networks, while also improve its capabilities for traffic engineering:

1. The routing decisions are carried out at the edge routers of the network through manual or automated systems, whereas in traditional IP networks the routing control by the administrator is relatively limited. In standard IP networks, the traffic is forwarded through the shortest paths based on the static weights of the links. However, MPLS makes it possible to define the paths explicitly.

2. MPLS networks allow for both traffic aggregation and disaggregation. In traditional networks, since all of the traffic is treated equally, it is usually not possible to disaggregate the traffic. Exceptionally, the technique called equal-cost multipath (ECMP) allows the shortest path first routing systems in standard IP networks to integrate a limited disaggregation mechanism [60]. With this mechanism, the total traffic to a destination is distributed among the equal-cost paths. ECMP is still not flexible enough, since the traffic is usually partitioned equally only on the paths of equal costs. However, in MPLS, the concepts of FEC and LSP make it possible to aggregate the traffic flows with similar characteristics. Additionally, the whole traffic between any ingress-egress pair can also be routed along multiple LSPs (may be of different lengths) where each LSP is assigned to a partition of the aggregated traffic.

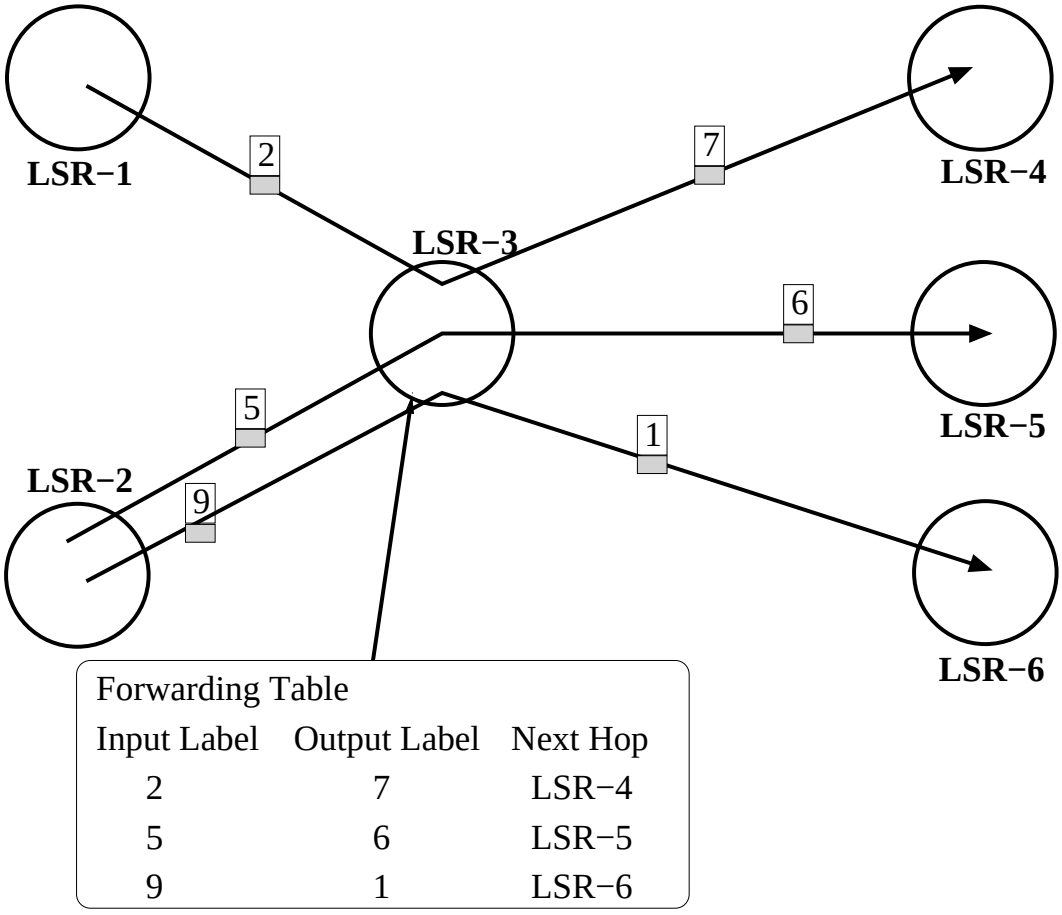

Figure 3.1: A label switching router forwards the packets according to their labels.

3.1.1 Policies and Constrained-based Routing

Today, network service providers work via SLAs to guarantee levels of service to their customers. The agreements may include some policies to impose administrative rules such as priority assignments to the traffic flows or security regulations. For example, a policy may state that the packets belonging to the banking applications of customer A can not visit a list of routers due to their security leaks.

Recent traffic engineering studies have introduced the term *constraint-based routing* which encompasses QoS routing and policy-based routing as subsets [5], [6]. Thus, constraint-based routing computes routes subject to a set of constraints and requirements. Requirements may be due to the QoS specifications of the traffic such as the bandwidth, delay, probability for packet loss, and the administrative policies. MPLS networks are advantageous for constraint routing, since they work on path basis.

3.2 Classification of Traffic Engineering

The taxonomy of traffic engineering is fully introduced in [6]. Here, we will only review some of the traffic engineering methods related to the study in this thesis.

3.2.1 Time-dependent Versus State-dependent

State-dependent traffic engineering methods base their solutions on the current state of the network. They intend to give quick responses to the changes in the network conditions. State-dependent traffic engineering systems usually require intelligent information gathering systems to measure continuously the performance parameters of the network.

On the contrary, time-dependent traffic engineering systems utilize the historical information that has been gathered over a relatively longer duration. Customer profiles, traffic measurements based on stochastic methods can be taken as an input in this methodology. Time-dependent systems can not respond quickly to the variability both in the real-time traffic and in the network conditions.

3.2.2 Off-line Versus On-line

The primary aim of off-line traffic engineering systems is to carry out extensive searches and deep analyses of the network performance. As in time-dependent traffic engineering systems, they utilize the forecast information as input.

The decisions in on-line systems are based on the current state of the network and the main idea is to increase the network performance by giving rapid responses to the changes in the network, as in state-dependent systems. However, a major drawback of on-line systems is that their solution to the problem at hand is usually sub-optimal. They may miss the global control of the network, as they intend to provide the network manager with fast solutions based on the current information at hand. Example 3.1, which is also studied in [80], explains a case where an on-line routing mechanism ends up with a relatively poor quality solution.

Example 3.1. In Figure 3.2, a simple network topology called *parking lot* is plotted. There are n ingress-egress pairs and for each of them there is a flow request of 1 unit/sec. All link capacities in the network are equal to unity. Assume that the requests arrive in the order of $(S_0, D_0), (S_1, D_1), \ldots, (S_n, D_n)$. A standard on-line routing algorithm, such as shortest path first, will send the first request throughout the single available path. Accepting the first request causes full utilization of $n + 2$ links in the network which results with the rejection of other requests. However, an optimal routing would reject the first flow and route the remaining n flow requests.

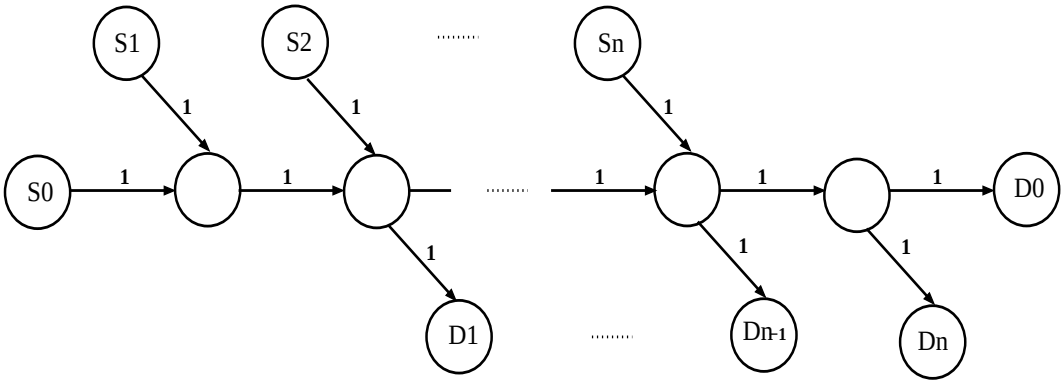

Figure 3.2: The performances of on-line and off-line traffic engineering may differ a lot.

3.3 Overview of Traffic Engineering Models

This section mainly focuses on some traffic engineering studies for MPLS networks. We categorize these studies in two main groups: on-line versus off-line. Some on-line traffic engineering studies in MPLS networks have been carried out in [22], [48], [50] and [80].

In [22], Elwalid et al. developed a multipath adaptive traffic engineering mechanism, called MATE, to avoid network congestion. Their mechanism distributes the total traffic between any two nodes across the pre-established LSPs according to the current state of the network in a way that the traffic loads on the LSPs are balanced and congestion is thus minimized. MATE consists of two phases: a monitoring phase and a load balancing phase. The monitoring invokes the load balancing phase when a change in the network state is detected. The load balancing phase first sends probe packets to the egress node to measure the packet loss and delay. Then the optimization algorithm tries to equalize the congestion measures among LSPs. Once the measures are equalized, the algorithm moves to the monitoring phase. The authors have experimented this methodology with small networks. The implementation of the algorithm is based on complex analytical methods and the performance of the algorithm is sensitive to the measurements that is obtained by the probe packets.

Kodialam and Lakshman presented a minimum interference routing algorithm (MIRA) for the on-line routing of bandwidth guaranteed traffic flows [48]. The model is based on the assumption that the traffic flow requests arrive one at a time and there is no priori knowledge regarding the future requests. The main idea in the algorithm is that the traffic flow on hand must follow the path which minimizes the "interference" with the routes that may be critical to satisfy a future demand. Their mathematical model utilizes the maximum flows that build an upperbound on the

total amount of traffic which can be sent between a given ingress-egress pair. The path with the minimal interference implies the one which maximizes the weighted sum of maximum flows over all of the ingress-egress pairs. They showed that this problem is NP-hard and provided a heuristic solution. Their algorithm works on a fine-grained level, since it is executed each time when a single flow arrives to the network. Their proposal is computationally expensive, since routing a single request requires maximum flow computations for every ingress-egress pairs. Suri et al. showed that MIRA can fail to give good results in some specific cases [80].

In [50], a mechanism was developed to distribute the traffic requests across the paths, as in [22]. In contrast to the study of Elwalid et al., Lagoa and Che used the tool *sliding modes* from nonlinear control theory in order to solve the optimization problem. For each traffic request a differentiable concave utility function is defined. The optimization model then aims at the maximization of the sum of the utility functions subject to the network resource constraints. Their implementation has the drawback, that the network resources are not shared by different QoS classes; each QoS class sees a separate logical network with dedicated resources which may not be realistic. Furthermore, their algorithm is based on the continuous control laws, although in real-life, measurements in the network has a discrete nature.

The on-line routing framework developed in [80] has a unique attribute among the other studies such that their approach has an off-line pre-processing step based on the expected bandwidth requirements between the ingress and egress pairs. In the off-line part, a linear multicommodity flow problem is solved in order to minimize the routing cost subject to the network resource constraints. According to the output of the multicommodity flow problem the bandwidth is pre-allocated on each link for each traffic. When a flow from a traffic arrives, the on-line algorithm returns a route on the reduced graph which consists of the pre-allocated capacities corresponding to this traffic.

Examples of off-line traffic engineering studies can be found in [59], [70], and [85]. In [59], Mitra and Ramakrishnan proposed a traffic engineering technique for data networks which support multi priorities according to the service classes. The optimization problem in this study aims at the maximization of the total throughput. The solution approach has a hierarchical structure where the QoS traffic (high priority) is mapped first onto the network. The initial step minimizes the total resource consumption by the QoS traffic based on the idea that decreasing the network usage by priority traffic causes an increase in the throughput of the best-effort traffic.

In [70], Schnitter and Haßlinger investigated the solution approaches to the off-line LSP-design problem which aims primarily at minimum utilization rate on the network. As a secondary goal they consider minimum resource consumption by the traffic. In this study, two cases are considered according to the traffic splitting rule. When the traffic is allowed to be split through multiple paths, the problem can be

solved by linear programming, but in case of unsplitted traffic the problem becomes NP-hard. For the second problem they proposed a heuristic approach based on simulated annealing.

Xiao et al. described in [85] a very simple algorithm whose solution is sub-optimal. In the off-line algorithm the LSPs for the traffic requests are built through the network one-by-one in the order of the decreasing bandwidth requirement.

The multiobjective off-line routing problems in telecommunications have been previously studied in [8], [44], and [67]. In [8], an optimization model is introduced to re-dimension the existing LSPs under changing traffic demands. Two objectives are taken into consideration: maximization of the total throughput and minimization of the sum of changes in the reserved bandwidth for LSPs. The objectives are aggregated into a singe objective by scalars. Bessler showed how these scalars effect the quality of the solutions. He also developed a heuristic solution approach to this problem.

Knowles et al. [44] selected the following objectives: minimization of the routing cost, minimization of the total positive deviations from the target utilization of the links, and minimization of the over-utilization of the links. A specific multiobjective evolutionary algorithm is applied to approximate the Pareto optimal solutions and its performance is investigated.

In [67], Resende and Ribeiro developed a model for permanent virtual circuit routing. The model minimizes a weighted objective function consisting of both a delay component and a load balancing component similar to the one used in this study.

The model to be introduced shortly is first studied in [26] and further developed in [25]. It is similar to the models in [44] and [67]. The main difference lies in the assumption about the traffic splitting rule. In these previous studies, traffic is not allowed to be split. On the contrary, our study considers the minimization of the splitted traffic as a third objective, which causes a dramatic change in the nature of the problem. Furthermore, this study supplies a more detailed discussion about load balancing in MPLS networks.

In general, off-line traffic engineering systems operate on an "expected" traffic demand matrix which can be either based on the SLAs or can be calculated by the stochastic methods. In [59], it is assumed that a single flow requires an effective bandwidth from the network which represents the characteristic of its burstiness and QoS metrics, such as delay and loss. Since a traffic request consists of an aggregation of multiple traffic flows of the same kind, the computation of the expected bandwidth requirements is based on the assumption that the flows arrive as a stochastic point process and live for randomly distributed time periods. As stated in [8], the traffic demand matrix can also be established according to the customer profiles, and SLA(s). Discussions about the methods for forecasting the traffic demand matrix is out of the scope of this study.

3.4 Problem Definition

For the ease of representation, we introduce the term *traffic request*.

Definition 3.2. *A* **traffic request** *is the aggregation of the traffic flows from the same class. The following attributes are necessary for a complete definition of a traffic request:*

- *ingress router,*
- *egress router,*
- *expected bandwidth demand,*
- *constraints exposed on the available paths due to the QoS requirements and policies.*

The basic problem is to select the optimal LSPs for traffic requests from different service classes in a capacitated network. The service classes include both traffic with QoS requirements and best-effort traffic. Within the QoS context, it is reasonable to put these service classes into priorities so that the traffic requests are given a relative importance. For example, when three priority levels are defined as "high", "medium" and "low", the traffic requests for voice and video data can be given the priority level "high", while traffic requests for world-wide-web and best-effort data can be assigned to "medium" and "low", respectively. The definition of the priority set and assignments can vary from network to network.

3.5 Model Formulation

For the mathematical model, the network is represented as a directed graph, where $V = \{1, 2, ..., N\}$ and $E = \{1, 2, ..., M\}$ define the set of the routers and links, respectively. The directed link m has capacity u_m (in units/sec).

The set of all traffic requests is denoted by T. The t^{th} traffic request has a bandwidth requirement d_t. The basic model introduced in this section assumes only one level priority. The extension of the model for the multiple priority case will be discussed soon. As stated in Definition 3.2, the application of constraint-based routing may expose some constraints on the LSPs. The routing performance of QoS traffic is highly dependent on the jitter, delay and reliability. As the number of hops on an LSP decreases, the traffic request experiences less jitter and delay. Moreover, using less hops increases the transmission reliability of the traffic request, since the probability of a failure on the LSP decreases. Therefore, the traffic requests from QoS classes may

have a constraint on the number of hops on their LSP(s) [59]. In order to implement this constraint, an *alternative path set* $P_t = \{p_t^1, ..., p_t^{L_t}\}$ is defined for each traffic request, where L_t denotes the number of paths in the set. The traffic requests with lower service classes may not have such a constraint.

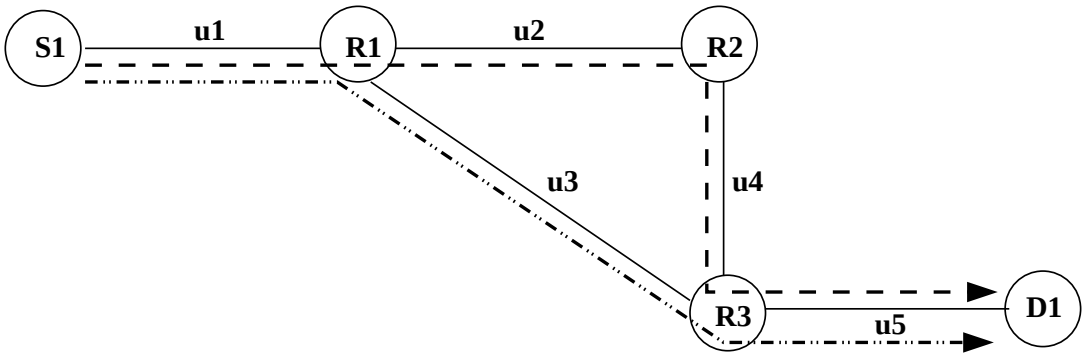

Figure 3.3: The LSPs may be constrained due to the implementation of QoS and policy-based routing.

Example 3.3. In Figure 3.3 an example topology is given for an MPLS network with one ingress router ($S1$), one egress router ($D1$) and three core routers ($R1, R2$ and $R3$). There exist two traffic requests from $S1$ to $D1$, one for QoS traffic and one for best-effort traffic. Two distinct paths between $S1$ and $D1$ are possible: $p^1 = (S1, R1, R2, R3, D1)$ and $p^2 = (S1, R2, R3, D1)$. QoS traffic has a requirement that it can be only assigned to paths with at most four hops on the path. Thus, the admissible path sets for the traffic requests are as follows:

$P_{BE} = \{p^1, p^2\}$,

$P_{QoS} = \{p^2\}$.

When traffic requests, like best-effort traffic, do not have a constraint with regard to the number of hops on their paths, they have an alternative path set that consists of all the paths between their ingress and egress router. However, as the size of the network increases, the number of possible paths between any two nodes grows exponentially. In that case, the number of available paths should be limited. An approach to give a bound on the number of available paths is to consider only k-shortest paths for the traffic requests which do not have any constraints on their LSPs. The solution of the k-shortest path problem returns the $1^{st}, \ldots, k^{th}$ loopless shortest paths between any two nodes according to the given link costs. In [53] an algorithm is given for this problem. The algorithm requires $O(k|V|)$ shortest path calculations, each of which has computational complexity of $O(|V|^2)$.

Three objectives are taken into consideration for the multiobjective traffic engineering problem for the MPLS networks. The rest of this section explains these objective functions.

Minimizing the total routing cost

The first objective in the model aims at minimizing the routing costs which is experienced by the traffic requests. Link m is assigned a value c_m to represent the routing cost on that link. The cost of the link may depend on some parameters, namely the speed, length, and reliability of the link. The first objective of our model is traffic-oriented rather than network-oriented. We now introduce indicator variables $a^l_{t,m}$, which is equal to 1 if p^l_t uses link m, and 0 otherwise. The cost of p^l_t is denoted by C^l_t and is equal to the sum of its links' costs:

$$C^l_t = \sum_{m \in E} c_m a^l_{t,m}. \tag{3.1}$$

Let x^l_t represent the amount of traffic that is routed on p^l_t.

$$\sum_{l=1}^{L_t} x^l_t = d_t \qquad \forall\ t \in T, \tag{3.2}$$

$$x^l_t \geq 0 \qquad \forall\ t \in T \text{ and } \forall\ l \in \{1, \ldots, L_t\}. \tag{3.3}$$

Constraints (3.2) ensure that the sum of the routed traffic meets the demand for each traffic request.

The first objective function minimizes the sum of the routing costs:

$$\min \sum_{t \in T} \sum_{l=1}^{L_t} C^l_t x^l_t. \tag{3.4}$$

A special case occurs when all of the links have a routing cost of unity in the network. In this case, the routing cost of a traffic request can be expressed by the sum of the multiplication of the bandwidth allocated on its paths with the number of links on the paths (bandwidth * links). Additionally, the sum of the routing costs under unity can also be interpreted as a measure for the total resource consumption by the traffic requests, and as the total flow in the network.

Balancing the load

The second objective aims at avoiding high utilization of some links while leaving others less utilized. The utilization rate of a link is measured by the proportion of the

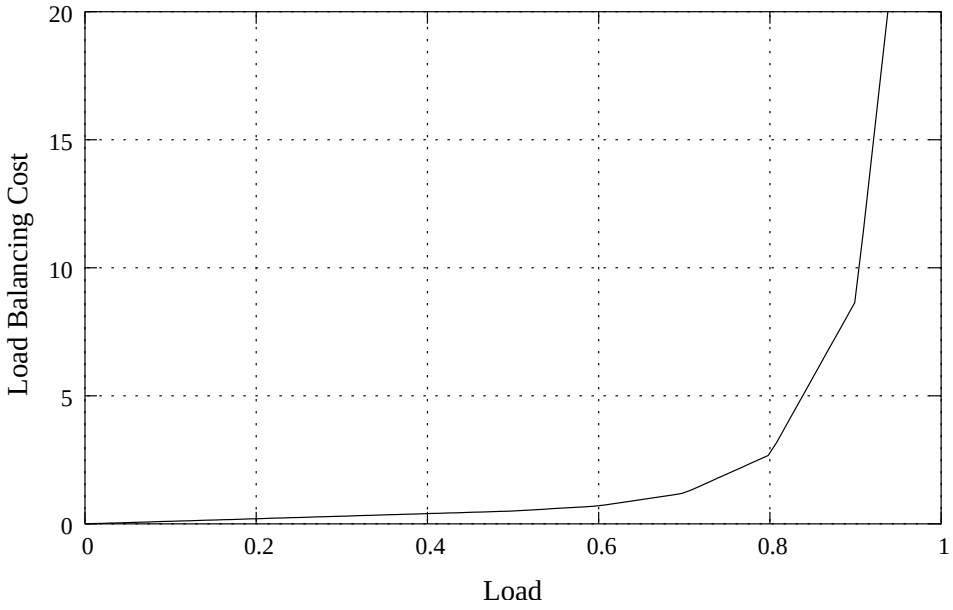

Figure 3.4: The load balancing cost function is piece-wise linear increasing and convex.

total traffic load on it to its capacity. Minimizing the maximum link utilization in the network is the most widely used objective function for balancing the load. However, a modified version of the function that was proposed in [27] is suggested in our model. In their study, a piece-wise linear cost function is defined for each link based on the link utilization rate. The aim is to minimize the sum of the links' costs. The idea behind the function is to penalize sending packets over a link as its utilization gets higher.

The load balancing function used in this study differs from the original function in the way that the over-utilization of the links is not allowed and it has different break points. In fact the exact shape of the function is not critical; the more important thing is that it is a piece-wise linear increasing and convex function [27]. The complete definition of the function depends highly on the network and traffic demand. The impact of the shape of the function on load balancing will be illustrated with examples. The load balancing cost function for a link of capacity of 1 unit/sec is given in Figure 3.4.

The load balancing cost depends on the total load carried on link m, which is denoted

by g_m. Constraints (3.6) imply that the links can not be over-utilized.

$$g_m = \sum_{t \in T} \sum_{l=1}^{L_t} a_{t,l}^m x_t^l \qquad \forall\, m \in E, \tag{3.5}$$

$$g_m \leq u_m \qquad \forall\, m \in E. \tag{3.6}$$

For link m with capacity u_m, the link utilization rate is equal to

$$\lambda_m = \tfrac{g_m}{u_m}.$$

The breaking points of the function are at the following utilization rates of the link: $0.5, 0.6, 0.7, 0.8$ and 0.9. An auxiliary variable ϕ_m is introduced to determine the value of the function for link m by giving upper bounds. The following constraints are used to determine the value of the load balance cost function:

$$\phi_m \geq g_m \qquad \forall\, m \in E, \tag{3.7}$$

$$\phi_m \geq 2g_m - \frac{1}{2}u_m \qquad \forall\, m \in E, \tag{3.8}$$

$$\phi_m \geq 5g_m - \frac{23}{10}u_m \qquad \forall\, m \in E, \tag{3.9}$$

$$\phi_m \geq 15g_m - \frac{93}{10}u_m \qquad \forall\, m \in E, \tag{3.10}$$

$$\phi_m \geq 60g_m - \frac{453}{10}u_m \qquad \forall\, m \in E, \tag{3.11}$$

$$\phi_m \geq 300g_m - \frac{2613}{10}u_m \qquad \forall\, m \in E. \tag{3.12}$$

Our second objective minimizes the sum of the load balancing costs,

$$\min \sum_{m \in E} \phi_m. \tag{3.13}$$

In the studies in [67] and [27], it is observed that using the piece-wise linear increasing cost function implies a decrease in the maximum link utilization rate in the network. Apart from this parallel relationship between these two functions, this study also considers the differences in their effects on the network performance. Using the balancing function can result in an LSP assignment which distributes the load in the network in a different manner from the function which minimizes the maximum link utilization in the network. This reasoning is illustrated in Figure 3.5. In this network there exists 4.0 routers and 5.0 unidirectional links (e_1,..., e_5) which are indicated by directed arcs. There are two traffic requests, one has a bandwidth demand of 10

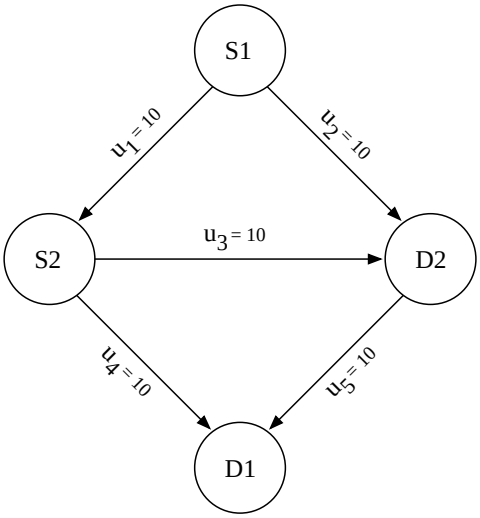

Figure 3.5: An example is given to show the performance differences between minimal sum of the piece-wise linear increasing costs and minimal maximum link utilization rate in the network.

units/sec from S1 to D1 and the other one has a demand of 10 units/sec from S2 to D2. The first traffic request has to be sent through link 3, since it is the only available path to its destination. So, the maximum utilization in the network is forced to be 1.0. Minimizing the maximum utilization as an objective would not care the rest of the network, and transmitting all of the demand of the second request through e_1 and e_4 may be its proposed solution. The solution obtained by minimizing the sum of the specified cost functions proposes a more balanced distribution of the traffic between the paths e_1 following e_4 and e_2 following e_5. In the solutions which are obtained by using this function, the utilization of the corresponding links will be 0.5. Hence, the high utilization of some links will be avoided. The total load balancing function values for various traffic distributions between two paths are given in Table 3.1. This type of bottleneck link problems may occur more frequently in the networks where some traffic requests have few paths in their alternative path set.

Balancing the load of high priority traffic on the network has the advantage that it decreases its impact on lower priority traffic. When the network is running, traffic from higher priority level is given precedence at the router interfaces. When the rate of high priority traffic on the link increases lower priority traffic may suffer from high waiting times in the queue. So, it is favorable to distribute QoS traffic on the network.

Table 3.1: As the traffic distribution becomes more unfair, the increment in the total load balancing costs also increases.

Load on Path-1	Load on Path-2	Load Bal. Cost
10.0	0.0	774
9.0	1.0	176
8.0	2.0	58
7.0	3.0	30
4.0	6.0	22
5.0	5.0	20

On the shape of the load balancing function

As stated in the previous section, the exact shape of the load balancing function depends on the traffic demand matrix. The sensitivity of the function to unfair traffic distribution changes with its exact shape. It is obvious that as the number of break points in the load balancing functions increases, the network becomes more sensitive to the unbalanced load on the links. However, introducing more break points results in additional constraints in the problem. The latter complicates the problem as it becomes larger in size. Now we will discuss how the shape of the load balancing function relates to the network performance. For the sake of the discussion, a very simple network in Figure 3.6 is used. The network is assumed to have two routers (one source router and one destination router) and two similar links between these with a capacity 10 units/sec. There is a bandwidth demand of 10 units/sec from router S to D.

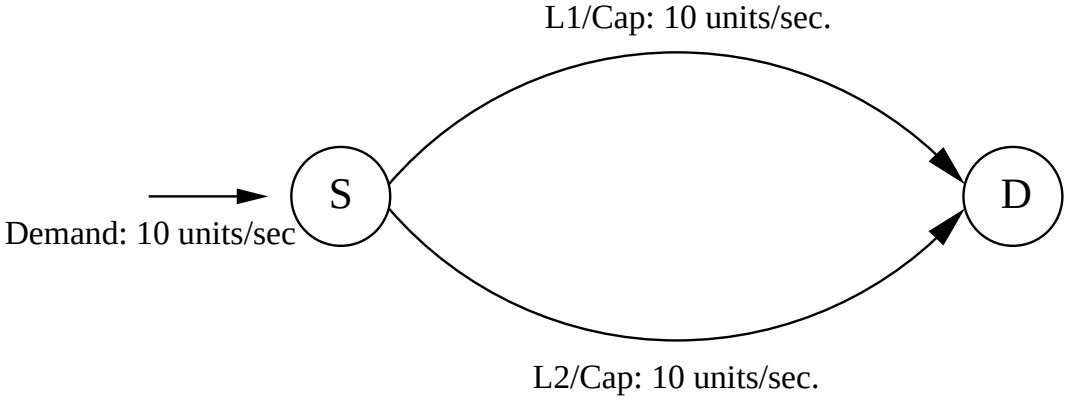

Figure 3.6: The sensitivity of the load balancing cost function to the unfair traffic distribution is explained by an example.

Table 3.2: Two load balancing functions are compared for the same problem over some possible solutions.

Load dist.	Cost for Φ^a	Rate to optimal (Φ^a)	Cost for Φ^b	Rate to optimal (Φ^b)
10.0 - 0.0	387	38.7	389	27.8
9.0 - 1.0	88	8.8	90	6.4
8.0 - 2.0	29	2.9	31	2.2
7.0 - 3.0	15	1.5	17	1.2
6.0 - 4.0	11	1.1	14	1.0
5.0 - 5.0	10	1.0	14	1.0

For each routing problem, a target utilization rate can be defined. In our example, the optimal solution is the equal distribution of the total demand between the paths. The target utilization rate for this network is 0.5. To show the effect of the exact shape of the function on the network performance, we will compare two different functions. The first function (Φ^a) is the one from previous section. The second function (Φ^b) has similarly 5 break points at the utilization rates $0.3, 0.6, 0.7, 0.8$ and 0.9. The derivatives of the function are kept same for the corresponding regions. Φ^b is defined by the following inequalities:

$$\begin{aligned}
\phi_m &\geq g_m && \forall\, m \in E,\\
\phi_m &\geq 2g_m - \frac{3}{10}u_m && \forall\, m \in E,\\
\phi_m &\geq 5g_m - \frac{21}{10}u_m && \forall\, m \in E,\\
\phi_m &\geq 15g_m - \frac{91}{10}u_m && \forall\, m \in E,\\
\phi_m &\geq 60g_m - \frac{451}{10}u_m && \forall\, m \in E,\\
\phi_m &\geq 300g_m - \frac{2611}{10}u_m && \forall\, m \in E.
\end{aligned}$$

Rate to optimal values in Table 3.2 show how much the load balancing functions penalize the points with respect to their distance from the optimal solution. As one deviates from the optimum, the penalty increases. When two functions are compared in terms of their rates to optimal values, it is observed that the penalties of Φ^a are usually much more effective for the unbalanced distributions. Furthermore, with Φ^b there exist several solutions which have minimal load balancing cost, even though they do not correspond to the optimal load distribution.

Another reasoning to support the first cost function can be explained as follows. When one moves from the load distribution of $6.0 - 4.0$ to $7.0 - 3.0$, the penalty due

to the increase in the load on one link (from 6.0 to 7.0) is weakened by the break point at the utilization rate of 0.3. The derivative of Φ^b increases as one moves from the load assignment of 3.0 to 4.0. With this observation, it is suggested that the first break point is assigned to the target utilization rate.

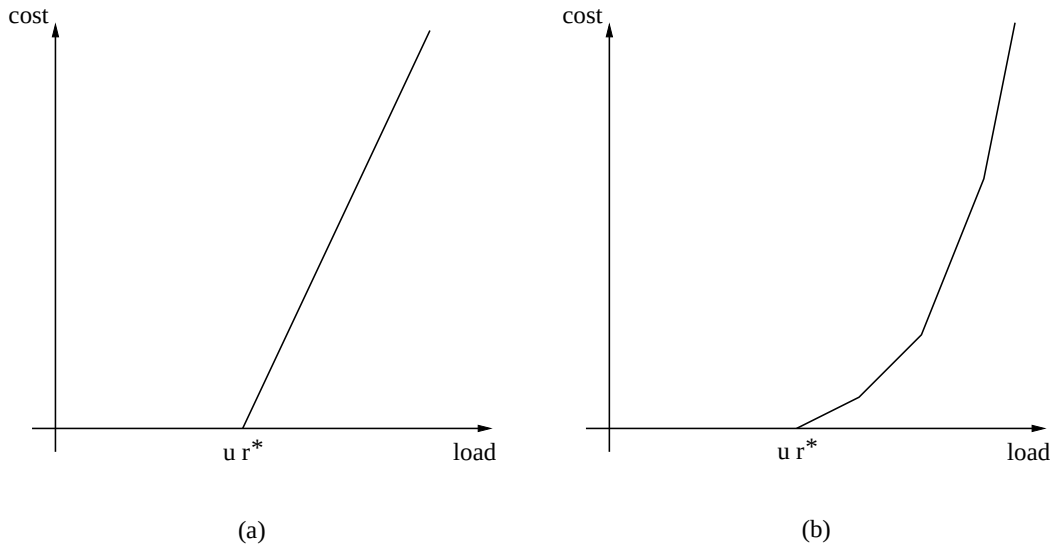

Figure 3.7: Alternative load balancing functions are possible.

Figure 3.7 illustrates alternative load balancing functions. The function in Figure 3.7(a) is used in [44]. This function has zero values for the load assignments less than the target r^*. It assigns positive costs proportional to the deviation from the target. Figure 3.7(b) shows the piece-wise linear increasing version with some break points. These functions may be more sensitive to load balancing, however they have a more non-smooth shape compared to the ones introduced previously. The non-smoothness results in an increase in the complexity of the problem. Since exact solution methods are employed in this study, smoother functions are used for load balancing.

Minimizing the number of LSPs

Our third objective is related to the number of LSPs used by the traffic requests. The more traffic requests are split over the network, the more LSPs will be established and the more complex the network management will be. Splitting the traffic requests over multiple paths will bring more messaging and labeling overhead. When the traffic flow is sent through multiple paths, the packets may experience more variant delay from each other and need to be reordered. Thus, the model aims at minimizing the number of LSPs assigned to the traffic requests.

For the introduction of the third objective into the model, we introduce decision variables y_t^l, which are equal to 1, if p_t^l is utilized, and 0 otherwise. For every candidate path in the alternative set of each traffic request, the following constraint is added to the model to settle the value of y_t^l:

$$x_t^l \leq d_t y_t^l \qquad \forall\, t \in T \text{ and } \forall\, l \in \{1, \ldots, L_t\}, \tag{3.14}$$

$$y_t^l \in \{0, 1\} \qquad \forall\, t \in T \text{ and } \forall\, l \in \{1, \ldots, L_t\}. \tag{3.15}$$

Hence, our third objective function has the following form:

$$\min \sum_{t \in T} \sum_{l=1}^{L_t} y_t^l. \tag{3.16}$$

The *multiobjective zero-one mixed integer programming problem* is summarized as in Pareto (vector) optimization context:

$$\begin{aligned} \Gamma: \quad & \min \left(\sum_{t \in T} \sum_{l=1}^{L_t} C_t^l x_t^l, \sum_{m \in E} \phi_m, \sum_{t \in T} \sum_{l=1}^{L_t} y_t^l \right) \\ \text{subject to} \quad & (3.2) - (3.3),\ (3.5) - (3.12),\ (3.14) - (3.15). \end{aligned}$$

3.6 Extensions of the Model

The basic model does not take into account the multiple priority case and admission control. The model can be extended to cover these issues.

3.6.1 Multiple Priority Case

There exist two basic approaches which handle the distribution of traffic from various priority levels within a network. The first one is based on differentiating the links' cost for each priority level, while all the traffic requests are mapped onto the network at once. When the links are more expensive for traffic with higher priority, the model attempts to assign the shorter paths to traffic with higher priority.

The other approach is to solve the model a series of times, each time for one priority level. The model is first solved for the traffic requests with highest priority level. Then it is executed for lower priority levels on the graph with reduced resources which are utilized by higher priority levels. The second approach has the following advantages. First of all, the visualization of the relationship and the trade-off between the objective functions become more apparent for each priority level. It is possible to apply

different strategies for LSP assignments of each priority level. This approach also divides the zero-one mixed integer traffic engineering problem into small subproblems. Instead of solving a large problem with more traffic requests, a series of subproblems are solved. However, this approach has the disadvantage that prior placement of higher priority traffic may affect the performance of the lower priority traffic. Consequently, this approach may end with a suboptimal solution.

3.6.2 Admission Control

The basic model is based on the assumption of a feasible traffic demand matrix for the network. In case of infeasibility, the network administrator should either overprovision the link capacities or apply an admission control mechanism to the current demand. Since network capacity management is within a totally different context, only an admission control mechanism is proposed here.

The expansion of the model for an admission control mechanism requires additional decision variables. If h_t represents the demand rejected from the t^{th} traffic request, then the following linear program leads to a feasible traffic demand matrix for the network.

$$\begin{aligned} & & \min \ \sum_{t \in T} h_t & \\ \text{subject to} \quad & \sum_{l=1}^{L_t} x_t^l + h_t = d_t & & \forall\ t \in T, \\ & g_m = \sum_{t \in T} \sum_{l=1}^{L_t} a_{t,l}^m x_t^l & & \forall\ m \in E, \\ & g_m \leq u_m & & \forall\ m \in E \\ & x_t^l \geq 0 & & \forall\ t \in T \text{ and } \forall\ l \in \{1, \ldots, L_t\}. \end{aligned}$$

The traffic demand matrix can be updated accordingly such that $d_t^{'} = d_t - h_t$ for all $t \in T$ where $d_t^{'}$ denotes the corrected demand. Although the rejection of the excess demand makes the problem feasible, it fails to satisfy all customers' demands.

3.7 Multiobjective Analysis of the Model by Exact Methods

One of the most interesting attributes of the model is that while two of its objective functions are continuous, the third objective is a discrete function. The counting

property of the third objective function allows for a nice decomposition of the model. It is removed from the original problem and is replaced as a constraint into the original problem. Hence, we substitute the original problem with a series of the decomposed multiobjective problems:

$$\begin{aligned} \Gamma(N): \quad & \min \ (\sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^l, \sum_{m\in E}\phi_m) && (3.17)\\ \text{subject to} \quad & (3.2)-(3.3),\ (3.5)-(3.12),\ (3.14)-(3.15), \\ & \sum_{t\in T}\sum_{l=1}^{L_t} y_t^l \le N, && (3.18) \end{aligned}$$

where N is the bound on the used paths. The following theorem enlightens the relationship between weakly Pareto optimal set of the original problem, stated as Γ, and the Pareto optimal set of $\Gamma(N)$. Note that any solution of the problem can be represented by $\mathbf{v} = (x_1^1, \ldots, x_{|T|}^{L_{|T|}}, g_1, \ldots, g_{|E|}, \phi_1, \ldots, \phi_{|E|}, y_1^1, \ldots, y_{|T|}^{L_{|T|}})$.

Theorem 3.4. *A solution $\mathbf{v}^*$ is a weakly Pareto optimal solution of Γ, if $\mathbf{v}^*$ is a Pareto optimal solution of $\Gamma(N)$ where $N = \sum_{t\in T}\sum_{l=1}^{L_t} y_t^{*l}$. If $\mathbf{v}^*$ is a weakly Pareto optimal solution but not a Pareto optimal solution of Γ, then there exists another solution $\mathbf{v}'$ in the Pareto optimal set of $\Gamma(n)$ $(n < N)$ which has the same routing and load balancing costs as $\mathbf{v}^*$.*

Proof. Assume that $\mathbf{v}^*$ is a Pareto optimal solution of $\Gamma(N)$, but not weakly Pareto optimal for Γ. Since it is a Pareto optimal solution of $\Gamma(N)$, then it is among the optimal solutions of the following problem (see Theorem 2.14).

$$\begin{aligned} \mu: \quad & \min(\sum_{m\in E}\phi_m) \\ \text{subject to} \quad & \sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^l \le \sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^{*l}, \\ & (3.2)-(3.3),\ (3.5)-(3.12),\ (3.14)-(3.15), \\ & \sum_{t\in T}\sum_{l=1}^{L_t} y_t^l \le N. \end{aligned}$$

If the solution $\mathbf{v}^*$ is not weakly Pareto optimal solution of Γ, then there exists a feasible solution $\mathbf{v}'$ with the following attributes:

$$\begin{aligned}
\sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^{'l} &< \sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^{*l}, \\
\sum_{m\in E} \phi_m^{'} &< \sum_{m\in E} \phi_m^{*}, \\
\sum_{t\in T}\sum_{l=1}^{L_t} y_t^{'l} &\leq N-1 < N.
\end{aligned}$$

However, these attributes contradict the fact that $\mathbf{v}^*$ is an optimal solution of μ.

The proof for the second part of the theorem is given shortly. If $\mathbf{v}^*$ is a weakly Pareto optimal solution but not a Pareto optimal solution of Γ, then three alternatives are to be investigated:

- There exists another solution $\mathbf{v}^{'}$ with the same number of used paths and routing cost (load balancing cost), but with strictly less load balancing cost (routing cost). However, this alternative is not possible, since it contradicts the fact that $\mathbf{v}^*$ is a Pareto optimal solution of $\Gamma(N)$.
- There exists another solution $\mathbf{v}^{'}$ with the same routing cost, but has strictly less load balancing cost and uses strictly less paths. This alternative contradicts the optimality of μ. A similar argument can be given for the possibility of another solution which has the same load balancing cost, but has strictly less routing cost and uses strictly less paths.
- There exists another solution $\mathbf{v}^{'}$ which has the same routing and load balancing costs but uses strictly less paths. This alternative is possible, and it does not contradict the optimality of μ. Both solutions are Pareto optimal for their corresponding subproblem.

□

It is interesting to note that for $\Gamma(N)$ we may end up with solutions which may use less than N paths. From this point on, without loss of generality we will assume that the trade-off curve of $\Gamma(N)$ will only include solutions which utilize exactly N paths.

Definition 3.5. *A trade-off curve A* **is not comparable** *to a trade-off curve B iff there are no $\mathbf{f}(\mathbf{v}) \in A$ and $\mathbf{f}(\mathbf{w}) \in B$ such that $\mathbf{v} \prec \mathbf{w}$ and $\mathbf{w} \prec \mathbf{v}$.*

Definition 3.6. *A trade-off curve A* **completely dominates** *a trade-off curve B iff for all $\mathbf{f}(\mathbf{w}) \in B$ there exists at least one point $\mathbf{f}(\mathbf{v}) \in A$ such that $\mathbf{v} \preceq \mathbf{w}$.*

Definition 3.7. *A trade-off curve A* **partially dominates** *a trade-off curve B iff there exist at least one* $\mathbf{f}(\mathbf{v}) \in A$ *and* $\mathbf{f}(\mathbf{w}) \in B$ *such that* $\mathbf{v} \preceq \mathbf{w}$*, and there are no* $\mathbf{f}(\mathbf{v}) \in A$ *and* $\mathbf{f}(\mathbf{w}) \in B$ *such that* $\mathbf{w} \prec \mathbf{v}$*.*

Corollary 3.8. *One of the following relationships is true between the trade-off curve of* $\Gamma(N)$ *and the trade-off curve of* $\Gamma(N-1)$*:*

- *the trade-off curve of* $\Gamma(N)$ *completely dominates the trade-off curve of* $\Gamma(N-1)$*,*
- *the trade-off curve of* $\Gamma(N)$ *partially dominates the trade-off curve of* $\Gamma(N-1)$*,*
- *both trade-off curves are incomparable to each other.*

The result of Corollary 3.8 is depicted in Figure 3.8, 3.9 and 3.10. As the constraint regarding the number of utilized paths gets tighter, the performances of the trade-off curves in terms of the total routing cost and load balancing cost either decrease or stay incomparable. Figure 3.8 illustrates the former case. It is also possible that two consecutive trade-off curves intersect. These intersection points occur when there exist multiple solutions which correspond to the same routing and load balancing cost values, but differ in terms of the number of utilized paths. Figure 3.9 shows the case when two consecutive trade-off curves are incomparable to each other. This may happen when the consecutive trade-off curves do not have a common value in any of the objective functions. Figure 3.10 gives an example where partial dominance occurs, in which case the trade-curves perform better for a partial range of the objective functions when the constraint relaxes.

The main interest of this chapter is to apply one of the exact methods to the multiobjective zero-one mixed integer traffic engineering problem. As already discussed in Chapter 2, the weighted sum method can not generate the whole Pareto frontier. To have a better representation of the Pareto frontier, the lexicographic weighted Chebyshev method, which is explained in Chapter 2, is applied to the problem $\Gamma(N)$. For the first step of the method, the main problem is solved with a specified weight vector $\delta = (\delta_1, \delta_2)$ to locate the points, which are at least weakly Pareto optimal.

$$Ch(N,\delta): \quad \min \alpha \tag{3.19}$$

$$\text{subject to} \quad \alpha \geq \delta_1 \Big(\sum_{t \in T} \sum_{l=1}^{L_t} C_t^l x_t^l - z_1^{ref}\Big), \tag{3.20}$$

$$\alpha \geq \delta_2 \Big(\sum_{m \in E} \phi_m - z_2^{ref}\Big), \tag{3.21}$$

$$\delta_1, \delta_2 \in \Delta, \tag{3.22}$$

$$\Delta = \{\delta \in \mathbb{R}_+^2 \mid \sum_{i=1}^{2} \delta_i = 1.0\}, \tag{3.23}$$

$$(3.2) - (3.3),\ (3.5) - (3.12),\ (3.14) - (3.15), (3.18).$$

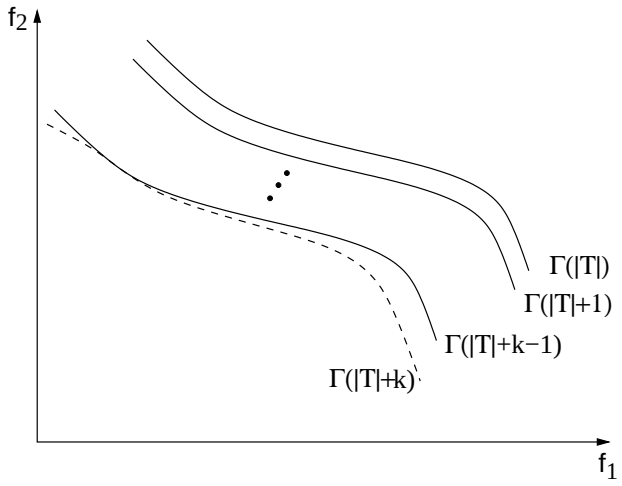

Figure 3.8: The trade-off curves may perform better as the constraint on the number of utilized paths relaxes. Notice that $\Gamma(T+k)$ completely dominates all other curves.

$z_i^{ref} = z_i^* - \epsilon_i$ for $i = 1, 2$ where

$$z_1^* = \min \sum_{t \in T} \sum_{l=1}^{L_t} C_t^l x_t^l$$

$$\text{subject to} \qquad (3.2) - (3.3),\ (3.5) - (3.12),$$

and,

$$z_2^* = \min \sum_{m \in E} \phi_m$$

$$\text{subject to} \qquad (3.2) - (3.3),\ (3.5) - (3.12).$$

The selection of the reference points is based on the global optimal values of the objective functions. They are kept same for each decomposed problem. ϵ_1 and ϵ_2 are selected such that they decrease the reference points to the nearest integer values.

In order to find the Pareto optimal solutions among the set which is known as weakly Pareto optimal, the problem, $Ch'(N, \delta)$, is solved. Here, α^* corresponds to the opti-

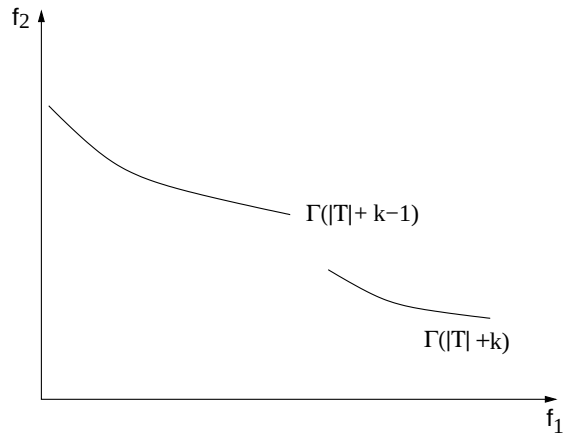

Figure 3.9: The trade-off curves may be incomparable to the previous ones as the constraint on the number of utilized paths relaxes.

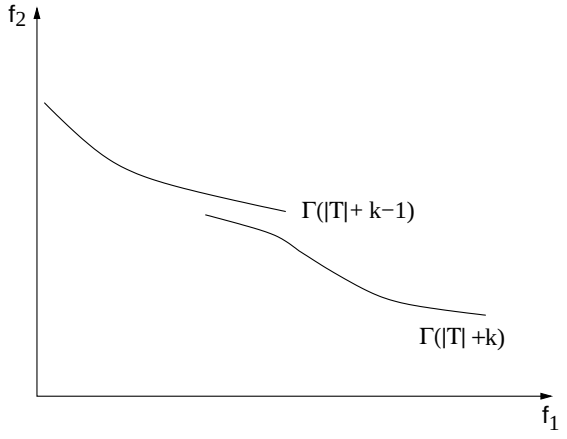

Figure 3.10: The trade-off curves may partially dominate the previous ones as the constraint on the number of utilized paths relaxes.

mal value of the problem $Ch(N, \delta)$.

$$
\begin{aligned}
Ch'(N,\delta): \quad & \min \sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^l + \sum_{m\in E}\phi_m \\
\text{subject to} \quad & \alpha = \alpha^* \\
& (3.2)-(3.3),\ (3.5)-(3.12),\ (3.14)-(3.15), \\
& (3.18),\ (3.20)-(3.23).
\end{aligned}
$$

It is observed that $Ch(N, \delta)$ has a non-linear objective function which further complicates the problem. The original problem Γ already has its non-smooth behavior due to the use of a piecewise increasing objective function. The application of weighted Chebyshev norm makes the problem even harder.

3.8 Complexity Analysis

For the complexity analysis, the decomposed model, $\Gamma(N)$, is specialized to a single objective problem and called the *number of paths restrained minimum cost multicommodity flow problem* . If the single objective version of a problem is NP-complete, we can conclude that the same problem with multiple objectives is also NP-complete.

Let $G = (V, E)$ be a connected digraph. Suppose that the ith *commodity* for the network is given as $\psi_i = (s_i, t_i, d_i)$, $0 < i \leq n$; $s_i, t_i \in V$, where s_i, t_i, and d_i represent the associated *source node*, *target node* and *flow demand*, respectively. P_i denotes the set of all possible paths for the i^{th} commodity.

$P_i = \{p : p \text{ is a path from } s_i \text{ to } t_i\}$

Let x_p be the flow on the path p. The capacities of the edges are shown by the vector $\mathbf{u}$. The flows assigned to the edges depend on $\mathbf{x}$ and are represented by a vector $\mathbf{g}(\mathbf{x})$. Here, $\mathbf{g}$ is a vector valued function for load mapping. $\mathbf{c}(\mathbf{g}(\mathbf{x}))$ is a vector valued cost function of linear or piece-wise linear increasing convex functions depending on the total load on the edges. The cost of a path, $C_p(\mathbf{g}(\mathbf{x}))$ is simply the sum of its edges' costs.

$$C_p(\mathbf{g}(\mathbf{x})) = \sum_{e \in p} c_e(\mathbf{g}(\mathbf{x})) \tag{3.24}$$

Without losing the generality, we assume that all input parameters of the problem are integer.

The *number of paths restrained minimum cost multicommodity flow problem* looks for a flow assignment satisfying the following constraints:

$$\mid S(P_1) \mid + \ldots + \mid S(P_n) \mid \ \leq N \tag{3.25}$$

where $S(P_i) \subseteq P_i$ and $S(P_i) \neq \emptyset$.

$$\sum_{p \in S(P_i)} x_p = d_i, \tag{3.26}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{u}, \tag{3.27}$$

$$\sum_{p \in S(P_1) \cup \ldots \cup S(P_n)} C_p(\mathbf{g}(\mathbf{x}), \mathbf{u}) \leq B \tag{3.28}$$

where B is the upper bound on the total cost.

Before we analyze the complexity of this problem, we introduce another problem called the *unsplittable minimum cost multicommodity flow problem*. The unsplittable minimum cost multicommodity flow problem minimizes the total routing cost according to link costs where the commodities are not allowed to be split. Although this problem is stated as an NP-complete problem in many studies, no formal proof can be found in the literature by the author[1].

[1]The general idea for the transformation is given in the lecture notes of A. Schrijver: "A Course in Combinatorial Optimization", CWI, The Netherlands. However, no formal proof is carried out.

Theorem 3.9. *The unsplittable minimum cost multicommodity flow problem is NP-complete.*

Proof. The proof is based on the polynomial time transformation from the edge-disjoint paths problem to unsplittable minimum cost multicommodity flow problem. The edge-disjoint paths problem is a known NP-complete problem [83]:

Given a graph G and a collection of $T = \{(s_1, t_1), \cdots, (s_n, t_n)\}$ of pairs of sources and destinations in G (commodities), the edge-disjoint paths problem tries to answer whether the pairs in T can be connected by the edge-disjoint paths.

Given an instance of the edge-disjoint paths problem, we will transform it to an instance of the unsplittable minimum cost multicommodity flow problem in polynomial time. We start transformation by adding an artificial edge for each commodity, directing from the source to the destination. These edges have a capacity of unity. They are assigned a cost scalar $c > |E|$. The original edges in G are assigned to a cost factor and a capacity of unity. Each commodity is assumed to have a demand value of 1 unit. An example for the transformation is given in Figure 3.11.

The cost of using an edge is the product of the total load on the edge with its cost scalar. The cost of a path is the sum of its edges' costs. We need to show that the edge-disjoint paths problem is solvable if and only if the unsplittable minimum cost multicommodity flow problem has a total cost less than or equal to $|E|$.

"$\Leftarrow$": In the solution of the unsplittable minimum cost multicommodity flow problem with a total cost value less than or equal to $|E|$, it is for certain that no commodity uses the artificial edges. Otherwise, the total cost will be larger than $|E|$. Since the capacity of each edge and the demand of each commodity are equal to unity, no commodity in the solution can use the same edge. Thus, the paths they follow are disjoint.

"$\Rightarrow$": If in the original graph the commodities can be sent through the disjoint paths, the unsplittable minimum cost multicommodity flow problem has a solution, where the commodities use only the original edges. The total cost is certainly less than or equal to $|E|$, since in the worst case all of the edges are used. □

Theorem 3.10. *The number of paths restrained minimum cost multicommodity flow problem is NP-complete.*

Proof. The proof is by restriction [28]. Restriction is based on showing that a special instance of the problem at hand is equal to one problem known as NP-complete in the literature. The number of paths restrained minimum cost multicommodity flow problem can be restricted to the unsplittable version by allowing only instances having $N = n$. By restricting the bound on the number of paths to the number of commodities to be transmitted on the network, we obtain the unsplittable minimum cost multicommodity flow problem which is known to be NP-complete. □

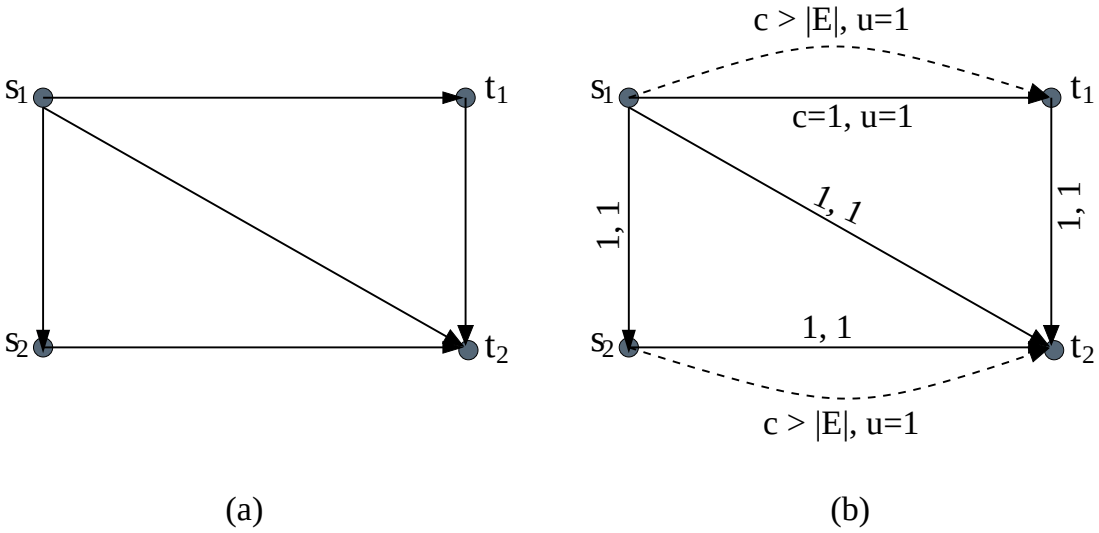

Figure 3.11: a) Original graph, b) Transformed graph.

3.9 Case Study

The multiobjective model is implemented for the network illustrated in Figure 3.12, which is also studied in [8]. We aim at locating the Pareto optimal solutions in order to conceive the trade-offs between the objective functions, by solving $Ch'(N, \delta)$ for various weight vectors and N values. In this case study, routing costs of unity are assigned to each link. The links are unidirectional and have a capacity of 50 units/sec. A full traffic demand matrix from a single priority is assumed, so there exist 90 traffic requests. The traffic demand matrix is given in Figure 3.13 and all the traffic can be accepted by the network without capacity violation. The traffic requests are allowed to use the paths which have at most four hops. The optimal solutions are obtained by solving the problems with the Cplex 6.6 optimizer [1].

In Figure 3.14, we show the trade-off curves for $\Gamma(N)$ when N is taken as 90, 92, 94, and 96. The efficient solutions, obtained by solving $Ch'(N, \delta)$ for various δ and each N value, are shown in the figure. The points are connected by lines in order to increase the visual perception of the trade-off. Our first observation in Figure 3.14 is that, the trade-off curves exhibit worse performance regarding the load balancing and routing costs, as Constraints (3.18) become more restricted. For a specific value of the load balancing costs (routing costs), the routing costs (load balancing costs) increase as the number of LSPs decreases.

The two objectives, minimizing the routing cost and minimizing the number of LSPs, support each other in the sense that both objectives can be minimized to their optimal values concurrently. However at this solution we obtain a very high load balancing cost. This solution corresponds to the values of the routing and load balancing costs

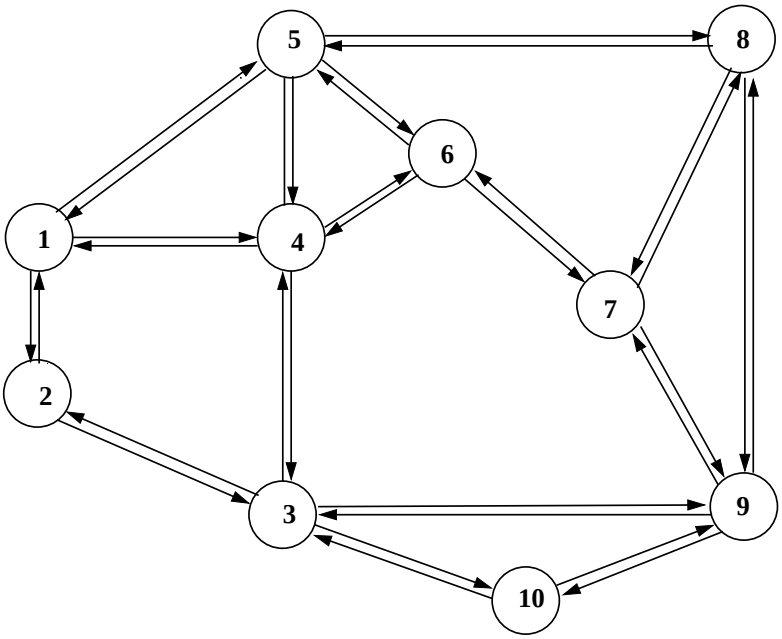

Figure 3.12: Example network topology consists of 10 routers and 32 links.

1	—	6.7	6.2	3.8	9.6	4.4	3.3	9.5	5.3	8.8
2	6.2	—	5.7	2.3	6.3	2.6	4.3	4.8	4.4	6.4
3	4.1	6.8	—	3.9	5.6	4.1	4.2	5.4	6.7	5.4
4	5	3	3.7	—	3.9	4.3	4.1	3.7	4.4	3.7
5	9.8	3.6	4.5	2.6	—	3	2.5	9.3	5	10.2
6	3.7	3.4	4.4	4.1	4	—	4.8	3.6	4.2	4.2
7	3.8	3.8	3.1	4.1	4	4.8	—	3.6	4.2	4.2
8	10.5	5.7	5.3	3.8	10.1	3.5	3.6	—	4.7	10.8
9	5.4	5.6	6.7	4.2	5.6	4.4	5	4.5	—	3
10	9.4	5.8	4	3.5	10.2	2.4	5	10.5	4.3	—

Figure 3.13: A full traffic demand matrix is assumed for the case study.

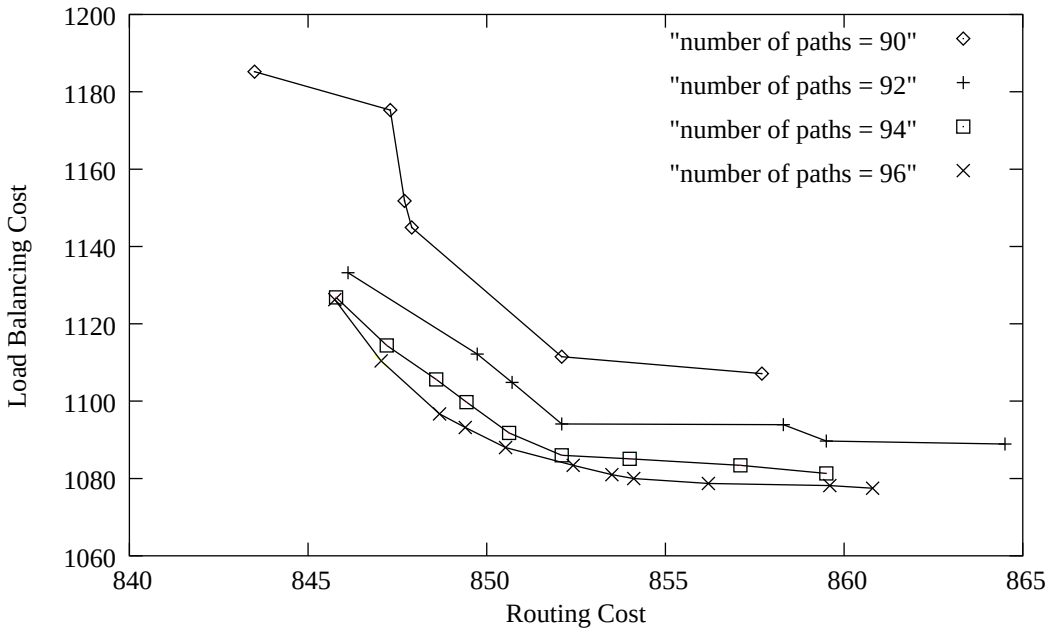

Figure 3.14: The trade-off curves perform better as the number of utilized paths constraint relaxes.

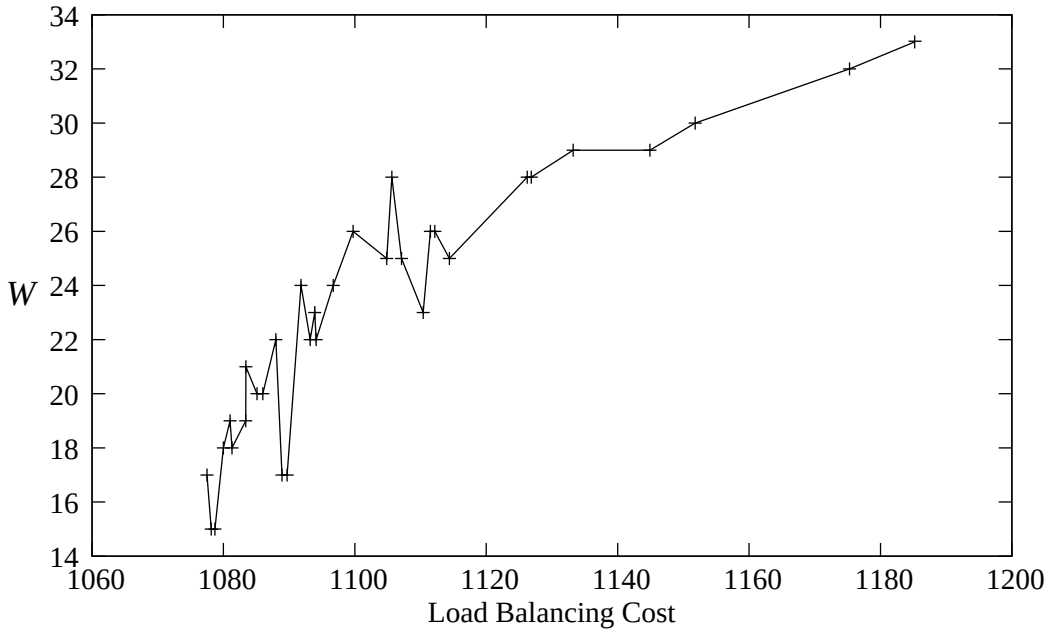

Figure 3.15: The deviation of the links utilization with regard to the load balancing function is illustrated graphically.

of 843.5 (optimal routing cost) and 1185.2, respectively.

Interestingly, the following general observation is true for all of the curves in Figure 3.14. When the routing costs are kept at low values (especially for values less than 852), the load balancing cost suffers dramatically. Moreover, the trade-off curves become closer as the number of LSPs increases. This is because of the fact that we loosen the number of LSPs constraints of the problems.

As a last remark on Figure 3.14, we observe that the shape of (some of) the trade-off curves may include non-convex parts. This supports our choice of the lexicographic weighted Chebyshev metric method over the weighted sum method for the trade-off analyses.

Additionally, we have made the following observation regarding the effect of load balancing on the network. The shape of the load balancing function allows for a classification of the links into regions according to their utilization rates. As seen in Figure 3.4, we have six regions: $[0, 0.5]$ – $(0.5, 0.6]$ – $(0.6, 0.7]$ – $(0.7, 0.8]$ – $(0.8, 0.9]$ – $(0.9, 1]$. During the case study we have observed that the average utilization rates (range between 0.527 and 0.541) and the maximum utilization rates (ranges between 0.7 to 0.74) don't change dramatically through the Pareto optimal solutions. However, the load balancing function has a more striking effect on the distribution of the links into the regions. When the second region is defined as the target (region of the average utilization rates), the balanced distribution of the load will imply having as many links as possible in the second region and having as few links as possible far away from the second region. The following weighting function gives a general idea about the distribution of the links around the target, $i^* = 2$.

$$W = \sum_{i=1}^{6} \mid i - i^* \mid \; n_i \tag{3.29}$$

where n_i denotes the number of the links in the i^{th} region. Notice that the weights increase as the distance from the target region becomes larger. Figure 3.15 shows the values of W versus the load balancing function. In the figure, we observe that W tends to increase as the load balancing function increases.

3.10 Summary

This chapter has introduced a multiobjective model for the off-line LSPs selections for traffic requests in an MPLS network. The objectives that are selected are namely; minimal routing cost, load balancing and minimal LSP assignment. The nature of the model is very robust, it allows itself to represent networks which have traffic from various priority levels.

In this chapter, the main focus has been on the complexity of the problem and the location of the Pareto optimal solutions to visualize the trade-off curves. For the exact solution approach, the main problem is decomposed into a series of subproblems. Additionally, a theoretical analysis is carried out to investigate the solution attributes of the decomposed problems. The lexicographic weighted Chebyshev method has been applied to the subproblems and its performance is observed for a case study. Although this exact method allows for the representation of the whole Pareto frontier, the method is not efficient and effective for such a large scale problem. We are mainly interested in the development of an approximation algorithm.

4 Modern Heuristic Approaches for Multiobjective Optimization

In [66], *heuristics* are defined as techniques which seek near-optimal solutions at reasonable computational costs without being able to guarantee the quality of the solutions (i.e., its feasibility and/or closeness to the optimal solution).

The main challenge facing multiobjective approximation techniques is the requirement to obtain a set of solutions approximating the Pareto front. The techniques based on the iterative approximations of some SOP problems constructed by the weighted aggregation of the objectives constitute a naive heuristic category in MOP. These techniques require multiple runs with different weights and knowledge about the shape of the objective functions. Furthermore, they are considered to be inefficient, since the Pareto optimal solutions visited during the runs are missed, and different runs may end up with the same solution. Thus, it will be interesting to utilize heuristic techniques for a more effective search process in MOP.

Lately in the literature, some heuristic algorithms inspired by some natural processes (thermodynamics, evolution, etc.) have been developed. They are known as *modern (meta) heuristic techniques*. This chapter first introduces the most popular modern heuristic techniques. After investigating the key difficulties within MOP, the main attributes of modern heuristic algorithms previously developed in the literature are explored. This chapter concludes with the introduction of the performance assessment metrics used in this thesis.

4.1 Overview of Modern Heuristic Approaches

Modern heuristic techniques can be classified into two main groups: heuristics based on *local* (*neighborhood*) *search* and heuristics based on *evolutionary methods*. An overview of some modern heuristic techniques is presented in this chapter.

The basic idea in local search is to evaluate a chain of solutions which are selected repeatedly from the neighborhood of the current solution. *Hill-climbing methods* described in [58] have the simplest form of local search. The basic structure is given in Algorithm 1. It starts with an initial solution and proceeds with the random selection of a solution from the neighborhood of the current solution. If the child solution

outperforms the parent one, it replaces the parent solution. The algorithm continues similarly until some certain termination conditions are met. The algorithm can run for a specified number of iterations or until there is no further improvement for some iterations.

Algorithm 1 Hill-climbing

Data: $\mathbf{v}^p$ parent solution
$\mathbf{v}^c$ child solution
Functions: $init()$ returns an initial solution
$local(\mathbf{v})$ returns a solution from the neighborhood of $\mathbf{v}$
$eval(\mathbf{v})$ returns the evaluation value of $\mathbf{v}$
$terminate()$ returns true if the terminating condition is met

$\mathbf{v}^p \leftarrow init()$
repeat
  $\mathbf{v}^c \leftarrow local(\mathbf{v}^p)$
  if $eval(\mathbf{v}^c)$ is better than $eval(\mathbf{v}^p)$ **then**
    $\mathbf{v}^p \leftarrow \mathbf{v}^c$
  end if
until $(terminate() = true)$

The basic forms of local search, like hill-climbing, have the disadvantage that they usually converge to a local optimum rather than a global one. Some approaches, like *tabu search* and *simulated annealing*, have been developed to overcome this deficiency.

Tabu search [66, 31] utilizes the concept of *memory* to guide the search. There are two types of memory: *recency-based* and *frequency-based*. The recency-based memory is used for short term strategies. At each iteration of tabu search, a subset of the neighbors of the current solution is evaluated. The neighbor with the best performance is chosen to continue with. The recency-based memory stores the moves which are *tabu* (not allowed) in the neighborhood of the current solution and is used to prevent the revisits to the areas that have been already encountered during the local search. The frequency-based memory is used for two types of longer-term strategies during the whole search process. *Intensification* aims at generating solutions by incorporating the attributes of the good solutions that have been encountered so far. *Diversification* strategies create solutions that have the attributes diverse from those evaluated previously and direct the search to the new regions.

Simulated annealing was first introduced by Kirkpatrick et al. in [43]. Its ideas are based on the techniques to simulate the cooling of materials in the heat bath until they reach a frozen state. This process is known as *annealing*. Simulated annealing is capable of locating the global optimal solution rather than being trapped in a local optima, since it may accept a neighbor as the working solution despite its

worse performance. The acceptance depends on the control parameter (*temperature*), and the magnitude of the decrease in the performance. The general algorithm for simulated annealing is given in Algorithm 2. A child solution $\mathbf{v}^c$ replaces the parent solution $\mathbf{v}^p$ with probability

$$\pi(temp, \mathbf{v}^p, \mathbf{v}^c) = \min\{1, \exp\left((eval(\mathbf{v}^p) - eval(\mathbf{v}^c))/temp\right)\}$$

where *temp* is the current temperature.

During the early iterations, when the temperature is high, the probability of accepting a worse solution is higher. However, as the temperature decreases, the acceptance of a worse solution becomes more difficult. During the implementation phase, the user should decide on some parameters. In [37], some advice is given according to the experiences.

- Initial temperature: the initial value of the temperature should be large enough to allow almost all transitions to be accepted.
- The cooling schedule: The user decides on the rate at which the temperature is decreased. Typically, the temperature is multiplied by a constant r: $temp = r\,temp$. Experiences suggest $0.8 \leq r \leq 0.99$.
- The terminating condition: The most popular way to stop the algorithm is when $temp < temp_{stop}$ where $temp_{stop}$ is a small value close to 0. Another possible terminating condition is an upperbound on number of iterations at which the evaluation value stays same.

These build only a subset of the variants of the algorithm. There are more alternative decision choices. For example, the number of replications where the temperature stays constant can also be dependent on the temperature. A more complete treatment of simulated annealing and its applications to some problems are carried out in [66].

Unlike the deterministic tabu search, simulated annealing is a stochastic approach. In the literature, a substantial amount of research has been carried out on the convergence of simulated annealing algorithms. It is interesting to note here simulated annealing is known to find the global optimum with probability of unity under certain conditions. Interested readers are pointed to [2] and [49] concerning the convergence issues. Due to the theoretical studies about its convergence, simulated annealing has gained high respect in the literature and it has been widely used.

The first observation about evolutionary algorithms is that they operate on a pool of solutions whereas local search-based methods rely on a single solution for exploration. This property supplies the main power of evolutionary algorithms, as it is analogous to a parallel operating machine. It can generate, evaluate and operate on multiple

Algorithm 2 Simulated annealing for a minimization problem

Data: $\mathbf{v}^p$ parent solution
$\mathbf{v}^c$ child solution
temp temperature
N_{rep} number of replications

Functions: *init*() returns an initial solution
init_temp() returns an initial temperature
local($\mathbf{v}$) returns a solution from the neighborhood of $\mathbf{v}$
eval($\mathbf{v}$) returns the evaluation value of $\mathbf{v}$
rand$(0, 1)$ returns a random number from the range $[0, 1)$
new_temp(*temp*) returns the updated temperature
terminate() returns *true* if the terminating condition is met

$\mathbf{v}^p \leftarrow init()$
$temp \leftarrow init\_temp()$
repeat
  for $i = 1$ to $i = N_{rep}$ **do**
    $\mathbf{v}^c \leftarrow local(\mathbf{v}^p)$
    if $eval(\mathbf{v}^c) \leq eval(\mathbf{v}^p)$ **then**
      $\mathbf{v}^p \leftarrow \mathbf{v}^c$
    else if $\exp\left((eval(\mathbf{v}^p) - eval(\mathbf{v}^c))/temp\right) > rand(0, 1)$ **then**
      $\mathbf{v}^p \leftarrow \mathbf{v}^c$
    end if
  end for
  $temp \leftarrow new\_temp(temp)$
until $(terminate() = true)$

solutions simultaneously. However one drawback of this approach is that the user has less control on the overall search mechanism.

An evolutionary algorithm operates on *individual*s (*chromosome*s) which represent an encoded solution to a specific problem. After a random initial population is generated, each individual is evaluated according to the objective function and is assigned a *fitness* value. At each iteration, the population enters a selection phase where the mating individuals are selected according to their fitness values. The chance of being selected for mating is higher for individuals with better fitness values. The next population is generated after applying *crossover* (i.e., *recombination*) and *mutation* operators on these individuals. The stopping criteria may either be a fixed number of generations or it can also be a user specified function (e.g., improvement from one generation to the next one). The structure of a standard evolutionary algorithm is given in Algorithm 3.

Genetic algorithm is a more specialized evolutionary technique which works on bit-

Algorithm 3 Evolutionary algorithm

Data: n iteration number
$P(n)$ population at iteration n

Functions: $init()$ returns an initial population
$eval(P)$ evaluates all individuals in P
$select(P)$ select individuals from P for mating
$recombine(P)$ performs crossover operation on P
$mutate(P)$ performs mutation operation on P
$terminate()$ returns $true$ if the terminating condition is met

$n \leftarrow 0$
$P(n) \leftarrow init()$
$eval(P(n))$
repeat
  $n \leftarrow n + 1$
  $P(n) \leftarrow select(P(n-1))$
  $recombine(P(n))$
  $mutate(P(n))$
until $(terminate() = true)$

string chromosomes. Genetic algorithms are the most popular type of evolutionary algorithms studied in the literature. Evolutionary and genetic algorithms are explained with some application examples in [32], [58] and [66]. Similar to simulated annealing, in the literature many theoretical studies have been carried out on the convergence properties of evolutionary algorithms [7], [32].

After an introduction, a short discussion and literature survey is now given on the implementation of the heuristics for MOP. The existing literature is studied deeply in [21], [41] and [45]. The surveys in [13], [42] and [90] focus only on evolutionary multiobjective optimization.

4.2 Key Issues in Multiobjective Heuristic Approaches

The aim of MOP is to obtain a final set which approximates the Pareto frontier. There are two main concerns with the approximation: *accuracy* and *diversity*.

Accuracy aims at finding solutions which are as close as possible to the Pareto optimal solutions. The concept of diversity has two tasks [17]: *extent* and *distribution*. Extent refers to the solutions at the extreme values of the objective functions, whereas the distribution concerns how uniformly the obtained solutions are located in the final set. Due to these multiple dimensions related to the performance of the final set, the MOP algorithms are usually difficult to compare.

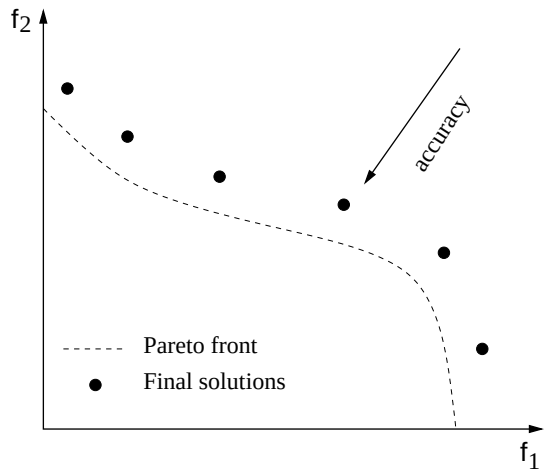

Figure 4.1: Accuracy increases as the solutions get closer to the Pareto frontier.

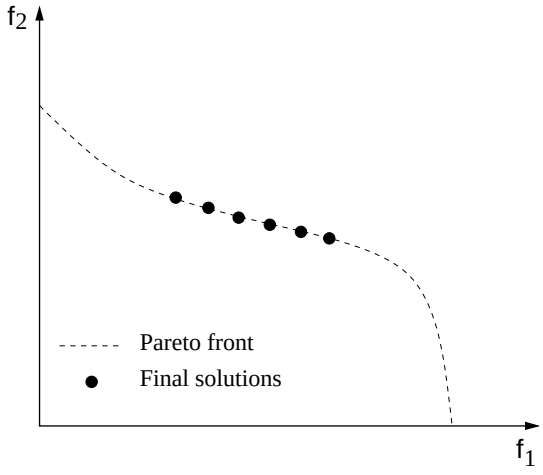

Figure 4.2: The extent of the final set is also important for the quality of the algorithm.

As it is illustrated in Figure 4.1, accuracy of the final set increases, as the solutions get closer to the true Pareto front. An example final set with a high accuracy but suffering in terms of the extent is depicted in Figure 4.2. Figure 4.3 shows a final set which is distributed nonuniformly over the Pareto front.

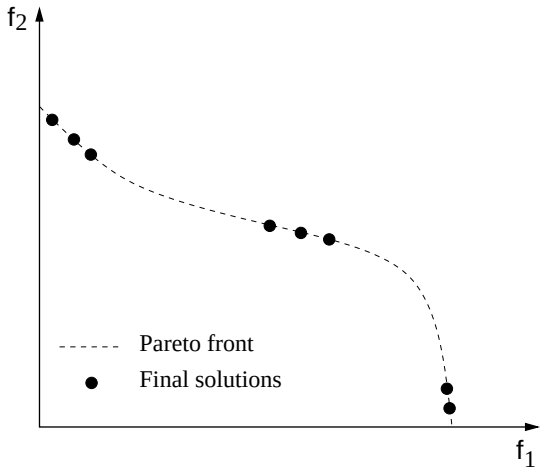

Figure 4.3: The distribution of the final set has an effect upon the quality of the approximate set.

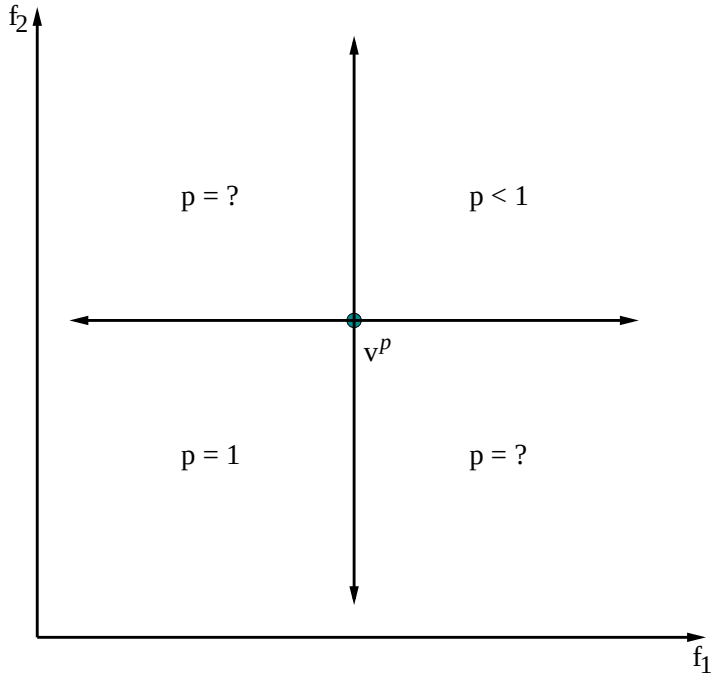

Figure 4.4: The acceptance criteria in simulated annealing methods differ from each other mainly in the regions where the child solution becomes indifferent to the parent.

4.3 Heuristic Techniques with Multiple Objectives

4.3.1 Local Search-Based Techniques for Multiobjective Problems

Local search methods approximate the Pareto front according to some acceptance rules usually based on the weighted aggregation of the objective functions. The determination of the weights can be a priori, guided and random [21].

Simulated annealing for MOP was first implemented by Serafini [73] to find a subset of the nondominated solutions. Three possible relations are defined between the parent solution $\mathbf{v}^p$, and the child solution $\mathbf{v}^c$:

Rel. (a) $\mathbf{v}^c$ weakly dominates $\mathbf{v}^p$,

Rel. (b) $\mathbf{v}^c$ and $\mathbf{v}^p$ are indifferent to each other,

Rel. (c) $\mathbf{v}^c$ is dominated by $\mathbf{v}^p$.

In the first case, the child solution is accepted as working solution with probability equal to one. For the third case the child solution should be accepted with probability strictly less than 1.0. However, three alternative approaches exist for the case of indifference, as it is depicted in Figure 4.4:

- Accept an indifferent solution with probability equal to unity. Serafini called this approach *weak criterion*.
- Accept an indifferent solution with probability strictly less than one. This approach is called *strong criterion*, since only the dominating points are accepted with probability equal to one. An acceptance rule based on the weighted Chebyshev norm of the objective function values is an example for this rule:

  $$\pi(temp, \delta) = \min\{1, \exp((\max_i\{\delta_i(f_i(\mathbf{v}^p) - z_i^{ref})\} - \max_i\{\delta_i(f_i(\mathbf{v}^c) - z_i^{ref})\})/temp)\}$$

  where $\sum_{i=1}^{Q} \delta_i = 1$. When the parent solution is selected as the reference point, we obtain

  $$\begin{aligned} \pi(temp, \delta) &= \min\{1, \exp(-\max_{i\in\{1,\ldots,Q\}}\{\delta_i(f_i(\mathbf{v}^c) - f_i(\mathbf{v}^p))/temp\})\} \\ &= \min\{1, \exp(\min_{i\in\{1,\ldots,Q\}}\{\delta_i(f_i(\mathbf{v}^p) - f_i(\mathbf{v}^c))/temp\})\}. \end{aligned}$$

  In this case, the acceptance probability will be equal to one if and only if the child weakly dominates the parent solution.
- These two approaches can be combined, i.e., the solutions indifferent to the parent are subject to an acceptance probability either equal to one or strictly less than one, depending on their objective function values and on the acceptance rule. E.g., an acceptance rule based on the weighted sum of the objective functions leads a subset of the indifferent solutions to be accepted with probability equal to unity:

  $$\pi(temp, \delta) = \min\{1, \exp(\sum_{i=1}^{Q} \delta_i(f_i(\mathbf{v}^p) - f_i(\mathbf{v}^c))/temp)\}$$

  where $\sum_{i=1}^{Q} \delta_i = 1$.

Apart from these examples, Serafini proposed some more rules under these criteria. These rules differ in their methods to aggregate the objective function into a weighted norm. He proposed to determine the weight vector a priori and change it randomly by small amounts during the run. He also analyzed the convergence properties of the algorithm to the nondominated set under some various rules.

Ulungu et al. [81] proposed a multiobjective simulated annealing algorithm which is called MOSA. MOSA utilizes the combined strategy in the case of indifference between the parent and child solution. The quality of a solution is measured by the weighted mean of the objectives. The nondominated solutions among the ones generated during the algorithm run are collected in an external set. The main algorithm is run several times, each time with a specific weight vector. All nondominated sets obtained from the runs are combined into a final set which is further filtered by a pairwise comparison process to remove the dominated solutions.

In [62], Nam and Park proposed another multiobjective simulated annealing algorithm called also MOSA. The algorithm works under the weak criterion. For the case of a dominated child generation, six different rules for the transition probability calculations are suggested and evaluated. The proposed multiobjective simulated annealing algorithm is compared with a multiobjective evolutionary algorithm for a specific problem. Their preliminary results show that simulated annealing outperforms evolutionary algorithm for the problems with a small search space.

In [41] another simulated annealing algorithm called Pareto simulated annealing (PSA) was presented. PSA differs dramatically from the other algorithms, since it uses a population of generating solutions where each of them utilizes simulated annealing to explore the search space. A specific weight vector is assigned to each generating solution and the weights of a solution are changed in order to explore the diverse areas in the search space. Both weak and combined acceptance approaches are investigated in case of Rel. (b). PSA keeps an external set to record the potentially Pareto optimal solutions. This set is updated at each iteration.

Hansen presented in [34] a tabu search multiobjective algorithm which has some similarities with PSA. Like PSA, it works on a set of generating solutions and an external set for the nondominated solutions encountered during the run. It assigns a weight vector to each generating solution and updates them to explore the diverse areas.

Another algorithm utilizing tabu search is developed by Abdelaziz and Krichen in [3]. The algorithm works on a single solution. At each iteration the algorithm either generates a set of solutions obtained by a diversification process, or a subset of the neighborhood of the current solution. The first case is implemented to direct the search to the diverse areas. The external set of potentially efficient solutions are updated by these newly generated solutions. The next solution is selected randomly from the generated solutions.

Knowles and Corne introduced in [46] a local search-based method called $(1 + 1)$ Pareto archived evolution strategy (PAES). The algorithm utilizes an archive to store the nondominated solutions previously found. At each iteration, the child is rejected, if it is dominated by the parent solution or it is accepted, if it is superior to the parent. In case of indifference, the acceptance rule depends on its relation with both

the parent solution and the solutions in the archive. The direction of the search is guided by the external set. The solution which resides in the less occupied region of the external set is accepted as working. The studies about the convergence of this algorithm to the nondominated set can be found in [45].

4.3.2 Evolutionary Multiobjective Algorithms and Archiving Strategies

Since evolutionary algorithms deal with a set of solutions simultaneously, they are particularly suitable for MOP. Evolutionary multiobjective studies have begun with the vector evaluated genetic algorithm (VEGA) by Schaffer [69]. In VEGA, the mating population is divided equally by the number of objectives in the problem. The selection is carried out separately for each objective function to fill the corresponding portion of the mating pool. After VEGA, Goldberg suggested a selection procedure where the fitness values of the individuals are calculated according to their dominance relations [32]. To keep diversity among the population he proposed to utilize niching mechanisms which aim at building stable subpopulations representing different subdomains of the search space. Niched Pareto genetic algorithm (NPGA) by Horn et al. [36] and (NSGA) by Srinivas and Deb [76] implemented Goldberg's idea. These studies have a common attribute where the last generation is accepted as the output of the algorithm.

Evolutionary algorithms for MOP have made a great improvement, after the introduction of *elitism* to incorporate the nondominated solutions encountered so far into the next generation. Following some initial studies utilizing elitism (e.g., [12], [61], [64]), a number of well-defined elitist evolutionary algorithms have been lately developed: strength Pareto evolutionary algorithm (SPEA) in [90], PAES (M-PAES)[1] in [46] and NSGA-II in [18].

In a single objective problem an elitist algorithm ensures the existence of the best individual(s) of the current population in the next population, even though they are not chosen at the selection process [7]. After its explicit success in the evolutionary MOP, the concept of the elitism is reformulated in [51]. An elitist approach guarantees a strictly positive probability for selecting at least one nondominated individual as operand for mutation and crossover operators. Most of the elitist algorithms are implemented with an external set (*archive*) storing the nondominated individuals encountered so far. The output of these algorithms is then this archive obtained after the last iteration, which is here denoted as the final set.

[1] (1+1) PAES lies in the domain of elitist methods, however elitism is developed for evolutionary algorithms. Therefore, it is in fact not very interesting to talk about elitism for local search-based heuristics.

The importance of the archive has led several proposals about its maintenance. The number of the Pareto optimal solutions can be quite large in a MOP problem. It may not be feasible to store all of the nondominated solutions encountered during the algorithm run. Therefore, the size of the archive should be bounded. As it is stated in Section 4.2, an ideal final set should contain solutions as diverse as possible over the Pareto front. Due to its active role in the guidance of the algorithm, it is of enormous importance to decide on the individuals which will survive in the archive. The experiments in [52] show that the effect of the size reduction operator for the archive increases with the elitism intensity (the probability to select a mating parent from the archive instead of the working population).

Zitzler and Thiele in [90] used a clustering approach to limit the size of the archive. At each iteration of the algorithm the set is updated with the newly found nondominated individuals. If the size of the external set exceeds its limit, a clustering operation is carried out. Each of the solution builds one cluster at the beginning. Then, the clusters are combined according to the average distances between each other. This step is repeated until the number of clusters decreases to the limit of the external set. For each cluster the centroid individual is selected as the representative, and, the others are deleted from the cluster. This approach has some weaknesses: it may lose the individuals with extreme objective function values, and, it has a high computation overhead.

Knowles and Corne developed in [46] an archiving system denoted as *adaptive grid archiving*. The system is based on the division of the objective space into some grid regions. The lower and upper boundaries of the objective functions are automatically determined by the nondominated solutions in the archive. They are updated, if necessary, when new individuals enter the archive. Practically, all of the objective functions have an equal number of partitions. For a problem with Q objectives and t partitions for each objective function, the number of grid regions will be t^Q. The number of grid regions stays constant during the algorithm run, but the coordinates of the grid regions change with the boundaries of the objective functions. When a new nondominated solution is found by PAES, the algorithm inserts it to the archive. If the size of the archive exceeds its limit, individuals from the most densely occupied grid region are deleted.

This algorithm is time-effective, when usual comparison, addition and delete operations are carried out. The algorithm has the weakness that each time the boundaries of the objective functions change, the archiving algorithm recalculates the coordinates of the grids and the location of each individual according to the new grid regions. In [45], the relationship between the size of the archive, the number of partitions and the archiving performance are investigated and some proposals on the archive size are given concerning the number of partitions and the number of objective functions.

Deb et al. introduced another approach for the density estimation to ensure diver-

sity in the population [18]. To estimate the density of the solutions surrounding a particular point in the decision space, the distances between two closest neighbors on either side of this point along each of the objectives are taken into consideration. The sum of these distances over all of the objectives is called the *crowding distance* of the point. A large crowding distance value indicates that the region of the corresponding point is relatively less occupied. The boundary points of each objective are assigned a crowding distance of infinity, so that they are always favorable. The crowding distance approach is computationally less expensive compared to the other archiving mechanisms.

4.3.3 Hybrid Approaches

Hybrid heuristic techniques are founded on the integration of evolutionary algorithms with local search-based methods. Several hybrid heuristic techniques have been developed in the past years for MOP problems. These algorithms are out of the scope of this thesis and will not be explained in detail. Examples for multiobjective genetic local search algorithms can be found in [39] and [41]. Knowles and Corne have developed an extended version of PAES called Memetic-PAES [47].

4.4 Performance Measures to Evaluate the Heuristics for Multiobjective Optimization

Although the algorithms for SOP problems locate a single solution, the algorithms for MOP problems end up with a set of solutions approximating the Pareto frontier. Several metrics have proposed in the framework of MOP to evaluate the performance of alternative algorithms. Two main challenges exist for the quality evaluation of the Pareto frontier approximations:

- As it has been discussed in the previous sections, the algorithms aim at several criteria for an approximation set: closeness to the Pareto front, diversity, uniform distribution of the solutions, etc.
- Since the main interest is on the problems of high complexity, usually the exact nondominated solutions are not available due to the high run-time requirements. As a result, one needs to compare some sets of solutions, all of which are approximations.

Under these issues, it is difficult to create a common agreement about the quality metrics for multiobjective optimizers. Different types of metrics have been proposed

lately in the literature. These metrics can be classified into two groups: specific-purpose metrics, general-purpose metrics. The former includes the metrics which evaluates only one aspect of the approximate set. The *error ratio* [82], *coverage* [89], [87] are examples for the metrics evaluating the dominance power of the approximation set. The metrics like *spacing* [71] and *spread* [17] measure the performance of an output set in terms of diversity. *$\mathcal{S}$-metric (hypervolume)* [89] evaluates the overall quality of the approximation set and belongs to the second group. For a theoretical analysis of the available metrics and for further information, the reader is pointed to the works [17], [45], and [88].

4.4.1 Performance Metrics used in this Thesis

Two metrics used in this thesis are explained in more detail here. These metrics have the advantage that both of these measures do not require the existence of Pareto optimal solutions.

The dominance metric

In this thesis, a modified version of the coverage metric is used. Coverage is used to compare the dominance relation between two solution sets. The coverage metric $\mathcal{C}(A, B)$ returns the proportion of the solutions in B, which are weakly dominated by the solutions in A:

$$\mathcal{C}(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|}$$

Since coverage is not symmetric, both of the directions should be taken into consideration for the comparison of two approximate sets. Note that $\mathcal{C}(B, A)$ is not necessarily equal to $1 - \mathcal{C}(A, B)$.

Although this metric originally aims at the comparison of approximate sets, it can be also used to evaluate the quality of an algorithm by taking A as a set of efficient solutions. However, in this case a perfect algorithm ending up with a set of solutions equal to the set of Pareto optimal solutions will get a coverage value of 1. For this reason, we changed the $\mathcal{C}$-metric such that it only gives the proportion of the solutions which are dominated by the reference set. To prevent confusion, we call it *dominance* and denote it with $\mathcal{D}$.:

$$\mathcal{D}(A, B) = \frac{|\{b \in B | \exists a \in A : a \prec b\}|}{|B|}$$

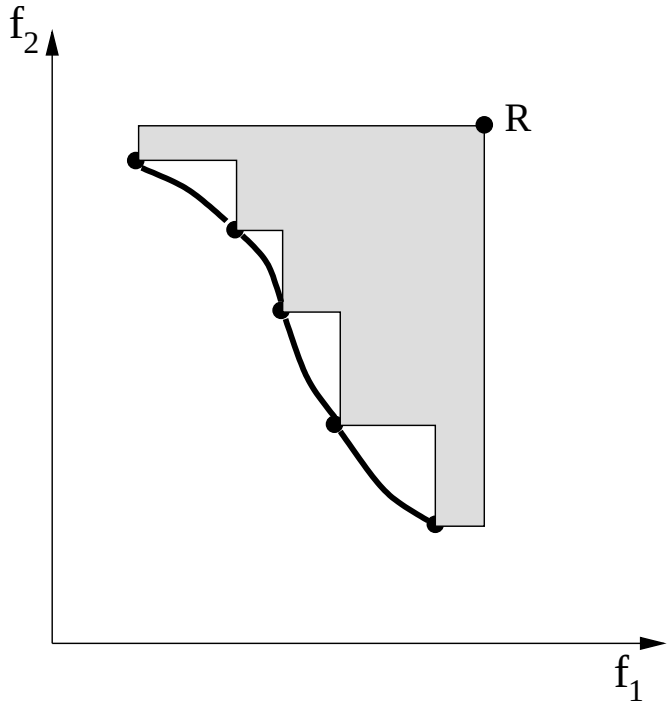

Figure 4.5: The $\mathcal{S}$-metric is based on the size of the dominated space by the nondominated solutions.

The $\mathcal{S}$-metric (Hyper-volume)

The $\mathcal{S}$-metric calculates the size of the objective space dominated by the output of the algorithm. This metric gives the volume enclosed by the union of the hyper-cubes constructed for each solution in the output set with a reference point. The reference point in the objective space is selected such that it is dominated by all of the solutions in the approximation set. The reference point and each solution build the diagonal corners of the hyper-cubes. Figure 4.5 illustrates these hyper-cubes for a bi-objective problem.

This metric gives an idea about the overall quality of an algorithm. The value of the metric grows as the solutions get closer to the Pareto frontier, and, as the diversity between the solutions and the extent of the set increase. This metric is sensitive to the scaling of the objectives. To overcome this issue Veldhuizen suggested to use the ratio of the hyper-volume of the approximate set and of the efficient set [82]. Another approach is to normalize the objective function values before the calculation. This metric has the disadvantage that the ordering of the approximate sets may change with the selection of the reference point [45].

5 A Simulated Annealing-Based Framework for Multiobjective Traffic Engineering

The primary aim of heuristic multiobjective optimizers is to detect a set of solutions approximating the Pareto front. As discussed previously, the greatest challenge facing these heuristics is the requirement on the members of their output set to have a uniform distribution through the extent of the objective functions additional to being as accurate as possible to the Pareto front. The multiobjective modern heuristic techniques well known in the literature are investigated in the previous chapter. Taking these investigations into consideration, a heuristic framework for the multiobjective off-line traffic engineering problem is introduced in this chapter. Its performance under alternative search strategies is investigated.

In [24] the traffic engineering problem has been solved by the author using a hybrid heuristic method which combines an evolutionary algorithm with mathematical programming. The basic experimental results have shown the insufficiency of this approach in guiding the search towards solutions using few paths. The heuristic framework introduced here mainly utilizes simulated annealing in order to search the feasible set for the good and possibly Pareto optimal solutions. The main algorithm simplifies the search by dividing the feasible set into some subsets similarly to the ϵ-constraint method. For each subset, an independent simulated annealing algorithm is carried out. In order to store the nondominated solutions encountered during the run, an archive is maintained during the whole run. The algorithm has a nested structure where the same archive is kept during the whole run, i.e., the archive is initialized only at the very beginning, not at each sub-run.

The neighborhood structure of the framework is based on the LP relaxation of the original problem. Two different neighborhood functions are defined within the framework. The performance of the algorithm under these functions is investigated. We start the explanation of the framework with the discussion of its neighborhood structure. Since the neighborhood set of a solution consists of the points one simplex pivoting move away, we will give a brief information about the simplex algorithm. The introduction of the whole algorithm will be completed with the presentation of its key components: Decomposition of the problem, acceptance criterion, archiving

system, biased neighborhood function.

5.1 Zero-One Mixed Integer Programming and the Simplex Algorithm

We recall the decision variables of the off-line traffic engineering problem introduced in Chapter 3, and present them in a vector format for the ease of representation.

$$\begin{aligned} \mathbf{x} &= \{x_1^1, \ldots, x_{|T|}^{L_{|T|}}\}, \\ \mathbf{g} &= \{g_1, \ldots, g_{|E|}\}, \\ \mathbf{\Phi} &= \{\phi_1, \ldots, \phi_{|E|}\}, \\ \mathbf{y} &= \{y_1^1, \ldots, y_{|T|}^{L_{|T|}}\}, \end{aligned}$$

where $|T|$ and $|E|$ denote the number of the traffic requests and the number of the links in the network, respectively. L_t is the number of alternative paths available for the traffic request t.

In our problem, the decision vectors $\mathbf{x}, \mathbf{g}$ and $\mathbf{\Phi}$ are continuous, whereas $\mathbf{y}$ is a binary vector. Thus, our model is a linear zero-one mixed integer programming (MIP) problem. Any zero-one MIP problem can be stated in general as follows:

$$\text{MIP: } \min \mathbf{cr} + \mathbf{ds}$$

$$\begin{aligned} \text{subject to } \mathbf{Hr} + \mathbf{Js} &\leq \mathbf{b}, && (5.1) \\ \mathbf{s} &\leq \mathbf{e}, && (5.2) \\ \mathbf{r}, \mathbf{s} \geq 0, \quad &\text{and} \quad \mathbf{s} \in \{0, 1\} && (5.3) \end{aligned}$$

where $\mathbf{c}$ is $1 \times n$, $\mathbf{r}$ is $n \times 1$, $\mathbf{d}$ is $1 \times l$, $\mathbf{s}$ is $l \times 1$, $\mathbf{H}$ is $m \times n$, $\mathbf{J}$ is $m \times l$, and $\mathbf{b}$ is $m \times 1$. $\mathbf{e}$ is the vector of ones with the size of $l \times 1$.

The off-line traffic engineering problem in MPLS network can also be represented in this format by multiplying the "$\geq$" constraints with -1. The equality constraints can be replaced by two constraints, one in "$\leq$" format and the other one in "$\geq$" format.

The problem at hand is especially difficult due to its mixed-integer nature. In the literature, the number of modern heuristic approaches developed for MIP problems is not as large as those for combinatorial optimization problems and general integer programming problems. In [72], the mixed integer structure is recognized as one of the most challenging attributes of engineering problems. Løkketangen summarizes in [55] some of the heuristic approaches and their variants formulated for zero-one MIP problems.

The heuristic framework is based on the LP relaxed version of the original problem. When the integer requirements for the zero-one variables are dropped, the problem becomes an LP problem which is shown to be solvable in polynomial time by the ellipsoid algorithm [63].

5.1.1 Linear Programming and the Simplex Method

A standard linear programming problem is stated as follows:

$$\text{LP:} \quad \min \mathbf{cr}$$

$$\begin{aligned} \text{subject to } \mathbf{Hr} &\leq \mathbf{b}, && (5.4)\\ \mathbf{r} &\geq 0 && (5.5) \end{aligned}$$

where $\mathbf{H}$ is $m \times n$, $\mathbf{r}$ is $n \times 1$, $\mathbf{b}$ is $m \times 1$.

LP problems are usually solved by the *simplex method* which is accepted efficient in practice [11], [63]. The simplex method searches an optimal solution by moving from one *extreme* (*corner*) *point* to another adjacent one of the polyhedron defined by the constraints of the problem. An extreme point of the polyhedron corresponds to at least one *basic feasible solution* of the problem and it is well known that an optimal solution is located at an extreme point.

The simplex method starts after the *slack* variables are inserted in the non-equality constraints to replace them with equality constraints. During the optimization process, it operates on both the slack and regular variables.

$$\min \mathbf{cr}$$

$$\begin{aligned} \text{subject to } \mathbf{Hr} &= \mathbf{b}, && (5.6)\\ \mathbf{r} &\geq 0 && (5.7) \end{aligned}$$

where $\mathbf{r} = (r_1, \ldots, r_n, r_{n+1}, \ldots, r_{n+m})$ denotes the decision vector including the regular and slack variables.

A basic feasible solution is obtained by partitioning the decision vector into the *basic* and *nonbasic* variables. At a basic solution, the value of each nonbasic variable is equal to its lower or upper bound and the basic variables build the current *basis*. The number of basic and nonbasic variables in any basic feasible solution is constant and depends on the rank of the constraint matrix. The following relationship exists between the basic and nonbasic variables:

$$\mathbf{r}_{\mathcal{B}} = \mathbf{H}_{\mathcal{B}}^{-1}\mathbf{b} - \mathbf{H}_{\mathcal{B}}^{-1}\mathbf{H}_{\mathcal{N}}\mathbf{r}_{\mathcal{N}}$$

where $\mathcal{B}$ and $\mathcal{N}$ denote the set of the basic and nonbasic variables, respectively. $\mathbf{H}_{\mathcal{B}}$ consists of the columns of $\mathbf{H}$ corresponding to the basic variables and it is singular. Similarly, $\mathbf{H}_{\mathcal{N}}$ is the matrix built by the columns of $\mathbf{H}$ related to the nonbasic variables.

The simplex algorithm moves from one extreme point to an adjacent one by changing the basis. At each iteration, it first selects a nonbasic variable which enters to the basis. The basic variable which has to leave the basis is selected after some matrix calculations so that the feasibility is kept. This process is known as *pivoting*. After pivoting, the adjacent basis $\mathcal{B}^{'}$ differs from $\mathcal{B}$ just by one variable.

$$\begin{aligned} \mathcal{B}^{'} &= \mathcal{B} \cup \mathcal{N}_r \setminus \mathcal{B}_s, \\ \mathcal{N}^{'} &= \mathcal{N} \cup \mathcal{B}_s \setminus \mathcal{N}_r \end{aligned}$$

where $\mathcal{N}_r$ and $\mathcal{B}_s$ denote the entering and leaving decision variables, respectively.

Example 5.1. Consider the following LP problem:

$$\min 3r_1 + r_2$$

$$\begin{aligned} \text{subject to } r_2 - r_1 + r_3 &= 0, \\ r_1 + r_2 + r_4 &= 6, \\ r_1 &\geq 2, \\ r_2, r_3, r_4 &\geq 0. \end{aligned}$$

In this example r_3 and r_4 are added as slack variables. At each iteration of the simplex method, an extreme point of the polyhedron is visited. In Figure 5.1, all of the extreme points are depicted by black dots. The optimal solution is found at $r_1 = 2, r_2 = 2$. The basic variables at this extreme point are r_2 and r_4, whereas r_1 and r_3 are nonbasic. r_1 is equal to its lower bound at this solution. The following relationship exists between basic and nonbasic solutions:

$$\begin{aligned} r_4 &= 6 - 2r_1 + r_3, \\ r_2 &= r_1 - r3. \end{aligned}$$

5.2 The Neighborhood Structure

In heuristics, the representation of the solutions plays an important role [58]. The representation deals with the encoding of the candidate solutions which undergo search operations during the run. For our local search based approach, a solution to our original problem is represented by the basis to its relaxed LP set. It is interesting

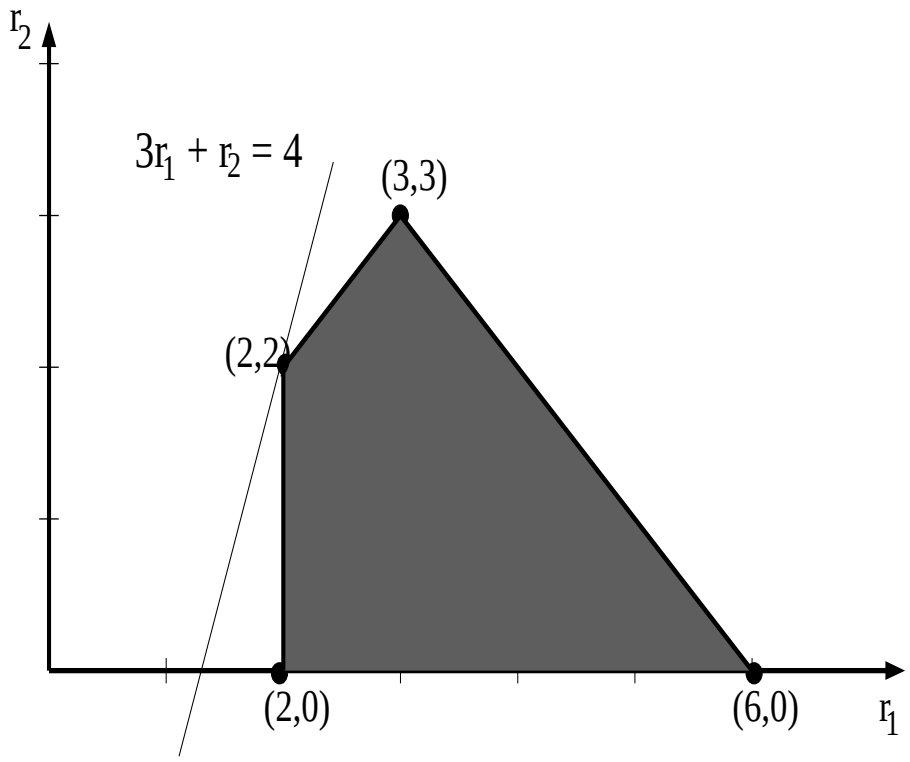

Figure 5.1: The simplex algorithm visits an extreme point at each iteration.

to note that the encoding should allow the heuristic methods to explore every possible solution that might be optimal. In the literature, it has been proved that every feasible solution to a general zero-one integer and zero-one MIP problem corresponds to one of the extreme points of its relaxed LP set [10], [30], [63]. In the literature, this fact resulted in some heuristic methods where the extreme points of the LP relaxation of zero-one (mixed) integer programming problems are searched systematically. E.g., Løkketangen and Glover investigated in [56] some tabu search strategies for single objective zero-one MIP problems where any solution to the problem is defined by a basis of the relaxed LP set.

Following the previous short explanation, we now enlighten this fact stated in [30] in its most general form. To this end, we need the general representation of a zero-one MIP after the addition of the slack vectors $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$:

$$\mathbf{Hr} + \mathbf{Js} + \boldsymbol{\alpha} = \mathbf{b}, \tag{5.8}$$

$$\mathbf{s} + \boldsymbol{\beta} = \mathbf{e}, \tag{5.9}$$

$$\mathbf{r}, \mathbf{s},\ \boldsymbol{\alpha}, \boldsymbol{\beta} \geq 0, \quad \text{and} \quad \mathbf{s} \in \{0, 1\}. \tag{5.10}$$

Theorem 5.2 (Glover, 1968, [30]). *If there is a feasible solution to Constraints (5.8) - (5.10) with* $\mathbf{s} = \mathbf{s}'$ *integer, then there is a basic feasible solution*

to the relaxed LP of these constraints with $\mathbf{s} = \mathbf{s}'$ *and* m *of the components of* $(\mathbf{r}, \boldsymbol{\alpha})$ *basic.*

The theorem implies that we can search the extreme points of the relaxed LP set for an optimal solution to the original problem. Please note that, the optimal solution of the original problem does not probably coincide with the optimal solution of the LP problem, it is hidden among the basic feasible solutions. When the binary constraints of the original off-line traffic engineering problem are relaxed, it becomes

$$\min(\sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^l, \sum_{m\in E}\phi_m, \sum_{t\in T}\sum_{l=1}^{L_t} y_t^l)$$

$$\begin{aligned} \text{subject to} \quad (3.2), (3.5) \;&-\; (3.12), \\ \frac{x_t^l}{d_t} \le y_t^l \;&\le\; 1 \quad \forall\, t \in T, \forall\, l \in \{1, \ldots, L_t\}, \qquad (5.11) \\ x_t^l,\; y_t^l \;&\ge\; 0 \quad \forall\, t \in T, \forall\, l \in \{1, \ldots, L_t\}. \qquad (5.12) \end{aligned}$$

Remark 5.3. *In the linear programming relaxation of the off-line traffic engineering problem, Constraints (5.11) and (5.12) are redundant.*

The remark expresses the redundancy of the constraints related with y_t^l and allows to work on an LP problem with smaller size, since $\mathbf{y}$ is totally removed from the problem. Its value is determined according to $\mathbf{x}$.

$$y_t^l = \begin{cases} 1 & \text{if } x_t^l > 0, \\ 0 & \text{otherwise.} \end{cases}$$

5.3 Key Components of the Heuristic Framework

Four key components of the framework will be discussed in details: decomposition of the problem, acceptance/rejection rules for the child solution, the archiving strategy, and the neighborhood function. The general structure of the framework is given with the explanation of the first component.

5.3.1 Decomposition of the problem

Due to its large scale, the main problem is divided into a series of subproblems similarly to the ϵ-criterion method explained in Chapter 2.

A constraint method-based heuristic approach for MOP problems has been previously proposed by Ranjithan et al. in [65]. The algorithm they introduced first relocates

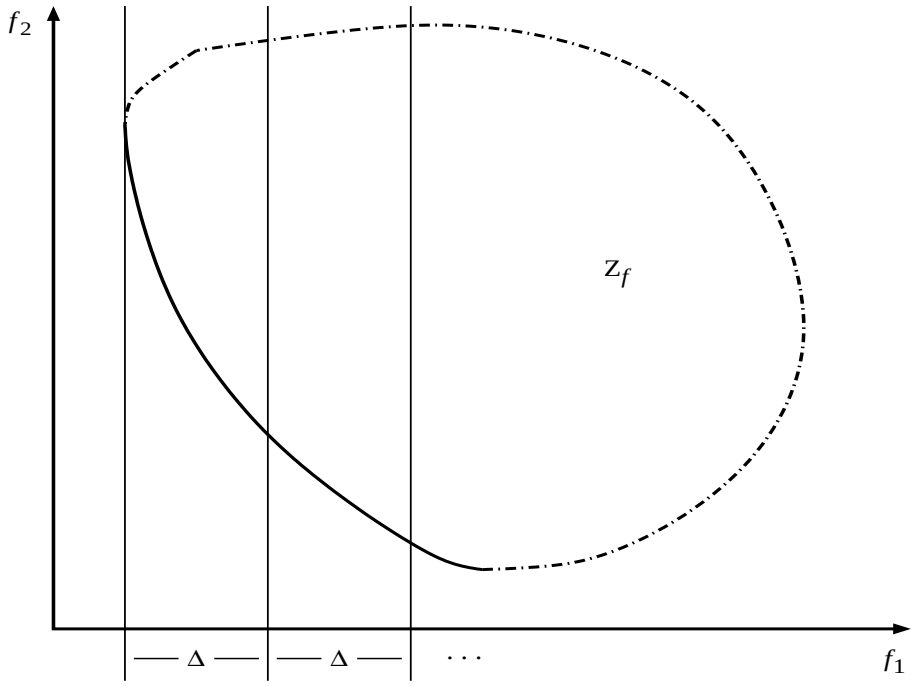

Figure 5.2: The feasible region is divided into sub-regions similarly to the ϵ-constraint method.

all of the objectives except one as a group of constraints. Thus, the MOP problem is converted into a SOP problem. At each iteration, a standard (single objective) EA is applied to obtain a single nondominated point. The constraints are updated systematically and the final population of the current iteration is used as a seed for the next evolutionary process with the idea that it will speed up its convergence to the nondominated solution. The final nondominated set is obtained by storing the best solutions found at the intermediate iterations.

Different from their approach, in our framework only one objective function is represented as a constraint. Hence, our framework operates iteratively on a series of sub-regions of the general LP relaxed feasible set of the off-line traffic engineering problem, where the sub-regions are specified with the addition of two constraints.

$$\sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^l \leq z_1^* + (k+1)\Delta, \tag{5.13}$$

$$\sum_{t\in T}\sum_{l=1}^{L_t} C_t^l x_t^l \geq z_1^* + k\Delta, \tag{5.14}$$

$$(3.2), (3.5) - (3.12),$$

$$x_t^l \geq 0 \quad \forall\, t \in T, \forall\, l \in \{1, \ldots, L_t\},$$

for $k = 0, \ldots, K-1$. In Constraints (5.13) and (5.14), the objective function minimizing the sum of the routing costs is bounded from above and below, respectively. At each main step of the framework the bounds of the constraint are shifted by Δ, as illustrated in Figure 5.2. Both the stepwise shift (shown with Δ) for Constraints (5.13), and (5.14) and the value of K representing the number of iterations are determined by the decision maker.

Multiobjective simulated annealing is applied to direct the search for each of the sub-feasible sets. The general framework is given in Algorithm 4.

The basic algorithm starts with an initially selected solution from the first sub-feasible set. Constraints (5.13), and (5.14) in the LP problem are updated with the function *update_cons*(k) where definition of k corresponds to the related constraints. The initial solutions are equalized to the optimal solutions which minimize the load balancing cost under related routing constraints. The framework keeps an external archive to store the nondominated solutions encountered during the whole algorithm run, whereas the algorithm in [65] aims to find a single solution at each iteration. In our framework, if a child solution is not weakly dominated by the parent, it is compared with the archive to delete the stored solutions dominated by the child and to find out whether the child is weakly dominated by the archive. The basic structure of the framework is given in Figure 5.3

5.3.2 Acceptance Criterion

For the acceptance strategy of the framework, we have chosen the combined approach introduced in Chapter 4. As it has been also emphasized in [81] and [45], a strong strategy focuses on the intensification, while a weak acceptance criterion focuses more on diversification. A compromise solution comes through a combined approach.

The acceptance rule is based on the weighted sum of the normalized objective function values and, is applied by the following expression:

$$\pi(\delta, temp, \mathbf{v}^p, \mathbf{v}^c) = \min\{1, \exp(\sum_{i=1}^{3} \frac{\delta_i(f_i(\mathbf{v}^p) - f_i(\mathbf{v}^c))}{R_i}/temp)\} \tag{5.15}$$

where δ_i and R_i are the weight vector and the range factor for the i^{th} objective function. $\sum_{i=1}^{3} \delta_i = 1$ and range factors are used to bring all of the objectives to the same scale: $[0, 100]$. R_1 is based on the values of Δ, whereas R_2 and R_3 take into account the global minimal values of the corresponding objectives functions and the maximal values encountered during the algorithm run.

As we have discussed in the Chapter 2, the weighted sum method fails to locate the solutions residing at the non-convex part of the Pareto frontier. Although the

Algorithm 4 Simulated Annealing-based Framework

Data: *archive* set keeping the nondominated individuals
k index for the sub-feasible sets
K number of sub-feasible sets
temp temperature
$\mathbf{v}^p$ parent solution
$\mathbf{v}^c$ child solution
N_{rep} number of replications
δ weight vector

Functions: *new_cons*(k) updates the constraint to define the related sub-feasible region
init_temp() returns an initial temperature
init(k) returns an initial solution from the corresponding sub-feasible set
init_arch() initializes the archive as an empty array
local($\mathbf{v}$) returns a neighbor solution of $\mathbf{v}$
domoreq($\mathbf{v}^1, \mathbf{v}^2$) returns *true* if $\mathbf{v}_1$ weakly dominates $\mathbf{v}_2$, $false$ o.w.[1]
new_arch(*archive*, $\mathbf{v}$) updates the archive with $\mathbf{v}$
$\pi(\delta, temp, \mathbf{v}^p, \mathbf{v}^c)$ function determining acceptance rule
rand$(0, 1)$ returns a random number from the range $[0, 1)$
new_temp(*temp*) returns an updated temperature
terminate() terminating condition

```
archive ← init_arch()
for k = 0 to k = K − 1 do
  new_cons(k)
  temp ← init_temp()
  v^p ← init(k)
  new_arch(archive, v^p)
  repeat
    for i = 1 to i = N_rep do
      v^c ← local(v^p)
      if domoreq(v^p, v^c) == false then
        new_arch(archive, v^c)
      end if
      if π(δ, temp, v^p, v^c) > rand(0, 1) then
        v^p ← v^c
      end if
    end for
    temp ← new_temp(temp)
  until (terminate() = true)
end for
```

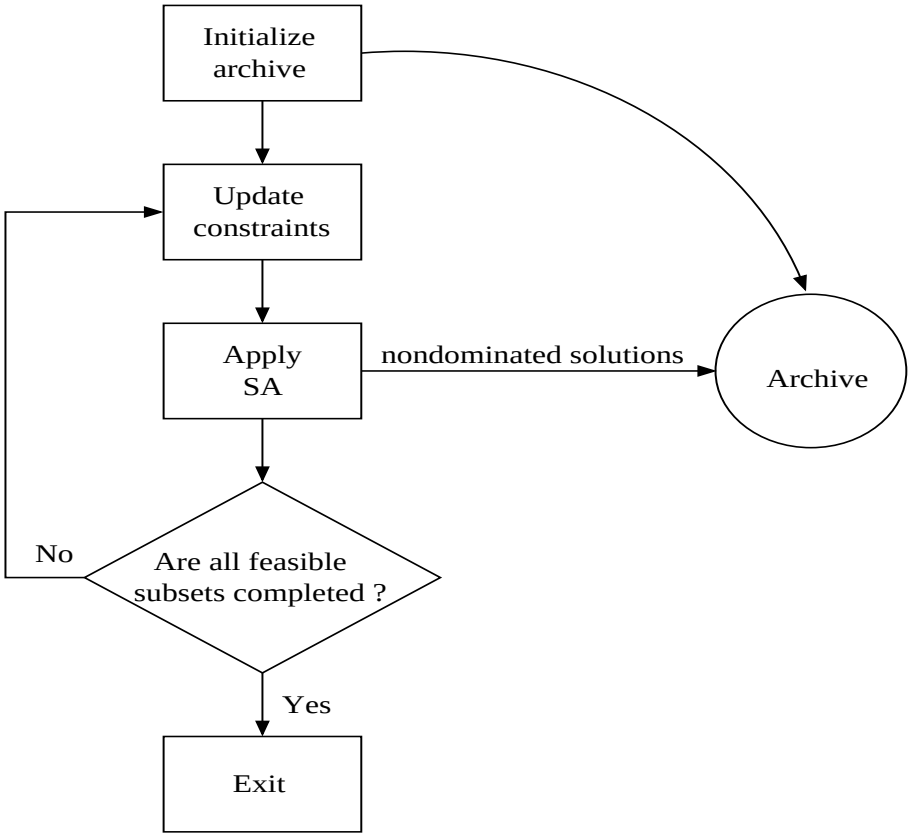

Figure 5.3: The chart displays the basic structure of the framework.

acceptance rule used here utilizes the weighted sum of the objective functions, it does not prevent the algorithm to visit the solutions at the non-convex part. Moreover, an acceptance rule based on the weighted Chebyshev norm coincides with a strong acceptance criterion, as shown previously in Chapter 4. In the case study carried out at the end of Chapter 3, we have observed that the trade-off curves are mostly convex. These issues motivate us for the selected acceptance rule. The acceptance rule is illustrated graphically in Figure 5.4.

5.3.3 Archiving Strategy

For the ease of maintenance, the general archive is divided into some sub-archives according the number of LSPs utilized by the solutions. It is reasonable to limit the number of LSPs in the final set by giving an upper-bound (shown with max_lsp). Thus, the solutions whose number of utilized LSPs is larger than this upper-bound are not accepted into the archive. Since the minimum number of LSPs possible to use in a solution can not be larger than the number of traffic requests (shown with min_lsp),

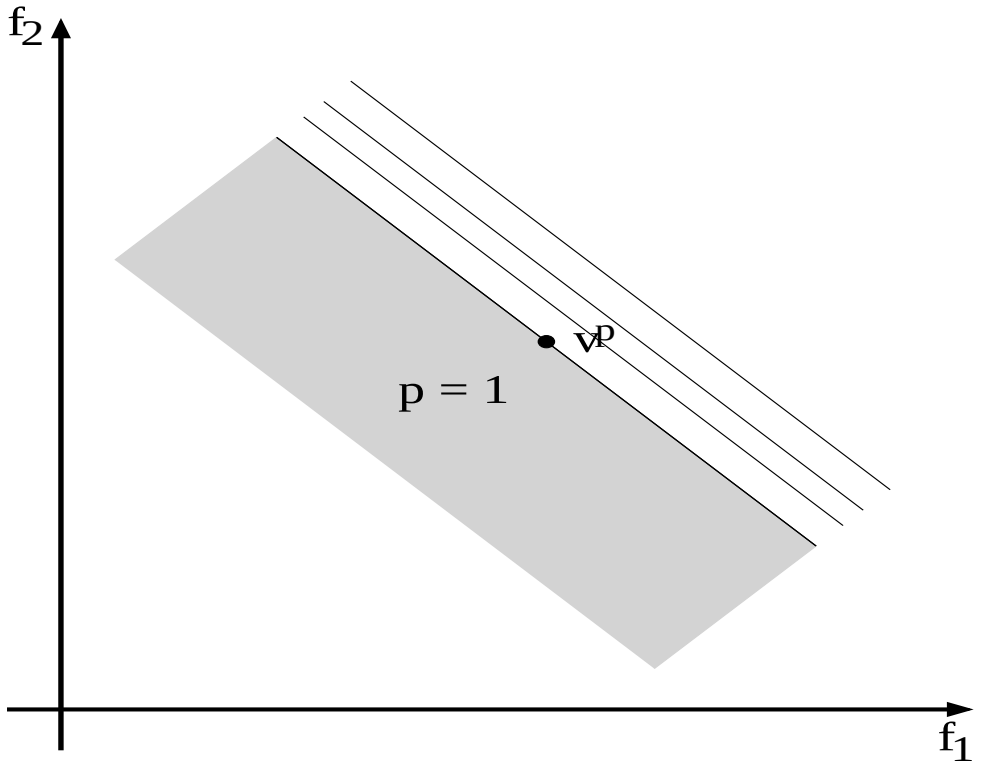

Figure 5.4: The acceptance rule and the iso-curves are illustrated graphically for the case of $\delta_1 = \delta_2 = 0.5$.

the number of sub-archives, maintained in the algorithm, is equal to $max\_lsp - min\_lsp + 1$.

For any MOP problem, the set of efficient solutions can be very large. Most of the solutions visited during the algorithm run are compared with the whole archive in order to find out the dominance relation of the solution at hand with the solutions in the sub-archives. For the sake of efficiency, in our implementation a child solution is compared with the archive only if it is not weakly dominated by its parent.

Furthermore, the sizes of the sub-archives should be limited. For the quality of the final output, it is of enormous importance to obtain uniformly distributed solutions over a wide extent of the objective functions in each sub-archive. The archiving strategy in MOP has become a key issue with the current developments in multiobjective EAs. According to the investigations in Chapter 4, the archiving strategy based on the *crowding distance* is applied in our framework.

The definition of the crowding distance is illustrated in Figure 5.5. Crowding distance of a point is based on the average distance of its two neighbors on either side of this point along each of the objective functions [18]. Thus, crowding distance gives the average side-length of the cuboid shown in the figure. The points with extreme

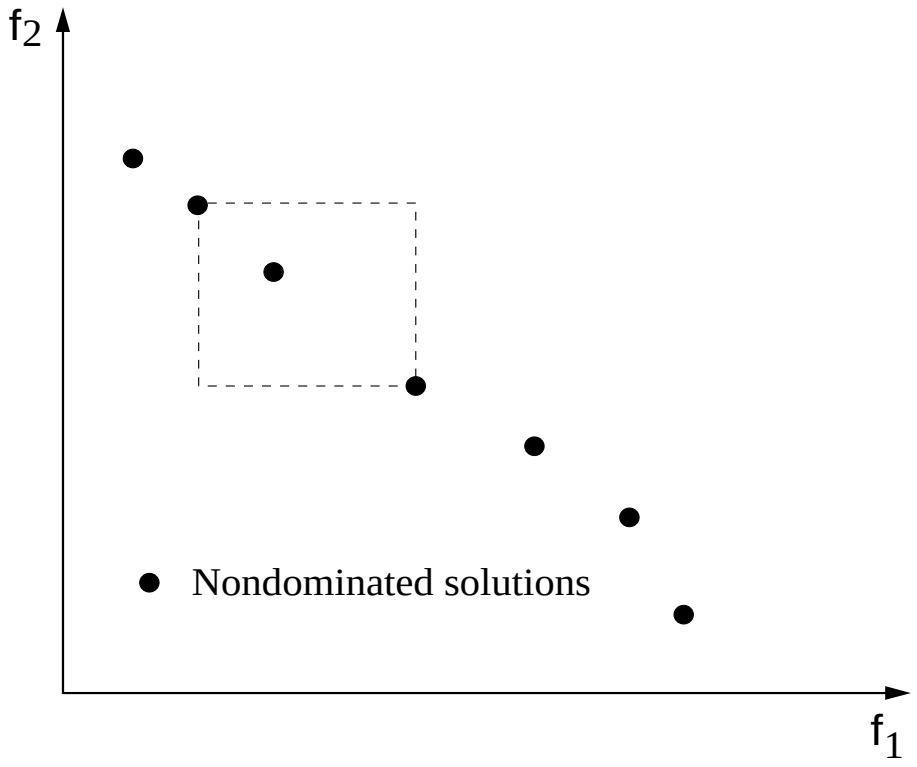

Figure 5.5: The crowding distance of a point is based on the distances to its neighbors along each of the objectives.

objective function values are given a crowding distance metric of infinity to keep the extent within a sub-archive at best.

Calculating the crowding distances requires an already sorted set of nondominated solutions. In our implementation, the order of the solutions in the sub-archives are always preserved. When the algorithm comes across with a new nondominated solution, this solution is inserted into the corresponding sub-archive by maintaining the order. If the addition of the solution causes the corresponding sub-archive to exceed its limit, the solution with the minimum crowding distance is deleted from the sub-archive.

The function $new\_arch(archive, \mathbf{v})$ used in the framework is described in detail below in Algorithm 5. The functions in the algorithm preserve the ordered structures of the sub-archives.

5.3.4 Biased Neighborhood Function

The experiments with the multiobjective traffic engineering problem have shown that it is more difficult to obtain solutions utilizing fewer LSPs. Therefore, the standard

Algorithm 5 *new_arch*$(archive, \mathbf{v})$

Data: *subarch* sub-archive corresponding to $\mathbf{v}$
Functions: *compare*$(archive, \mathbf{v})$ deletes solutions in *archive* dominated by $\mathbf{v}$ & returns *true* if $\mathbf{v}$ is weakly dominated, $false$ o.w.
insert$(subarch, \mathbf{v})$ inserts $\mathbf{v}$ into the corresponding sub-archive
size$(subarch)$ returns *true* if the size limit is exceeded, $false$ o.w.
crowding_calculate$(subarch)$ calculates crowding distances
del_min$(subarch)$ deletes the member with min. crowding distance

```
if compare(archive, v) == false then
  insert(subarch, v)
  if size(subarch) == true then
    crowding_calculate(subarch)
    del_min(subarch)
  end if
end if
```

neighborhood function is slightly modified such that it is biased towards the solutions with fewer paths. The set of neighbor solutions of $\mathbf{v}^i = (\mathbf{x}^i, \mathbf{g}^i, \mathbf{\Phi}^i)$ is represented by NB^i, and can be separated into two groups:

$$NB_1^i = \{\mathbf{v}^j = (\mathbf{x}^j, \mathbf{g}^j, \mathbf{\Phi}^j) \in NB^i \mid |\mathbf{X}_{>0}^j| < |\mathbf{X}_{>0}^i|\},$$

$$NB_2^i = \{\mathbf{v}^j = (\mathbf{x}^j, \mathbf{g}^j, \mathbf{\Phi}^j) \in NB^i \mid |\mathbf{X}_{>0}^j| \geq |\mathbf{X}_{>0}^i|\}.$$

Here, $\mathbf{X}_{>0}^i = \{x_t^{i,\,l} | x_t^{i,\,l} > 0,\, t \in T,\, l \in \{1, \ldots, L_t\}\}$. Thus, NB_1^i includes the neighbors utilizing less paths than $\mathbf{v}^i$.

In order to implement the biased neighborhood function, the whole neighborhood of the point at hand has to be investigated. The members of NB_1^i can be built practically by the solutions which have an exiting basic variable from $\mathbf{x}^i$ and an entering nonbasic variable from the rest of the regular variables $(\mathbf{g}^i, \mathbf{\Phi}^i)$ and the slack variables. The rest of the neighborhood builds NB_2^i. This strategy is said to be practical, since the solutions collected according to the entering and leaving variables may cover NB_1^i as a subset in case of degeneracy. Degeneracy is a special case of basic feasible solutions encountered during simplex method. Degeneracy occurs when at least one of the basic variables is equal to its lower and upper bound.

The biased neighborhood function is implemented in a way such that to select a solution from $\widehat{NB_1^i}$ is given the probability θ. $\widehat{NB_1^i}$ and $\widehat{NB_2^i}$ approximate the sets NB_1^i and NB_2^i according to the strategy explained above, respectively. Algorithm 6 shows how the biased neighborhood function of the parent solution $\mathbf{v}^p$ is randomized.

Algorithm 6 *biased_local*$(\mathbf{v}^p, \theta)$

Data: θ probability of selecting a solution from NB_1^p
$\widehat{NB_1^p}$ set of neighbors approximating NB_1^p
$\widehat{NB_2^p}$ set of neighbors approximating NB_2^p
$\mathbf{v}^c$ child solution

Functions: $rand(0,1)$ returns a random number from the range $[0,1)$
$local_1(\mathbf{v})$ returns a random solution from $\widehat{NB_1}$
$local_2(\mathbf{v})$ returns a random solution from $\widehat{NB_2}$

if $(|\widehat{NB_1^p}| > 0) \wedge (rand(0,1) < \theta)$ **then**
  $\mathbf{v}^c \leftarrow local_1(\mathbf{v}^p)$
else
  $\mathbf{v}^c \leftarrow local_2(\mathbf{v}^p)$
end if
return $\mathbf{v}^c$

The biased neighborhood method is not very practical due to its high computation overhead. To select a neighbor solution, it requires to investigate all the solutions in the neighborhood. The latter increases dramatically with the size of the problem. To overcome this drawback, we have utilized the *sampling method* which only takes into account a sample of the neighborhood. The method investigates only the randomly selected sample of the neighbor solutions for the biased application. Replication of the same solution in the sample is allowed for the ease of implementation. The size of the sample is set to 100 in all of the runs of our experiments.

5.4 Experiments

In the experiments, we are mainly interested in the performance of the biased neighborhood application via the sampling method. The sampling method is compared with the algorithm where no biasing is applied, which is called the *unbiased method* in this study. The algorithms are coded in C, and *Mersenne Twister* [57], an open-source software to generate random numbers, is integrated into the code. For the neighborhood structure, the shareware optimizer, Cplex Callable Library 6.6, is utilized [1]. Since simulated annealing is a probabilistic algorithm, 5 runs each with a different seed of the random number generator have been carried out for each problem and algorithm.

Experimental problems

The experiments are carried out for four different problems. The attributes of the problems are given in Table 5.1. The number of routers, directed links, traffic requests are directly related to the size of the problem. The last two columns in the table give the number of columns and rows of the relaxed LP set. The rows include also the constraints specifying the bounds on the total routing cost (see Constraints (5.13), and (5.14)). In order to decrease the size of the problem, the set of decision variables represented by $\mathbf{g}$ are removed from the problem. Constraints (3.5) are integrated into Constraints (3.7)-(3.12). The upperbound on g_m is restated as an upperbound on ϕ_m. Thus, the numbers in the fourth column of the table exclude g_m for all $m \in E$.

Problem A is the case study investigated in Chapter 3. The other topologies are generated with GT-ITM [86], an open-source software developed especially for modeling Internet topology. The capacities of the links are assigned equally in all of the problems. In problem A, the traffic requests are allowed to use paths with at most 4 hops, whereas in the last two problems they can use paths limited to 5 hops. The bandwidth demands of the traffic requests are also generated randomly. Two levels of demand ranges are assumed for the problems; one has with lower limits and the other one with higher limits. The demands are selected randomly from the corresponding ranges. The traffic demand is assumed to be symmetric, which means if there exists a demand from the vertex i to vertex j, then there is also a demand from vertex j to i, and both of the demand values are generated randomly from the same range.

Table 5.1: The experiments are carried out for 4 problems of different sizes.

	# Nodes	*# Links*	*# Requests*	*# Columns*	*# Rows*
Prb. A	10	32	90	292	284
Prb. B	14	42	84	320	338
Prb. C	15	44	94	388	360
Prb. D	30	130	132	1832	914

Algorithm parameters

This section discusses the parameters of simulated annealing that are fine tuned in the problems. The experiments are based on an initial accepting probability instead of an initial temperature. We start with the temperature equal to 1 and increase it according to the 10 initial solutions. The absolute differences of the objective function values are taken into account during this heating process. The temperature is increased in these iterations to guarantee an acceptance probability of 0.5 based

on the absolute value differences. N_{rep} is set to 250 and the control parameter *temp* is gradually decreased. We use a static reduction function $temp = 0.95temp$. Finally, the search stops when *temp* drops under a certain value ($temp_{stop} = 0.1$). The pre-studies have shown that these values supply a good compromise between the quality of the output and the run-time.

Table 5.2 give the parameters which are specific to the problems at hand and are determined by the decision maker. The first column gives the lower and upper bounds on the number of utilized paths that are acceptable to the archive. The second and third column show the minimal routing cost and the largest bound on the routing cost, respectively. Δ_s in the last column specifies the increase in the righthandsides of the constraints for the total routing cost (see Constraints (5.13), and (5.14)). Different load balancing functions are used for each problem. The number of break points in the load balancing functions is set 5 for all of the problems. However, their break points are at different utilization rates, depending on the density of the traffic demand matrix. The values of these parameters are not very critical in this study, they are kept same for each algorithm variant. We are here mostly interested in the performance comparisons of the algorithms under same conditions.

Table 5.2: The following parameters specific for each problem are used in the standard runs.

	min_lsp – max_lsp	*Minimum Routing Cost*	*Largest Bound on Routing Cost*	Δ_s
Prb. A	90–96	843.5	867.5	4.0
Prb. B	84–90	1566.4	1580.4	2.0
Prb. C	95–112	2881.6	2913.6	4.0
Prb. D	132–170	2872.2	3016.2	24.0

It is interesting to note here that, in Problem *C* *min_lsp* is 95 which is larger than the number of traffic requests. None of the runs carried out in this section were able to obtain any solution which uses 94 paths. Thus, we conclude that the constraint imposing a single path for each traffic request would be infeasible for this problem.

Performance assessment metrics

As stated in Chapter 4, the $\mathcal{D}$-metric is used to compare the nondominance relation of the output sets with each other. Since the calculation of the $\mathcal{S}$-metric is computationally expensive and complex for problems with 3 objectives, we calculated it independently for the subsets of solutions which utilize the same number of LSPs. The calculations are based on the routing and load balancing costs for each subset.

Two kinds of $\mathcal{S}$-metric are used in the studies. The first one calculates the weighted average of the $\mathcal{S}$-metrics, shown as $\mathcal{WS}$. The weight increases, as the number of utilized LSPs decreases.

$$\mathcal{WS} = \frac{\sum_{t=min\_lsp}^{max\_lsp} w(t)\mathcal{S}(t)}{\sum_{t=min\_lsp}^{max\_lsp} w(t)}$$

where $w(t)$ is the weight for the set of solutions using t LPSs. $\mathcal{S}(t)$ is the $\mathcal{S}$-metric value of the set of solutions using t paths and it is calculated according to the first two objective functions. $w(max\_lsp)$ is set to 1, and, $w(t) = w(t+1) + 1$.

The second type of the $\mathcal{S}$-metric is defined to evaluate the algorithms for the cases where few paths are used. It takes the average of the $\mathcal{S}$-metric values for the subsets of the solutions which uses paths less than or equal to t^*. t^* is selected according to the size of the problem. This metric is represented with $\mathcal{AS}(t^*)$.

Higher values of $\mathcal{WS}$ and $\mathcal{AS}$-metrics indicate algorithms with better quality. Both kinds of $\mathcal{S}$-metrics are calculated according to the normalized objective function values. The reference points are set equal to the maximum objective function values existing in the output sets which are compared. Since a set of exact solutions with 59 members is available for Problem A, the values of $\mathcal{WS}$ and $\mathcal{AS}$-metrics are based on the rates of these metrics of the output sets and the exact set.

The effect of the weight vector in the acceptance criterion

In this subsection, we analyze the effect of the weight vector in the acceptance rule of the sampling method. The biasing factor θ is set to 0.6 in all of the runs of this section. Three different weights are compared with each other. The first vector gives almost equal weights to all of the objective functions. In the second one, the least importance is given to the routing cost and the rest is shared equally between the minimization of the number of used paths and the minimization of the total load balancing cost. The third weight vector focuses mostly on the load balancing cost:

$$\begin{aligned} \delta^1 &= (0.33, 0.33, 0.34), \\ \delta^2 &= (0.05, 0.475, 0.475), \\ \delta^3 &= (0.05, 0.9, 0.05). \end{aligned}$$

For the ease of representation, any variant of the sampling method is denoted with the following notation: $[s, \theta, \Delta, \delta]$. Through the rest of the chapter, the following abbreviations are used:

$$\begin{aligned} s1 &\leftarrow [s, 0.6, \Delta_s, \delta^1], \\ s2 &\leftarrow [s, 0.6, \Delta_s, \delta^2], \\ s3 &\leftarrow [s, 0.6, \Delta_s, \delta^3]. \end{aligned}$$

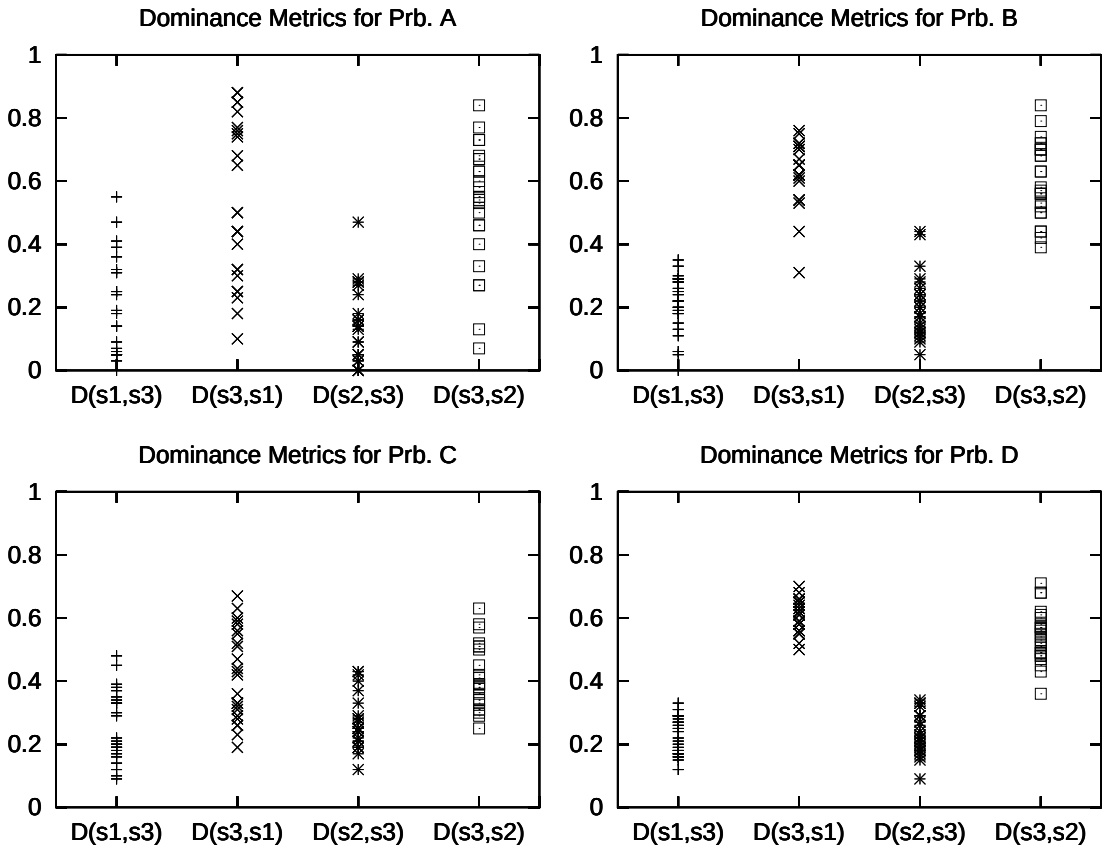

Figure 5.6: The figure compares the performances of three different weight vectors in the acceptance rule of the sampling method in terms of the dominance metric. Since 5 runs are carried out for each algorithm variant, there exist 25 pairwise comparisons. y-axes show two way $\mathcal{D}$-metric values of $s3$ with $s1$ and $s2$. Each graph corresponds to a different problem as it is stated in the titles.

Figure 5.6 shows the dominance metric values of $s3$ with $s1$ and $s2$. Since there are 5 output sets for each algorithm variant, 25 metric values are depicted in each case. Each graph shows that the output sets obtained by $s3$ tend to outperform the outputs of $s1$ and $s2$ in terms of the dominance metric. The results obtained in Problem D are especially significant.

Figure 5.7 and 5.8 analyze the algorithms in terms of the $\mathcal{S}$-metric. For Problem A and B $\mathcal{AS}(3)$-metrics are compared, whereas for Problem C $\mathcal{AS}(5)$, and for Problem D $\mathcal{AS}(10)$ values are calculated. The values in parentheses are selected according to the sizes of the problems. The graphs depict that three kinds of the sampling algorithm are difficult to compare in terms of $\mathcal{WS}$-metric. However, according to

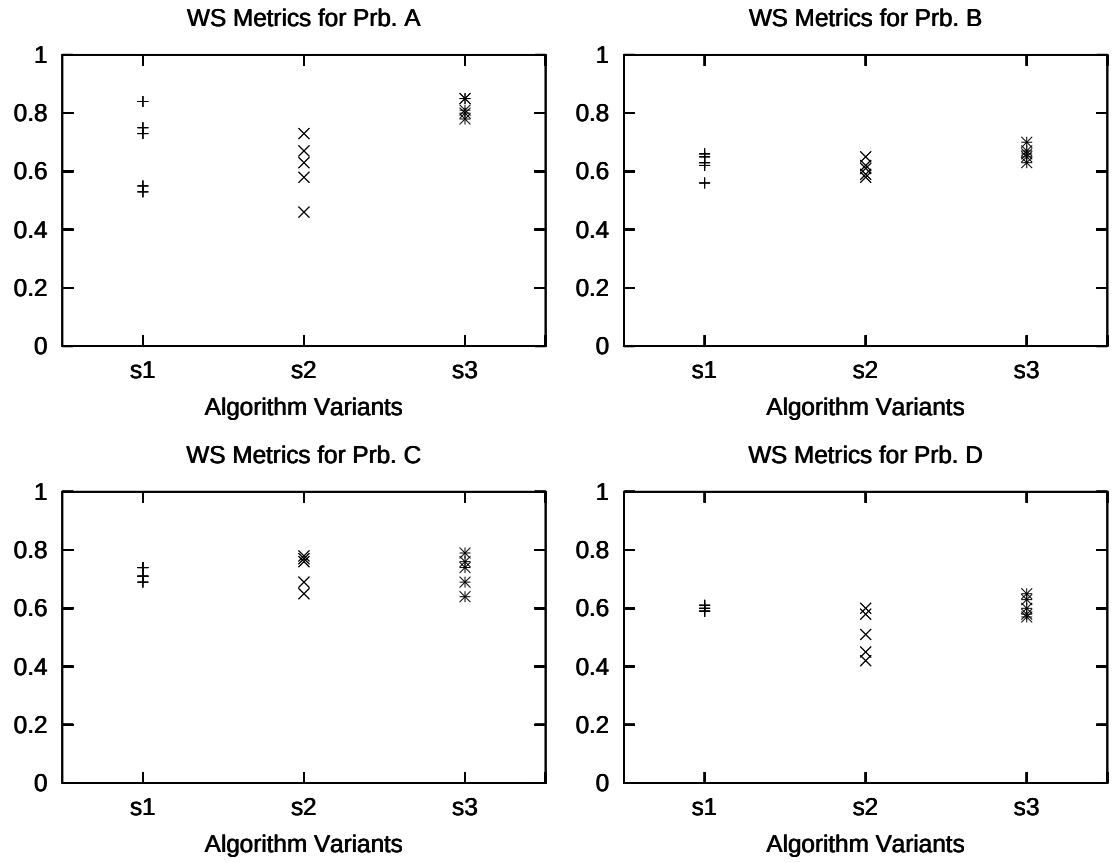

Figure 5.7: The $\mathcal{WS}$-metric values of the sampling method variants are depicted in the figure. Higher values indicate output sets with better quality.

Figure 5.8, $s3$ generally performs better when the number of utilized LSPs is low. Hence, we have obtained better outputs with the sampling method which emphasizes the total load balancing cost. The reasoning behind this observation can be explained as follows: The sampling method takes the minimization of the total routing cost into consideration through Constraints (5.13), and (5.14). The application of the biased neighborhood approach directs the search to the solutions where fewer paths are used. Hence, a sampling method with a weight vector focusing mostly on the load balancing cost results with better outputs, especially in terms of the $\mathcal{D}$ and $\mathcal{AS}$-metrics.

Comparison of the sampling method with the unbiased method

The analyses in this section compare the sampling method with the unbiased method. Another interest is also in the effect of θ on the quality of outputs. Thus, the following

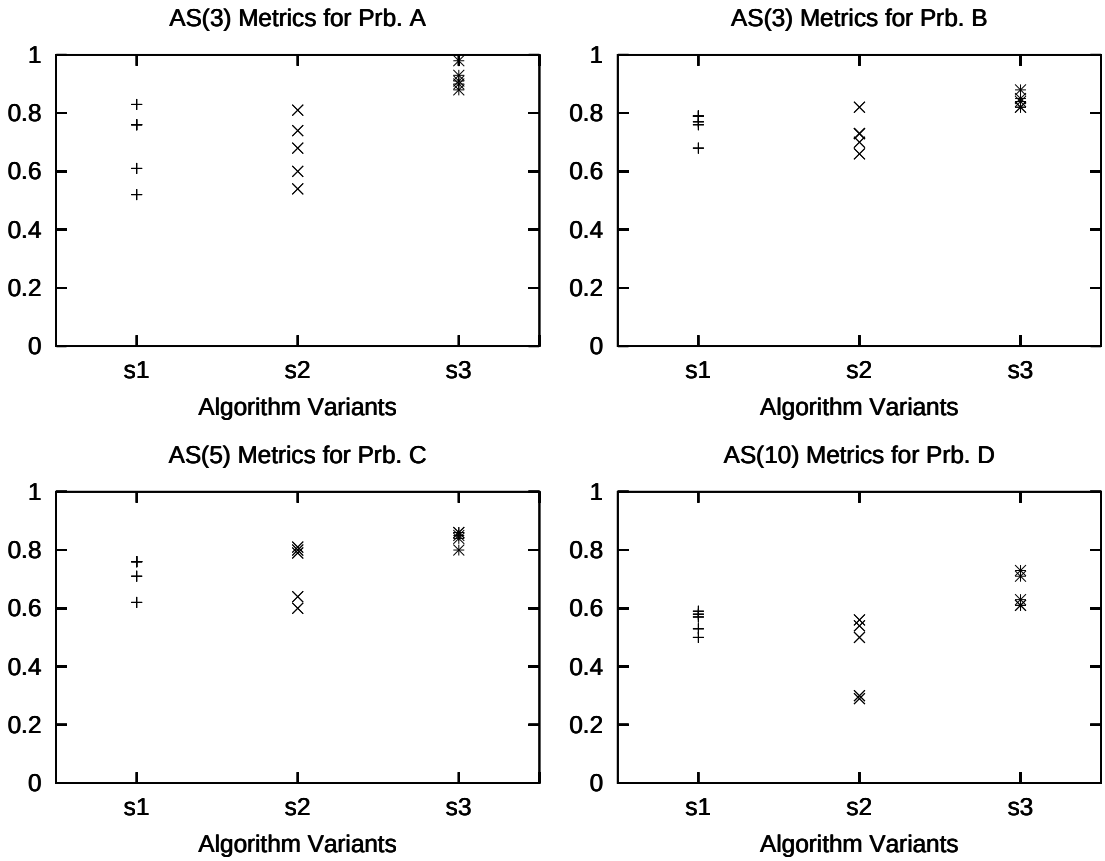

Figure 5.8: The variants of the sampling method are compared in terms of $\mathcal{AS}$-metric. The $\mathcal{AS}(3)$-metric values for Problem A and B, the $\mathcal{AS}(5)$-metric values for Problem C, and the $\mathcal{AS}(10)$-metric values for Problem D are shown in the figure. Higher values indicate output sets with better quality.

types of the sampling method are compared with the unbiased method in this section.

$$\begin{aligned} s3 &\leftarrow [s, 0.6, \Delta_s, \delta^3], \\ s4 &\leftarrow [s, 0.4, \Delta_s, \delta^3], \\ s5 &\leftarrow [s, 0.2, \Delta_s, \delta^3]. \end{aligned}$$

In the pre-studies with the unbiased method, we have performed some initial runs with a weight vector of δ^3. The results have shown us that the unbiased method may fail to find solutions utilizing few paths, especially for Problem D which is of large size. Thus, the standard unbiased method requires multiple runs with different weight vectors which give more importance to the objective function considering the minimization of the number of used paths. We repeated the unbiased method

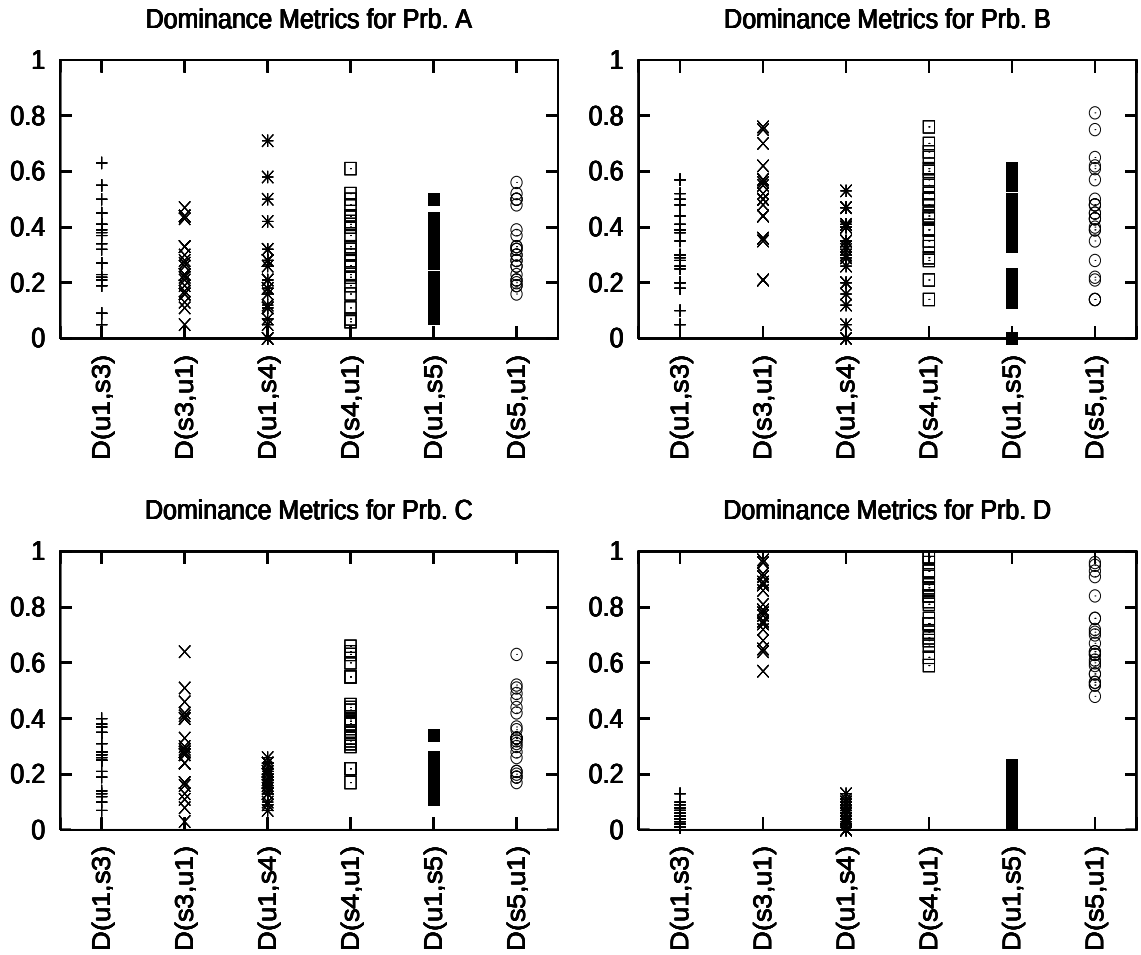

Figure 5.9: The figure compares the dominance relations of various sampling algorithms with the unbiased method.

with different weight vectors, and a single archive is maintained during all of the repetitions. In the weight vectors, the routing cost is given the least importance. The unbiased method is repeated for the following 4 weight vectors and a single output set is obtained:

$$\begin{aligned} \delta^3 &= (0.05, 0.9, 0.05), \\ \delta^4 &= (0.05, 0.62, 0.33), \\ \delta^5 &= (0.05, 0.34, 0.61), \\ \delta^6 &= (0.05, 0.06, 0.89). \end{aligned}$$

A variant of the unbiased method is denoted with $[u, \Delta]$. $u1 = [u, \Delta_s]$ represents the

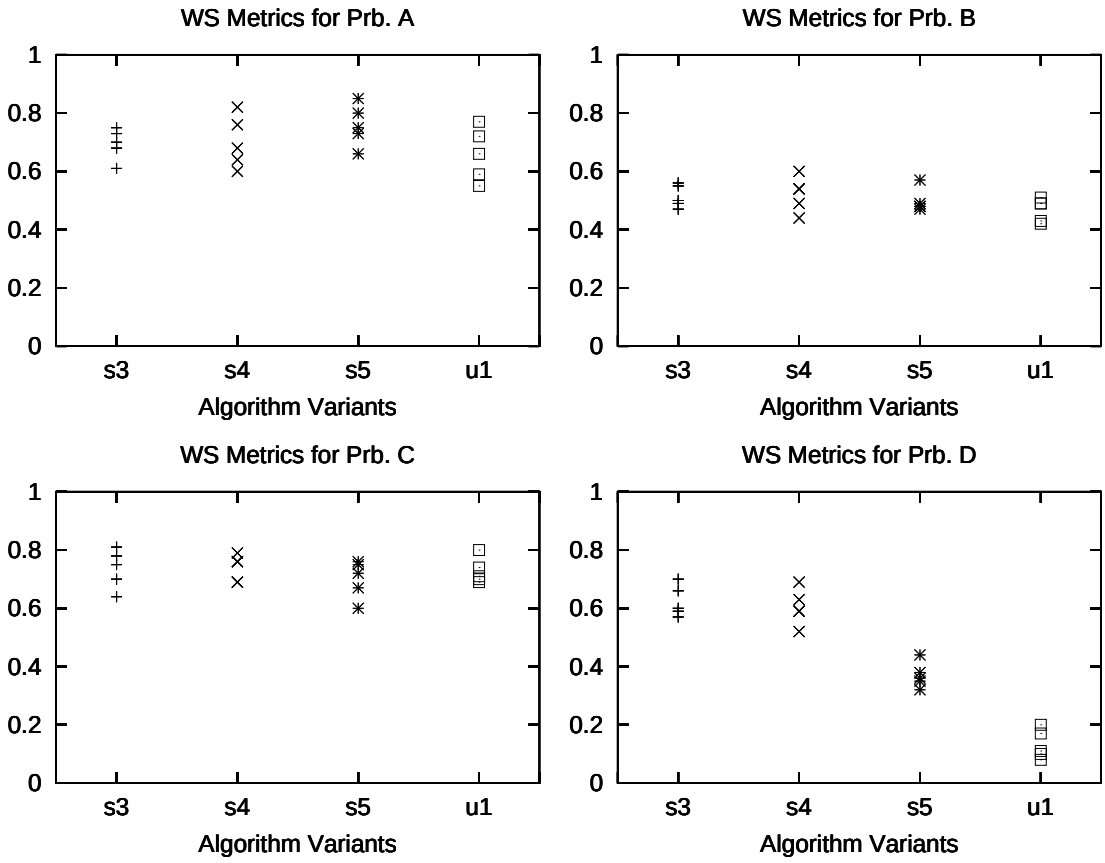

Figure 5.10: The $\mathcal{WS}$-metric values of the algorithms applying the sampling method with different θ values and the algorithm applying the unbiased method are depicted in the figure.

standard unbiased method applied in this section.

Since Problem A was previously solved by an exact algorithm, the nondominated sets obtained in this section are compared with the exact set in terms of $\mathcal{D}$-metric. In the average, the exact set dominates 82% of the outputs of $s3$, 73% of the sets of $s4$, 71% of the outputs of $s5$, and 71% of the sets of $u1$. Apart from that, the approximate sets obtained by the algorithm variants are compared also with each other in terms of their dominance relations, and similar results are obtained.

Figure 5.9 plots the pairwise dominance metric values of the sampling method variants with the unbiased method. According to the results, for problem A, $u1$ tends to perform slightly better than the sampling methods (only $s5$ performs almost equally as $u1$). In the other problems, all kinds of the sampling method generally perform better than the unbiased method. Particularly, in problem D, the difference is sig-

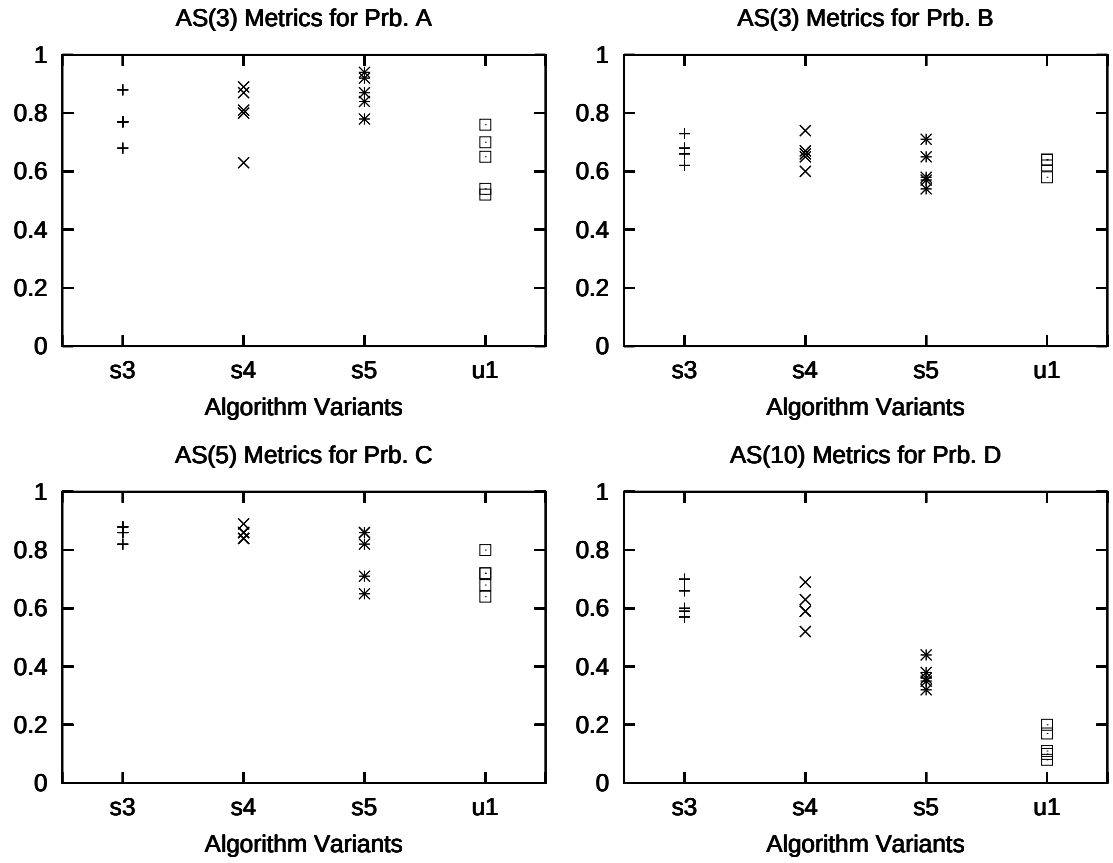

Figure 5.11: The $\mathcal{AS}$-metric values of the algorithms applying the sampling method with different θ values and the algorithm applying the unbiased method are compared. The $\mathcal{AS}(3)$-metric values for Problem A and B, the $\mathcal{AS}(5)$-metric values for Problem C, and the $\mathcal{AS}(10)$-metric values for Problem D are plotted.

nificant. The differences between the performances of the variants of the sampling method are not substantial.

In Figure 5.10 and 5.11, the $\mathcal{WS}$ and $\mathcal{AS}$-metric values of the output sets are plotted. The reference points used in the $\mathcal{S}$-metric calculations are selected differently at each type of analysis. The selection is based on the output sets at hand. Thus, the metric values of $s3$ are different from the values in the previous subsection. According to the figures, in the first three problems the performance of the unbiased method in terms of $\mathcal{WS}$ and $\mathcal{AS}$-metrics does not differ dramatically from the performances of all sampling methods. However, in the last problem both $s5$ and $u1$ are outperformed significantly by $s3$ and $s4$. Furthermore, $s5$ performs better than $u1$. These results

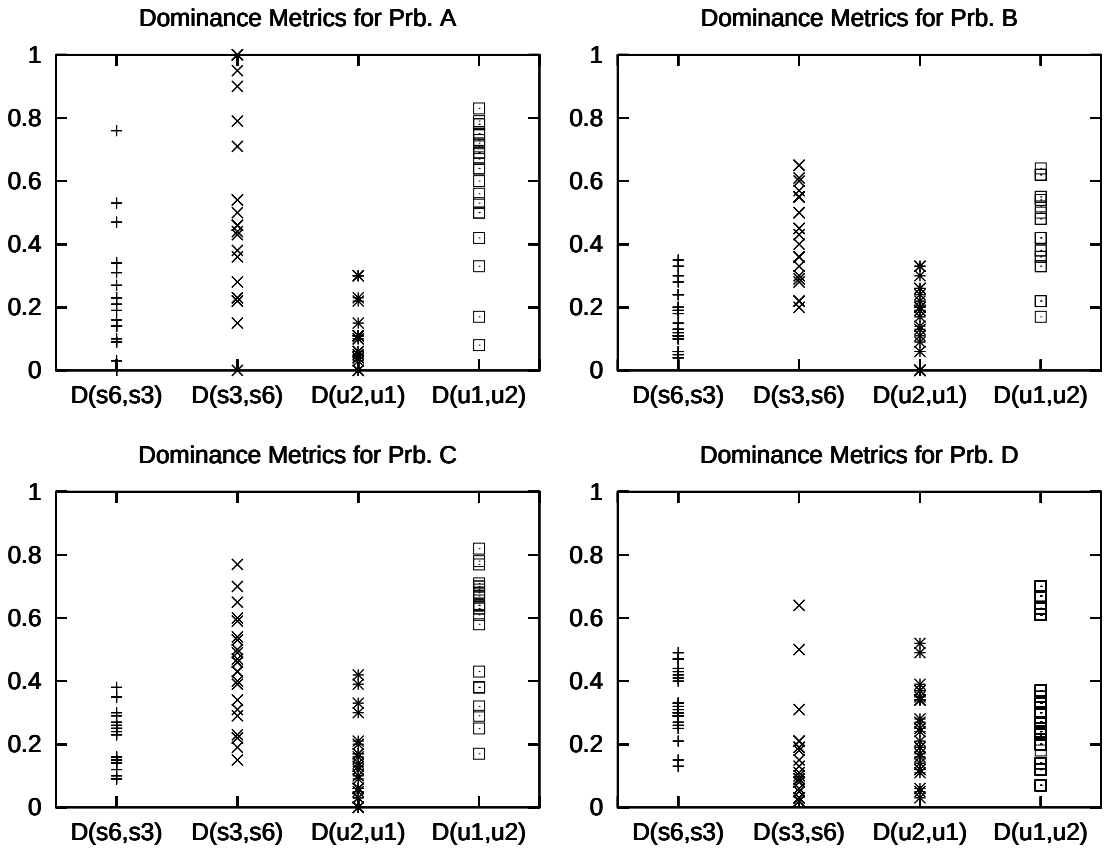

Figure 5.12: The effect of an increase in Δ is investigated in terms of the dominance metric.

suggest that the sampling method generally ends up with better outputs, whereas one has to be careful with the selection of θ.

The sampling method is also compared with the unbiased method in terms of run times. A set of exact solutions is available for Problem A, which gives us an idea how efficient the approximate algorithms are. All of the runs are carried out in an unloaded Alphaev6-dec-osf5.0 machine with two processors at 667 MHz with 1994 Mbyte memory having Digital UNIX V5.0 as the operating system. For Problem A, it required about 108[2] minutes to run the algorithms with the weights corresponding to the solutions in the exact set (59 members). Actually, the total run time for the experiments with the exact method took much longer. Same solutions were obtained with different weight vectors. In the experiment with the uniformly distributed weight

[2]These values are based on the two step application of the lexicographic weighted Chebyshev method.

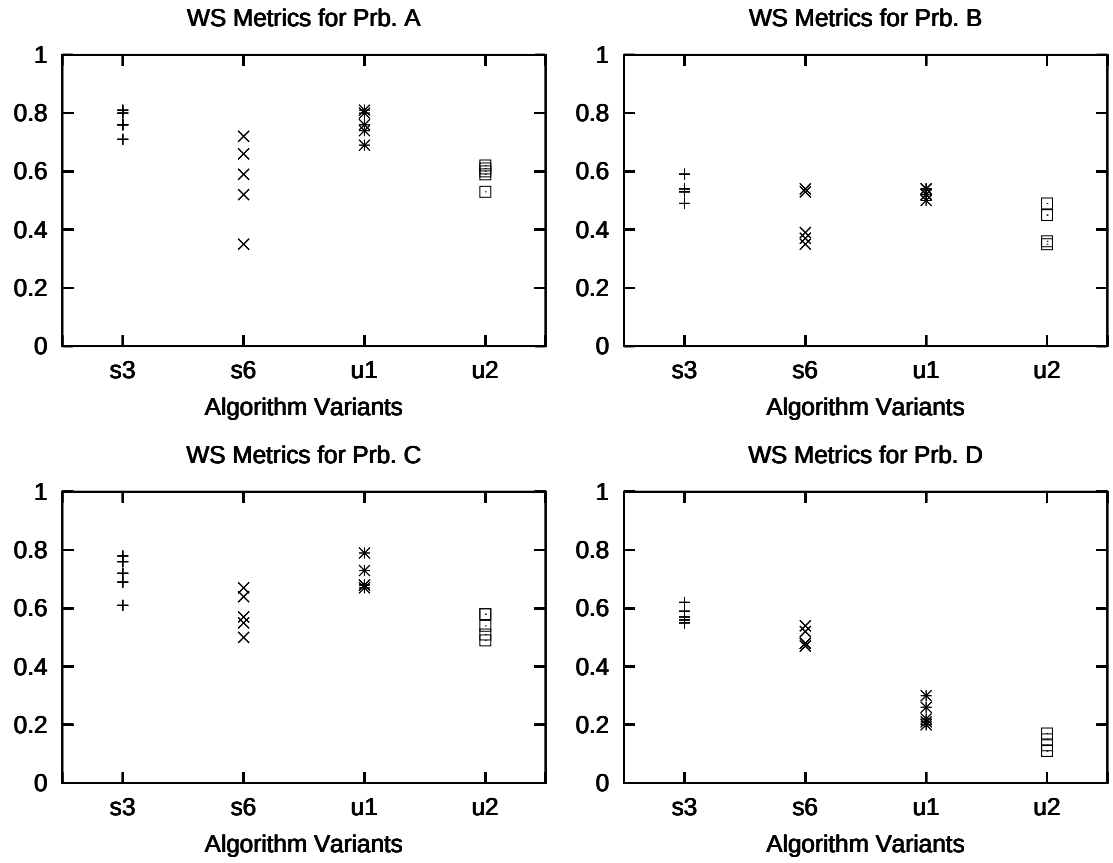

Figure 5.13: The $\mathcal{WS}$-metric values of the algorithms applying the sampling and unbiased method with different Δ values are depicted in the figure.

vectors, we have seen that it takes about 7.2 minutes for a run of the exact method. Table 5.3 lists the average run times of $s3$ and $u1$ in minute scale. For Problem A, the average run times of $s3$ and $u1$ are much smaller than the total run time necessary for the exact approach. $u1$ usually takes less time than $s3$, except the first and last problem. It is possible to decrease and increase the run time of the unbiased method by changing the number of repetitions with different weight vectors. The run time of the sampling method depends also on the size of the sample.

The effect of Δ on the performances of the algorithms

The analyses in this section aim at finding out how the quality of the solutions are affected, when the value of Δ is increased. It is expected that, an increase in Δ will end up with outputs of worse quality, nevertheless it will have a positive effect on the

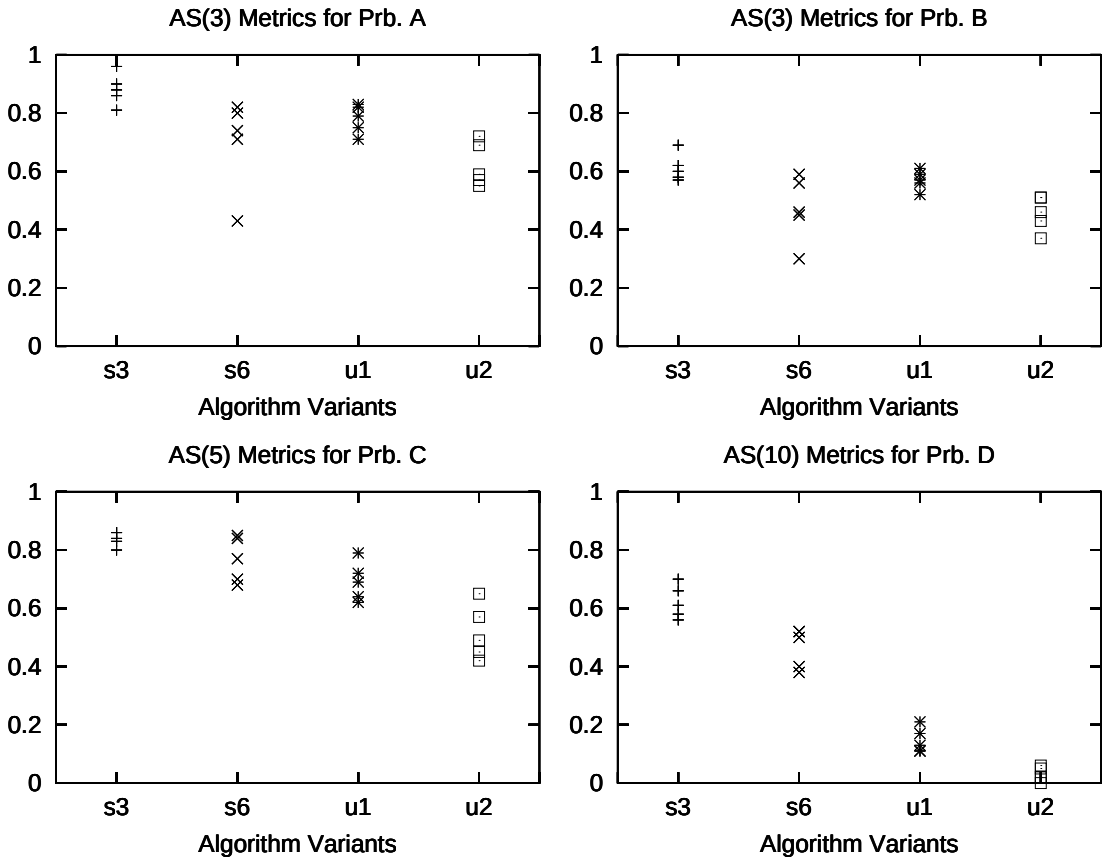

Figure 5.14: The $\mathcal{AS}$-metric values of the algorithms applying the sampling and unbiased method with different Δ values are compared. The $\mathcal{AS}(3)$-metric values for Problem A and B, the $\mathcal{AS}(5)$-metric values for Problem C, and the $\mathcal{AS}(10)$-metric values for Problem D are plotted.

Table 5.3: The average run times (in minutes) of $s3$ and $u1$ are given. The algorithms are executed on an unloaded machine.

	$s3$	$u1$
Prb. A	6.9	8.7
Prb. B	14.0	9.3
Prb. C	17.7	12.8
Prb. D	19.0	23.8

run times. The updated Δ values are shown in Table 5.4.

Table 5.4: The effect of an increase in Δ is analyzed.

	Prb. A	*Prb. B*	*Prb. C*	*Prb. D*
Δ_u	8.0	3.5	8.0	48

In the analyses, the following algorithms are compared with $s3$ and $u1$.

$$\begin{aligned} s6 &\leftarrow [s, 0.6, \Delta_u, \delta^3], \\ u2 &\leftarrow [u, \Delta_u]. \end{aligned}$$

Figure 5.12 depicts the pairwise dominance relations of $s3$ with $s6$, and $u1$ with $u2$. In the plots it is observed that the dominance power of the nondominated sets decreases, when Δ is increased. This effect is not very clear in Problem D. The reason may be due the scaling of the objective functions. As stated before, the scaling of the routing cost depends on Δ (see Equation (5.15)).

Figure 5.13 and 5.14 show the $\mathcal{WS}$ and $\mathcal{AS}$-metric values, respectively. According to the figures, the increase in Δ affects negatively the quality of the nondominated sets in terms of these metrics. Especially for Problem D, $u2$ almost fails to find solutions which use few paths (between 132 and 139).

Apart from these, the run times of $s6$ and $u2$ are investigated. The average run times are given in Table 5.5. As expected, the run times are decreased as Δ is increased. Thus, there is a trade-off between the size of Δ and the run times.

Table 5.5: The average run times (in minutes) of $s6$ and $u2$ are given. The algorithms are executed on an unloaded machine.

	$s6$	$u2$
Prb. A	4.6	4.4
Prb. B	9.4	5.7
Prb. C	10.3	6.7
Prb. D	9	11.9

6 Conclusion

In this thesis, the multiobjective LSP-design problem in MPLS networks has been investigated in detail. MPLS networks have received great interest during the past few years due to their flexibility and efficiency in traffic engineering.

In the multiobjective off-line traffic engineering problem, three objectives were taken into consideration: minimal total routing cost, optimal load balancing in the network, minimal splitting of the traffic requests. The first objective is traffic oriented, whereas the second objective is network oriented. The last objective is founded on both a network and customer perspective. We have shown in a case study that all of these objectives may be in strong conflict. Being aware of the trade-offs among the objectives will help the network managers and the ISPs to realize the effects of the decisions taken.

We have seen that exact solution approaches are inconvenient for this multiobjective mixed zero-one integer programming problem due to its size and complexity. We have proposed a heuristic framework using simulated annealing as the search strategy. The algorithm works on the linear programming relaxation of the original problem. The neighborhood structure to make moves in the search space is based on the pivoting steps of the simplex method.

Variant strategies within this framework have been investigated and compared with each other. We have especially focused on two kinds of neighborhood function: biased towards the solutions which use less paths versus unbiased. The experiments carried out in Chapter 5 have shown that methods based on biasing generally perform better than the unbiased method. The heuristic variants are also compared in terms of running times. There were no large differences between the running times of the biased and unbiased approaches, whereas both perform much better than the exact solution approach.

The future research following this thesis may be developed mainly in the following ways:

- The mathematical model can be updated to take different aspects of networks into consideration. Especially, different load balancing functions can be introduced and its effect on the network performance can be compared with the one currently existing in the model. It is also possible to remove the minimization of traffic splitting from the model. When the traffic is not allowed to be split,

the problem will be combinatorial. More effective solutions approaches can be developed for this reduced problem.

- In this thesis, some heuristic algorithms based on only simulated annealing are investigated. It will be highly interesting to compare the performance of simulated annealing for the off-line traffic engineering problem with other types of heuristics. Heuristics based on tabu search, evolutionary algorithms, etc. and the application of different types of neighborhood structure can be compared with the algorithms proposed in this thesis.
- We have observed that the application of the biased neighborhood structure results in better outputs. The effect of biased neighborhood approach can also be investigated in other types of MOP problems, e.g., multiobjective knapsack, packing, and nonlinear problems.

Bibliography

[1] Cplex 6.6. http://www.ilog.com/.

[2] E. H. L. Aarts and J. H. M. Korst. *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Wiley, Chichester, 1989.

[3] F. B. Abdelaziz and S. Krichen. A tabu search heuristic for multiobjective knapsack problems. Technical report, RUTCOR Research Group, Rutgers University, New Jersey, December 1997.

[4] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow. RSVP-TE: Extensions to RSVP for LSP tunnels. *RFC [1] 3209*, December 2001.

[5] D. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. McManus. Requirements for traffic engineering over MPLS. *RFC 2702*, September 1999.

[6] D. O. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. *RFC 3272*, May 2002.

[7] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York, 1996.

[8] S. Bessler. Label switched paths re-configuration under time-varying traffic conditions. In *Internet Traffic Engineering and Traffic Management, Proceedings of the* 15^{th} *ITC Specialist Seminar*, pages 33–38, Würzburg, Germany, 2002. Department of Distributed Systems, Institute of Computer Science, Bayerische Julius-Maximilians-Universität Würzburg.

[9] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for Differentiated Service. *RFC 2475*, 1998.

[10] A. V. Cabot and Jr. A. P. Hurter. An approach to zero-one integer programming. *Operations Research*, 16:1206–1211, 1968.

[1] Request for Comments

[11] V. Chvátal. *Linear programming*. A Series of Books in Mathematical Sciences. W. H. Freeman and Company, New York, 1983.

[12] S. E. Cieniawski, J. W. Eheart, and S. Ranjithan. Using genetic algorithms to solve a multiobjective groundwater monitoring problem. *Water Resources Research*, 31(2):399–409, February 1995.

[13] C. A. Coello Coello. A short tutorial on evolutionary multiobjective optimization. In E. Zitzler, K. Deb, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, pages 21–40, Berlin, 2001. Springer-Verlag.

[14] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. A Framework for QoS-based Routing in the Internet. *RFC 2386*, 12(3), 2000.

[15] I. Das and J. Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14(1):63–69, 1997.

[16] B. S. Davie and Y. Rekhter. *MPLS: Technology and Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2000.

[17] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Chichester, UK, 2001.

[18] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In *Parallel Problem Solving from Nature – PPSN VI, 6^{th} International Conference*, volume 1917 of *Lecture Notes in Computer Science*, pages 849–858, Paris, France, September 2000.

[19] R. P. Dick and N. K. Jha. MOGAC: A multiobjective genetic algorithm for hardware-software co-synthesis of distributed embedded systems. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, October 1998.

[20] M. Ehrgott. *Multicriteria Optimization*. Lecture Notes in Economics and Mathematical Systems 491. Springer-Verlag, 2000.

[21] M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22 (4):425–460, 2000.

[22] A. Elwalid, C. Jin, S. H. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings IEEE INFOCOM 2001*, pages 1300–1309, Anchorage, Alaska, April 2001.

[23] C. Erbas, S. C. Erbas, and A. D. Pimentel. A multiobjective optimization model for exploring multiprocessor mappings of process networks. In *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/ Software Codesign and System Synthesis (CODES + ISSS'03)*, pages 182–187, Newport Beach, USA, October 2003.

[24] S. C. Erbas. Utilizing evolutionary algorithms for multiobjective problems in traffic engineering. In W. Ben-Ameur and A. Petrowski, editors, *Proceedings of the International Network Optimization Conference (INOC) 2003*, pages 207–212, Evry-Paris, France, October 2003.

[25] S. C. Erbas and C. Erbas. A multiobjective off-line routing model for MPLS networks. In J. Charzinski, R. Lehnert, and P. Tran-Gia, editors, *Providing Quality of Service in Heterogeneous Environments - Proc. of the 18th International Teletraffic Congress (ITC-18)*, pages 471–480, Berlin, Germany, August-September 2003.

[26] S. C. Erbas and R. Mathar. An off-line traffic engineering model for MPLS networks. In *Proceedings of the* 27^{th} *Conference on Local Computer Networks LCN 2002*, pages 166–174, Tampa, Florida, November 2002. IEEE Computer Society.

[27] B. Fortz and M. Thorup. Internet traffic engineering by optimizing OSPF weights. In *Proceedings IEEE INFOCOM 2000 (2)*, pages 519–528, Tel-Aviv, Israel, 2000.

[28] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, New York, 1979.

[29] E. Gelenbe, R. Lent, and Z. Xu. Design and performance of cognitive packet networks. *Performance Evaluation*, 46:155–176, 2001.

[30] F. Glover. A note on linear programming and integer feasibility. *Operations Research*, 16:1212–1216, 1968.

[31] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, 1997.

[32] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, Mass, 1989.

[33] Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:296–297, 1971.

[34] M. P. Hansen. Tabu search in multiobjective optimization: MOTS. In *Trends in Multicriteria Decision Making: Proceedings of the 13th International Conference on Multiple Criteria Decision Making (MCDM'97)*, Cape Town, South Africa, 1997.

[35] J. Horn. F1.9: Multicriteria decision making and evolutionary computation. In T. Baeck, D. B. Fogel, and Z. Michalewicz, editors, *Handbook of Evolutionary Computation*. Institute of Physics Publishing, Bristol, UK, 1997.

[36] J. Horn, N. Nafpliotis, and D. E. Goldberg. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, volume 1, pages 82–87, Piscataway, NJ, 1994.

[37] J. Hromkovic. *Algorithmics for Hard Problems. Introduction to Combinatorial Optimization, Randomization, Approximation and Heuristics*. Texts in Theoretical Computer Science, An EATCS Series. Springer-Verlag, Berlin, first edition, 2001.

[38] G. Huston. *ISP Survival Guide: Strategies for running a competitive ISP*. John Wiley & Sons, Inc., USA, 1998.

[39] H. Ishibuchi and T. Murata. A multi-objective genetic local search algorithm and its application to flowshop scheduling. *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, 28(3), August 1998.

[40] B. Jamoussi, L. Andersson, R. Callon, R. Dantu, L. Wu, P. Doolan, T. Worster, N. Feldman, A. Fredette, M. Girish, E. Gray, J. Heinanen, T. Kilty, and A. Malis. Constraint-based LSP setup using LDP. *RFC 3212*, January 2000.

[41] A. Jaszkiewicz. Multiple objective metaheuristic algorithms for combinatorial optimization. Habilitation thesis, 360, 2001. Poznan University of Technology, Poznan.

[42] M. D. Johnston and H.-M. Adorf. Multi objective evolutionary optimization. *OR News*, pages 11–16, July 2003.

[43] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[44] J. Knowles, M. Oates, and D. Corne. Advanced multi-objective evolutionary algorithms applied to two problems in telecommunications. *BT Technology Journal*, 18(4):51–65, October 2000.

[45] J. D. Knowles. *Local-Search and Hybrid Evolutionary Algorithms for Pareto Optimization*. PhD thesis, University of Reading, Reading, UK, 2002.

[46] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the Pareto archived evolution strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[47] J. D. Knowles and D. W. Corne. M-PAES: a memetic algorithm for multiobjective optimization. In *Proceedings of the Congress on Evolutionary Computation (CEC00)*, pages 325–332, Piscataway, NJ, 2000. IEEE Press.

[48] M. S. Kodialam and T. V. Lakshman. Minimum interference routing with applications to MPLS traffic engineering. In *Proceedings IEEE INFOCOM 2000 (2)*, pages 884–893, Tel-Aviv, Israel, 2000.

[49] P. J. M. Van Laarhoven and E. H. L. Aarts. *Simulated Annealing: Theory and Applications*. Kluwer, Dordrecht, 1987.

[50] C. Lagoa and H. Che. Decentralized optimal traffic engineering in the internet. *SIGCOMM Computer Communication Review*, 30(4), October 2000.

[51] M. Laumanns, E. Zitzler, and L. Thiele. A unified model for multi-objective evolutionary algorithms with elitism. In *Proceedings of Congress on Evolutionary Computation (CEC00)*, pages 46–53, Piscataway, NJ, 2000. IEEE.

[52] M. Laumanns, E. Zitzler, and L. Thiele. On the effects of archiving, elitism, and density based selection in evolutionary multi-objective optimization. In E. Zitzler, K. Deb, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, pages 181–196, Berlin, 2001. Springer-Verlag.

[53] E. Lawler. *Combinatorial Optimization, Networks and Matroids*. Holt, Reinhart, and Winston, New York, 1976.

[54] S.-S. Leu and C.-H. Yang. GA-based multicriteria optimal model for construction scheduling. *Journal of Construction Engineering and Management*, 125:420–427, 1999.

[55] A. Løkketangen. Heuristics for 0 - 1 mixed-integer programming. In P. M. Pardalos and M. G. C. Resende, editors, *Handbook of Applied Optimization*. Oxford University Press,, New York, 2002.

[56] A. Løkketangen and F. Glover. Solving zero-one mixed integer programming problems using tabu search. *European Journal of Operational Research*, 106:624–658, 1998.

[57] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3–30, January 1998. http://www.math.keio.ac.jp/ matumoto/emt.html.

[58] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer-Verlag, Berlin, Heidelberg, 2000.

[59] D. Mitra and K. G. Ramakrishnan. A case study of multiservice, multipriority traffic engineering design for data networks. In *Proceedings IEEE GLOBECOM 99*, volume 1B, pages 1077–1083, Brazil, December 1999.

[60] J. Moy. OSPF Version 2. *RFC 2178*, July 1997.

[61] T. Murata, H. Ishibuchi, and H. Tanaka. Multi-objective genetic algorithm and its applications to flowshop scheduling. *Computers & Industrial Engineering*, 30(4):957–968, 1996.

[62] D.K. Nam and C. H. Park. Multiobjective simulated annealing: a comparative study to evolutionary algorithms. *International Journal of Fuzzy Systems*, 2(2):87–97, 2000.

[63] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons, New York, 1988.

[64] G. T. Parks and I. Miller. Selective breeding in a multiobjetive genetic algorithm. In A. E. Eiben, Thomas Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature - PPSN V,* 5^{th} *International Conference*, pages 250–259, Amsterdam, Holland, 1998.

[65] S. R. Ranjithan, S. K. Chetan, and H. K. Dakshina. Constraint method-based evolutionary algorithm (CMEA) for multiobjective optimization. In E. Zitzler, K. Deb, C. A. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, pages 299–313, Berlin, 2001. Springer-Verlag.

[66] C. Reeves. *Modern Heuristic Techniques for Combinatorial Problems*. Advanced Topics in Computer Science. McGraw-Hill, London, 1995.

[67] M. G. C. Resende and C. C. Ribeiro. A GRASP with path-relinking for private virtual circuit routing. *Networks*, 41:104–114, 2003.

[68] E. Rosen, A. Viswanathan, and R.Callon. Multiprotocol Label Switching Architecture. *RFC 3031*, January 2001.

[69] J. D. Schaffer. *Some Experiments in Machine Learning Using Vector Evaluated Genetic Algorithms*. PhD thesis, Vanderbilt University, Nashville, TN, 1984.

[70] S. Schnitter and G. Haßlinger. Heuristic solutions to the LSP-design for MPLS traffic engineering. In *Proceedings of the* 10^{th} *International Telecommunication Network Strategy and Planning Symposium (Networks 2002)*, Munich, Germany, June 2002.

[71] J. R. Schott. Fault tolerant design using single and multicriteria genetic algorithm optimization. Master thesis, 1995. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts.

[72] Martin Schütz and Hans-Paul Schwefel. Evolutionary approaches to solve three challenging engineering tasks. *CMAME (Computer Methods in Applied Mechanics and Engineering)*, 186:141–170, 2000.

[73] P. Serafini. Simulated annealing for multi objective optimization problems. In G. H. Tzeng, H. F. Wang, U. P. Wen, and P. L. Yu, editors, *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making*, volume 1, pages 283–292, Berlin, 1994. Springer-Verlag.

[74] S. Shenker. Fundamental design issues for the future Internet. *IEEE Journal on Selected Areas in Communications*, 13(7):1176–1188, 1995.

[75] J. Sridhar and C. Rajendran. Scheduling in flowshop and cellular manufacturing systems with multiple objectives - a genetic algorithmic approach. *Production Planning & Control*, 7(4):374–382, 1996.

[76] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[77] R. E. Steuer. *Multiple Criteria Optimization: Theory, Computation and Application*. John Wiley & Sons, New York, 1986.

[78] R. E. Steuer. An overview in graphs of multiple objective programming. In *Evolutionary Multi-Criterion Optimization. First International Conference, EMO 2001*, volume 1993 of *Lecture Notes in Computer Science*, Zurich, Switzerland, 2001. Springer-Verlag.

[79] R. E. Steuer and E.-U. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26(1):326–344, 1983.

[80] S. Suri, M. Waldvogel, and P. R. Warkhede. Profile-based routing: A new framework for MPLS traffic engineering. In M. I. Smirnov, J. Crowcroft, J. R., and

F. Boavida, editors, *Quality of Future Internet Services: Proceeding of QofIS 2001*, volume 2156 of *Lecture Notes in Computer Science*, pages 138–157, Coimbra, Portugal, 2001. Springer-Verlag.

[81] E. L. Ulungu, J. Teghem, P. H. Fortemps, and D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-criteria Decision Analysis*, 8:221–236, 1999.

[82] D. A. Van Veldhuizen. *Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations*. PhD thesis, Department of Electrical and Computer Engineering, Graduate School of Engineering, Air Force Institute of Technology, Ohio, 1999.

[83] J. Vygen. Np-completeness of some edge-disjoint paths problems. *Discrete Applied Mathematics*, 61:83–90, 1995.

[84] J. Wroclawski. The use of RSVP with IETF Integrated Services. *RFC 2210*, 1997.

[85] X. Xiao, H. Hannan, B. Bailey, and L. M. Ni. Traffic engineering with MPLS in the internet. *Network Magazine*, March 2000.

[86] E. W. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE Infocom '96*, San Francisco, CA, 1996. http://www.cc.gatech.edu/projects/gtitm/.

[87] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, November 1999.

[88] E. Zitzler, M. Laumanns, L.Thiele, C. M. Fonseca, and V. G. da Fonseca. Why quality assessment of multiobjective optimizers is difficult. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 666–674. Morgan Kaufmann Publishers, New York, NY, USA, July 2002.

[89] E. Zitzler and L. Thiele. Multiobjective optimization using evolutionary algorithms – a comparative case study. In A. E. Eiben, Thomas Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving From Nature - PPSN V, 5^{th} International Conference*, pages 292–301, Amsterdam, Holland, 1998.

[90] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

List of Acronyms

ATM	Asynchronous Transfer Mode
CR-LDP	Constraint-based Routing Label Distribution Protocol
ECMP	Equal-Cost MultiPath
FEC	Forwarding Equivalence Class
IP	Internet Protocol
ISP	Internet Service Provider
LP	Linear Programming
LSP	Label Switched Path
LSR	Label Switching Router
MIP	Mixed Integer Programming
MOP	MultiObjective Optimization
MOSA	MultiObjective Simulated Annealing
M-PAES	Memetic-Pareto Archived Evolution Strategy
MPLS	MultiProtocol Label Switching
NPGA	Niched Pareto Genetic Algorithm
NSGA	Nondominated Sorting Genetic Algorithm
PAES	Pareto Archived Evolution Strategy
PSA	Pareto Simulated Annealing
QoS	Quality of Service
RSVP	Resource Reservation Protocol
SLA	Service Level Agreement
SOP	Single Objective Optimization
VEGA	Vector Evaluated Genetic Algorithm

List of Symbols

Chapter 2

$\mathbf{v}$	decision vector
$\mathbf{f}, \mathbf{f}(\mathbf{v})$	vector valued functions
Q	number of objectives
$\mathbf{X}_f$	feasible decision space
$\mathbf{z}$	objective vector
$\mathbf{Z}_f$	feasible objective space
iff	"if and only if"
$<$	component-wise order relation
$\leq$	weak component-wise order relation
$\ll$	strict component-wise order relation
$\prec$	"dominates"
$\preceq$	"weakly dominates"
$\sim$	"is indifferent"
$\mathbf{X}_p$	Pareto optimal set
$\mathbf{Z}_p$	efficient set
$\mathbf{X}_a$	feasible subset
$\mathbf{X}_w$	weakly Pareto optimal set
$\mathbf{Z}_w$	weakly efficient set
$\mathbb{R}^Q$	space of real numbers in Q dimensions
$\mathbb{R}^Q_-$	nonpositive orthant
$\oplus$	Minkowski set addition
δ	weight vector
ϵ	a vector of real numbers
$S_k(\epsilon)$	a single objective optimization problem of the ϵ-constraint method
$\mathbf{z}^{ref}$	reference objective vector
$\mathbf{z}^*$	vector of optimal objective function values
$P(\delta)$, $P(\delta)'$	problem corresponding to the first and second steps of the lexicographic weighted Chebyshev method
α	value of the weighted Chebyshev metric

$\mathbf{X}_\delta$	set of optimal solutions of P_δ
ρ	scalar

Chapter 3

V	the set of nodes in a network
E	the set of directed links in a network
G	graph
u_m	capacity of link m
T	set of traffic requests
d_t	bandwidth demand of t^{th} traffic request
P_t	set of alternative paths of t^{th} traffic request
p_t^i	i^{th} path of t^{th} traffic request
L_t	number of paths in P_t
c_m	cost of link m
$a_{t,m}^l$	equal to 1 if path p_t^l uses link m, and 0 otherwise
C_t^l	cost of corresponding path
x_t^l	amount of traffic routed on the corresponding path
g_m	total load assigned on link m
λ_m	utilization rate of link m
ϕ_m	load balancing cost of link m
Φ^a, Φ^b	load balancing cost functions
y_t^l	equal to 1 if corresponding path is used, otherwise 0
Γ	multiobjective zero-one mixed integer programming problem
h_t	demand of traffic request t not accepted to the network
$d_t^{'}$	updated demand of traffic request t after admission control
$\Gamma(N)$	decomposed version of Γ, where at most N paths can be utilized
μ	optimization problem introduced in the proof of Theorem 3.4
$\mathbf{w}$	decision vector
$Ch(N,\delta)$,$Ch'(N,\delta)$	first and second steps of the lexicographic weighted Chebyshev method corresponding to the decomposed problem
N	bound on the number of used paths
α^*	optimal value of weighted Chebyshev metric
ψ_i	commodity in the network
s_i, t_i	source and destination node
$\mathbf{g}(\mathbf{x})$	vector valued function returning total loads on the links

$\mathbf{u}$	vector of capacities
$\mathbf{c}(\mathbf{g}(\mathbf{x}))$	vector valued cost function
$C_p(\mathbf{g}(\mathbf{x}))$	cost of path p
x_p	amount of flow assigned to path p
$\mathbf{x}$	vector of decision variables representing flows assigned to paths
B	upper bound on the total cost
$S(P_i)$	subset of P_i
W	see Equation 3.29
i^*	target utilization region
n_i	number of links in region i

Chapter 4

$\mathbf{v}^p$, $\mathbf{v}^c$	parent and child solution
$eval(v)$	function to evaluate the performance of the solution
$temp$, $temp_{stop}$	temperature and stopping temperature
r	constant
$\pi(temp, \delta)$	simulated annealing acceptance function
S-metric	metric called also hyper-volume (see Section 4.4)
$\mathcal{C}(A, B)$, $\mathcal{D}(A, B)$	coverage and dominance metrics related with the dominance relation between two nondominated sets

Chapter 5

$\mathbf{g}$	vector of decision variables representing total loads on the links
$\mathbf{\Phi}$	vector of decision variables representing load balancing costs
$\mathbf{y}$	vector of zero-one decision variables
$\mathbf{c}$, $\mathbf{d}$	objective function coefficients
$\mathbf{r}$	vector of continuous decision variables
$\mathbf{s}$	vector of zero-one decision variables
$\mathbf{b}$	righthandside vector
$\mathbf{e}$	vector of ones
$\mathbf{H}, \mathbf{J}$	matrix
$\mathcal{B}, \mathcal{N}$	set of basic and nonbasic variables
$\mathbf{H}_\mathcal{N}, \mathbf{H}_\mathcal{B}$	matrix built by the columns of H corresponding to $\mathcal{N}$ and $\mathcal{B}$
$\mathbf{H}_\mathcal{B}^{-1}$	inverse of $\mathbf{H}_\mathcal{B}$

$\mathcal{N}_r$	nonbasic variable entering to the basis
$\mathcal{B}_s$	basic variable leaving the basis
z_1^*	minimal total routing cost
Δ	stepwise increase in the righthandsides of the constraints imposing lower and upper bounds on the routing cost
K	number of sub-feasible sets
N_{rep}	number of replications in SA
R_i	range factor for i^{th} objective function
max_lsp	upperbound on the number of used paths put for the archiving strategy
min_lsp	minimal number of paths possible to use
NB^i, NB_1^i, NB_2^i	set and subsets of the neighbor solutions of $\mathbf{v}^i$
θ	biasing factor
$\widehat{NB_1^i}$, $\widehat{NB_2^i}$	approximated subsets of the neighbor solutions of $\mathbf{v}^i$
Δ_s, Δ_u	values of Δ in runs
$\mathcal{WS}$	weighted average of the $\mathcal{S}$-metrics
$\mathcal{S}(t)$	$\mathcal{S}$-metric value of the set of solutions using t paths
$w(t)$	weight for the solutions with t paths (used in the calculation of $\mathcal{S}(t)$)
$\mathcal{AS}(t)$	average of the $\mathcal{S}$-metrics (see Section 5.4)
$s1, \ldots, s6$, $u1, u2$	algorithm variants

Lebenslauf

06.11.1975	Geb. in Turgutlu/Manisa, Türkei Familienstand: verheiratet
09.81–06.1983	Namik Kemal Grundschule, Turgutlu/Manisa, Türkei
09.1983–06.1986	Kars Grundschule, Bornova/Izmir, Türkei
09.1986–06.1993	Bornova Anadolu Gymnasium, Bornova/Izmir, Türkei
10.1993–06.1998	Studium des Studiengangs Industrial Engineering an der Bilkent Universität, Ankara, Türkei
09.1998–08.1999	Master-Studium des Studiengangs Industrial and Systems Engineering an Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA
09.1998–08.1999	Wissenschaftliche Angestellte an Department of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, Georgia, USA
02.2000–08.2000	stellvertretende Direktorin bei Tansas A.S., Izmir, Türkei
02.2001–02.2004	Stipendiatin im Rahmen des Graduiertenkollegs "Software für Kommunikationsnetze", RWTH Aachen, Deutschland
seit 02/2004	wissenschaftliche Angestellte am Institut d'administration et de Gestion, UCL, Louvain-la-Neuve, Belgien