MULTIPLE OPTIMIZATION OF NETWORK CARRIER AND TRAFFIC FLOW GOALS USING PREFERENCE FUNCTION MODELING

by

Wayne Sylvester Goodridge

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

Major Subject: Engineering Mathematics

at

DALHOUSIE UNIVERSITY

Halifax, Nova Scotia June, 2005

Your file Votre référence
ISBN: 0-494-13050-4
Our file Notre référence
ISBN: 0-494-13050-4

DALHOUSIE UNIVERSITY

To comply with the Canadian Privacy Act the National Library of Canada has requested that the following pages be removed from this copy of the thesis:

Preliminary Pages
- Examiners Signature Page
- Dalhousie Library Copyright Agreement

Appendices
- Copyright Releases (if applicable)

To my family and friends

Table of Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

At the outset, I would like to express my appreciation to Professor David Mellor from Cable and Wireless and co-supervisor Dr. Robertson for giving me the opportunity to pursue my PHD studies at Dalhousie University. As my co-supervisor, Dr. Robertson played the important role as the "big picture person" and the person that you approach when you are looking for funds to go to conferences. His observations and comments helped me to establish the overall direction of the research and to move forward with investigation in depth.

I would like to express my appreciation to my other supervisor Dr. Phillips who teamed up with Dr. Sivakumar, a member of my research committee, who provided invaluable direction to my research. They both engaged me in constant debate that helped shaped my approach to research. I am also grateful to Dr. Sivakumar for culturing me in the discipline of paper writing.

I would like to express my gratitude to Dr. Cyrus and Dr. Ilow for providing valuable direction. Dr. Cyrus introduced me to Dr. Barzilai who in turn helped me to understand the basic concepts of Preference Function Modeling.

I would like to express my sincere thanks to Gus Osburg who played an important role with helping my family to settle in Halifax.

I would like to express thanks to Dr. Hansen and Dr. Akpan for encouraging me to walk with Christ. Without Christ I would be lost.

I would like to thank my colleagues Nauman Aslan and Maen Artimy with whom I have had the pleasure of working over the last three years. Finally, I would like to thank my wife Jackie for putting up with the late nights in the library, and for giving me motivation for finishing this thesis.

ABSTRACT

The integration of traffic from diverse applications in core networks is placing higher demands on service quality of networks. Traffic Engineering provides a means of controlling traffic through a network by intelligent traffic mappings based on multiple traffic constraints and network carriers optimization goals. However, the problem of finding paths that satisfy the QoS requirements of traffic flows is an NP-complete problem.

The major contribution of this thesis is the proposal of a multiple constraint optimization algorithm called routing decision system (RDS) that uses the concept of preference functions to address the problem of finding paths in core networks that satisfy traffic-oriented performance problems while simultaneously satisfying resource-oriented performance problems. The RDS algorithm is used in conjunction with an exact algorithm called Constraint Path (CP) algorithm or a heuristic algorithm called Constraint Path Heuristic (CP-H) algorithm both of which are novel approaches introduced in this thesis to find a set of constraint paths between source and destination nodes in a network.

A comprehensive Mathematical analysis of the CP and RDS algorithms is given and it is shown that the CP algorithm is an exact algorithm, and therefore guaranteed to find a path that meets the specified constraints, provided that such a path exists. The RDS algorithm is shown to always find a pareto optimal path provided such a path exists in the set of constraint paths. This result is significant since the primary purpose of the RDS is to find optimal paths based on user optimization goals.

In this thesis extensive simulations are performed to compare both the exact and heuristic versions of the CP algorithm used in conjunction with the RDS algorithm with other state of the art QoS algorithms. Results show that the algorithm has a 100 percent success rate in finding paths satisfying multiple user constraints. In addition, it is found that although exact algorithms tend to have high execution times that the CP/RDS algorithm has better running times than other exact algorithms. On the other hand, the success rate of the CP-H/RDS is shown to be significantly better than other heuristic based algorithms under strict constraints. In addition, it is shown that the associated execution time of the CP-H/RDS algorithm is slightly higher than other heuristic based algorithms but good enough for use in an online TE application.

A framework for implementing the CP-H/RDS as a routing server is proposed. The most important advantage is that the complexity introduced by QoS awareness remains outside the network. Simulations to assess how the CP/RDS algorithm benefits the network carrier and traffic flows are presented. It is shown that the RDS portion of the CP/RDS algorithm makes it possible for user flow and network policies to be easily communicated to the path selection process which in turn, significantly impact network and resource utilization.

Chapter 1

Introduction

1.1 Motivation

The flexibility of Internet Protocol (IP) networks has encouraged the integration of a number of network applications including web browsing, e-mail, telephony, voice and video distribution, data mining, and e-commerce. Operators of core networks are faced with the challenge of supporting huge traffic growth in a reliable and cost-effective way. In addition, the traffic in the core has a mixture of best-effort and Quality of Service (QoS) demanding traffic that need protected and predictable service. The merging of these diverse traffic types into virtual pipes imposes new requirements on routing and reservation management in IP networks. Intelligent path selection algorithms based on multiple constraints and a reliable reservation system are now needed in core networks.

The network carrier or Internet Service Provider (ISP) is concerned with the type of routes that are used by user applications. Traffic Engineering (TE) provides a means of controlling traffic through a network, offering services tailored for traffic flow requirements while ensuring economical use of network resources. High service quality, efficiency, survivability, and scales of economy are crucial factors to network carriers. Performance objectives [6] can be traffic-oriented and/or resource-oriented. Traffic-oriented performance objectives relate to the improvement of the user's QoS and can be measured by the packet loss, delay, jitter, and throughput. When service level agreements (SLAs) are involved, protecting traffic flows that comply with their SLAs from those that are noncompliant becomes an important factor in the attainment of traffic-oriented performance objectives. On the other hand, resource-oriented

performance objectives relate to the optimization of the bandwidth asset. Because bandwidth is sold as a commodity [7], more available bandwidth means more revenue for the network carrier.

There are two dimensions to the problem of providing QoS to applications in an IP network. The first is finding the best path to route packets for a given connection and the second is to reserve network resources on that path. Current Internet routing protocols including Open Shortest Path First (OSPF) and Routing Information Protocol (RIP) use routing that is optimized for a single arbitrary metric (shortest path routing). However, routing algorithms using one metric to calculate the path cannot satisfy the diverse QoS requirements needed by multimedia applications. Also, computation becomes more difficult as constraints are introduced into the path calculation problem. This problem can be described as the Multi-constraint path (MCP) problem. The inclusion of optimizing QoS constraints to the MCP problem is described as the Multi-constraint optimal path (MCOP) problem. QoS routing problems involving multiple metrics are NP-complete [8], which means that suitable paths satisfying the user QoS requirements may not be found in polynomial time. The majority of past efforts have concentrated on finding various heuristics that simplify the MCP/MCOP problem. However, heuristics have poor performance and only work well under certain conditions. Some researchers propose exact solutions to the MCP/MCOP problem in light of the fact that the NP-complete character of graphs is not common in real networks. This fact has driven approaches like the Self Adaptive Multiple Constraints Routing Algorithm (SAMCRA) [9] and A*prune [10] algorithms. However, these algorithms have very high running times relative to heuristic approaches.

When core networks use multi-metric QoS routing algorithms in TE, meeting the traffic-oriented QoS goals is just part of the problem. The other is to find paths that

also satisfy the network carrier goals such as maximizing bandwidth usage, maximizing the number of flows in the network and meeting administrative policies. These additional requirements further complicate the path selection problem.

1.2 Research Goals and Results

The major contribution of this thesis is the proposal of a multiple constraint optimization algorithm called Routing Decision System (RDS) that uses the concept of preference functions [11] [3] to address the problem of finding paths in core networks that satisfy traffic-oriented performance problems while simultaneously satisfying resource-oriented performance problems. The RDS algorithm is used in conjunction with an exact algorithm called Constraint Path (CP) algorithm or a heuristic algorithm called Constraint Path Heuristic (CP-H) algorithm both of which are novel approaches introduced in this thesis to find a set of constraint paths between source and destination nodes in a network.

In this thesis extensive simulations are performed to compare both the exact and heuristic versions of the CP algorithm used in conjunction with the RDS algorithm with other state of the art QoS algorithms. Results show that the Constraint Path Routing Decision System (CP/RDS)[1] algorithm has a 100% success rate in finding paths satisfying multiple user constraints. Unlike the conclusion made by the authors in [2] it is found that, typically, exact algorithms tend to have high execution times. However, in the work reported here it was found that CP/RDS performed significantly better than other exact algorithms in terms of execution times under a wide range of user constraints and graph topologies. On the other hand, results show that the CP-H/RDS[2] algorithm has a 90 to 93 percent success rate of finding paths satisfying

[1]The CP/RDS algorithm is the CP algorithm used in conjunction with the RDS algorithm

[2]The CP-H/RDS algorithm is the CP-H algorithm used in conjuction with the RDS algorithm

multiple user constraints under strict [12] user and network constraints. The success rate of the CP-H/RDS algorithm is significantly better than other heuristic based algorithms under strict constraints. In addition, although the associated execution time of the algorithm is slightly higher than other heuristic based algorithms this is compensated for by the gain in higher success rates. Nevertheless, the execution time associated with the CP-H/RDS is good enough for the algorithm to be used in a TE application.

This thesis introduces a framework called the Routing Decision System Server (RDSS) that combines QoS models that have been proposed by the Internet Engineering Task Force (IETF) to address the demand of applications requesting specific services from the network. The framework uses aspects of the Integrated Services (IntServ) / Resource Reservation Protocol (RSVP) model [13], the Differentiated Services (DiffServ) model [13] [14], and Multiple Protocol Label Switching (MPLS) [15] (See Appendix A). In this thesis simulations are designed and implemented that involve the use of the RDSS. The goal of the simulations is to study how using the RDSS can affect user and network performance. It is shown that using the CP-H/RDS allows special optimization goals to be communicated to the network with resultant improvements for traffic and network utilization.

1.3 Structure of the thesis

The structure of the thesis is as follows: Chapter 2 introduces the concept of QoS routing, its benefits and drawbacks. In addition, a literature survey of QoS routing algorithms, their time and space complexities, and problems associated with using these algorithms are discussed. Finally, QoS algorithms used in TE are discussed.

Chapter 3 introduces the CP/RDS as an exact solution for finding constraint paths in a network. This chapter begins by examining the fundamental concepts of exact algorithms and then develops a motivation for the proposal of a new exact

QoS algorithm that uses the RDS optimization algorithm. This chapter outlines the two main algorithm components of the CP/RDS algorithm and gives an example to illustrate how the concept works. Mathematical analysis for both the CP and RDS algorithms is then examined.

Chapter 4 concentrates on the performance analysis of the CP/RDS algorithm with respect to success rates, execution times and probability of producing pareto optimal paths. A comparative analysis is done with the SAMCRA [9] and Jaffe algorithms [16].

Chapter 5 introduces a heuristic version of the CP/RDS algorithm called CP-H/RDS. The main goal of the CP-H/RDS algorithm is to gain faster execution times while maintaining high success rates. This makes it possible for the CP-H/RDS algorithm to be used in an online TE environment. A performance analysis of the CP-H/RDS algorithm with respect to success rates, execution times and probability of producing pareto optimal paths is discussed.

Chapter 6 examines the rationale for using an intelligent path finding algorithm in TE. The CP-H/RDS algorithm is adapted for use in a traffic engineering environment. The RDSS framework is then proposed under which the CP-H/RDS is used as a TE online routing algorithm.

Chapter 7 describes simulation results using the RDSS framework for traffic engineering. The main goal of the simulation is to investigate how a multiple objective optimization algorithm like the CP/RDS could affect application performance.

Finally, Chapter 8 summarizes the thesis and discusses possible directions for further work.

Chapter 2

QoS Routing, Algorithms and Applications

In this chapter, the objectives, benefits and drawbacks of QoS routing are examined. A formal description of the MCP and MCOP problems is then given. A discussion of the current QoS algorithms that provide generic and non-generic solutions to the MCP and MCOP problems is given. The problems associated with current research techniques are examined as each algorithm is discussed. The general problems associated with current exact and heuristic solutions are also discussed. Applications of QoS routing algorithms to the TE environment are examined. In addition, problems associated with using QoS in core networks are discussed.

2.1 Background

2.1.1 QoS Routing

Traditional link state and distance vector routing protocols such as OSPF always forward packets on the shortest path. This can cause problems for flows with a need for QoS guarantees if the shortest path does not have enough resources to meet the requirements. Both IntServ and Diffsev provide mechanisms for flows to reserve resources on the shortest path. However, the shortest path cannot make the reservation if there are not sufficient resources along the path. What is missing is a framework that can find a path, if one exists, which has the requested resources available. The QoS routing [17] [18] [19] [2] [20] [21] scheme considers the quality of service requirements of a flow when making routing decisions.

QoS routing can be divided into two major classes: unicast routing and multicast routing. Unicast routing refers to finding a feasible path between a single source and

a single destination. On the other hand, multicast routing refers to finding a feasible tree covering a single source and a set of destinations.

QoS routing algorithms can also be classified according to the employed routing strategy. The three routing strategies are source routing, distributed routing and hierarchical routing. In source routing algorithms, each network node maintains information about the entire network so that a feasible path can be locally computed at each network node based on the source and destination addresses of the traffic. In distributed routing mechanisms every network node maintains a distance vector and a path computed on a hop-by-hop basis. Finally, in hierarchical routing, nodes are clustered into groups, which are further clustered into higher-level groups recursively, creating a multi-level hierarchy. Each node maintains an aggregated global state. This state contains the detailed state information about the nodes in the same group, and the aggregated state information about the other groups. Source routing is used to find a path between groups from the aggregated state information, and a border node is then responsible for computing the path within its own group.

2.1.2 Objectives and Benefits of QoS Routing

Dynamic determination of feasible paths is one of the key objectives of QoS Routing. That is, given the set of QoS requirements for a flow, the goal is to find a path that can accommodate these requirements. A QoS routing algorithm should achieve this objective by optimization of resource usage. This is a very important objective since network resources are wasted if the QoS routing process chooses a path that is more desirable than the user-specified QoS, as the user does not gain any additional utility. On the other hand, if a path is not found that meets the user-specified QoS then this may affect the user application performance. Hence, integrated communication networks require a routing paradigm that emphasizes searching for an acceptable path satisfying various QoS requirements and a global resource utilization function.

Finally, when the network is congested QoS routing should be able to provide better throughput in the network than best effort routing. That is, more graceful performance degradation should be observed under QoS routing schemes.

The main benefit of QoS routing can be seen with improvement of user application performance. This is because both the network and the user application agree on the level of service the network will deliver before the application data is accepted into the network. QoS Routing optimizes the resource usage in the network as follows [17]:

1. Selecting feasible routes by avoiding congested nodes or links.

2. Offering multiple paths for transferring additional traffic if work load exceeds the limit of existing paths.

3. Selecting alternative paths for quick recovery without seriously degrading the quality if network node failure occurs.

4. Letting traffic classes, with different QoS requirements and identical source and destination, travel different paths.

2.1.3 QoS Metrics

A metric is a characterization parameter for a path or link in a network. Metrics define the types of QoS guarantees the network is able to support. Traffic transmitted through communication networks is characterized by five primary metrics [1], namely loss, delay, jitter, bandwidth and security. Table 2.1 adopted from [1] shows a list of common applications and their QoS requirements.

Since the flow QoS requirements have to be mapped onto path metrics this means that metrics define the types of QoS guarantees the network can support. Metrics can be classified into three types depending on the nature of the metric [8]:

Table 2.1: Common applications and their QoS requirements. Adopted from [1]

Application	Type	Loss	Delay	Jitter	Bandwidth	Security
Email	Data	High	Low	Low	Low	Low
Confidential email	Data	High	Low	Low	Low	High
File Transfer	Data	High	Low	Low	Low, High, Medium	High
Money Transactions	Data	High	Low	Low	Low	High
Audio on demand (AOD)	Real-Time	Low	Low	High	Medium	Low
Video on Demand (VOD)	Real-Time	Low	Low	High	High	Low
Telephony	Real-Time	Low	High	High	High	Low
Video Conferencing	Real-Time	Low	High	High	High	Low
Confidential Video Conferencing	Real-Time	Low	High	High	High	Low

1. Additive Metrics

   $w(P) = \sum_{i=1}^{l} w(L_i)$

   where $w(P)$ is the total value for metric w on path P, the value $w(L_i)$ represents the weight of each link with respect to w on path P, and l is the number of links in the path.

2. Concave metrics

   $w(P) = \min(w(L_i))$

   Bandwidth is an example of such a metric.

3. Multiplicative metrics

   $w(P) = \prod_{i=1}^{l} w(L_i)$

   Packet Loss is an example of a multiplicative metric since it is directly related to the packet success rate and can be expressed as a multiplicative metric consisting of probabilities that a packet will be successfully transmitted on a given link.

Metrics can also be classified as path-constrained or link constrained. Concave metrics are link-constrained because the metric for a path depends on the link's bottleneck value. Additive and multiplicative metrics are path-constrained because the metric for a path depends on all the values along the path.

2.2 Problem Definition

The QoS routing problem can be categorized as the Multi-constraint path (MCP) problem and the Multi-constraint optimal path (MCOP) problem. Formal definitions of both the MCP and MCOP problems are now presented.

2.2.1 MCP/MCOP Problem

Definition 1 (Multi-constraint path (MCP) problem [12]) *Consider a graph $G(N; E)$ consisting of a set of additive metrics $w_i(e)$ for each link $e \in E$, and user requested constraints L_i, $i \in [1, m]$. The goal of a MCP algorithm is to find a path P from source node s to destination node t such that $w_i(P) \leq L_i$ for all i.*

Remark 1 *A path that satisfies all m constraints is often referred to as a feasible path.*

There may be multiple paths in the graph $G(N; E)$ that satisfy the constraints, any of which are solutions to the MCP problem. Hence it may be necessary to use some type of optimization criteria to select a path from the set of feasible paths. This more difficult problem is called the MCOP problem.

Definition 2 (Multi-constraint optimal path (MCOP) problem [12]) *Consider a graph $G(N; E)$ consisting of a set of additive metrics $w_i(e)$ for each link $e \in E$, user requested constraints L_i, $i \in [1, m]$, and a path cost function $cost(.)$. The path p from s to t is called a multi-constrained optimal path if:*

1. *$w_i(P) \leq L_i$ for all i*
2. *$cost(p) \leq cost(p')$ for any p' satisfying condition 1.*

Wang and Crowcroft in [8] stated that: "The problem of finding a path subject to constraints on two or more additive and multiplicative metrics in any possible combination is NP-complete. The results are applicable to any metric that follows additive or multiplicative composition rules and to any metrics that can be transformed to equivalent metrics that follow the additive or multiplicative composition rule". They suggested that the only feasible combinations are one concave metric, bandwidth, and one metric of additive or multiplicative types if exact results are to be obtained.

2.2.2 The nature of the NP-complete problem

A computational problem is said to be in class P (Polynomial) if it can be solved in polynomial time by a deterministic Turing machine [22]. This means that execution time is a polynomial function of the problem size. On the other hand, if a computational problem can be verified in polynomial time then it is said to be in the class of NP (Non-deterministic Polynomial). Clearly, if a decision problem can be solved in polynomial time, then it may be verified in polynomial time, that is $P \subseteq NP$. The class NP-complete is a subset of the class NP, with the property that if any problem can be solved in the NP-complete class in polynomial time then all the problems in NP class can be solved in polynomial time.

As stated previously, given any m additive/multiplicative metrics and their respective constraints, the problem of finding a path satisfying the m constraints is

NP-complete. The proof given in [8] has dramatically influenced the research community, resulting in the common belief that exact QoS routing is intractable in practice. However, it is suggested in [12] that the MCP may not be a "Strong NP-complete" since the combination of network conditions that lead to such behaviour may not be frequent in real networks. The network conditions that induce NP-complete behaviour are the link topology, the link weight correlation structure of the network graph, and the user constraints [4].

Common solutions to NP-complete problems include:

1. Using a heuristic to solve a reasonable fraction of the common cases of the problem.

2. Solving the problem approximately instead of exactly. Often it is possible to come up with a fast algorithm, that does not solve the problem exactly but comes up with a solution that can be proven to be close to right.

3. Using an exponential time solution to solve the problem exactly.

4. Choosing a better abstraction of the problem, since it is possible that the NP-complete problem being solved ignored some important detail of a more complicated real world problem.

2.3 Related Work

In this section existing QoS algorithms are discussed. Solutions proposed for the MCP and MCOP can be classified as either generic or non-generic. Non-generic algorithms focus on specific QoS parameters and, since they are restricted to a predefined set of metrics, are not able to solve the true MCP problem. On the other hand, generic algorithms are able to compute a path from source to destination based on an arbitrary set of constraints. An organizational chart for QoS algorithms is provided in Figure 2.1.

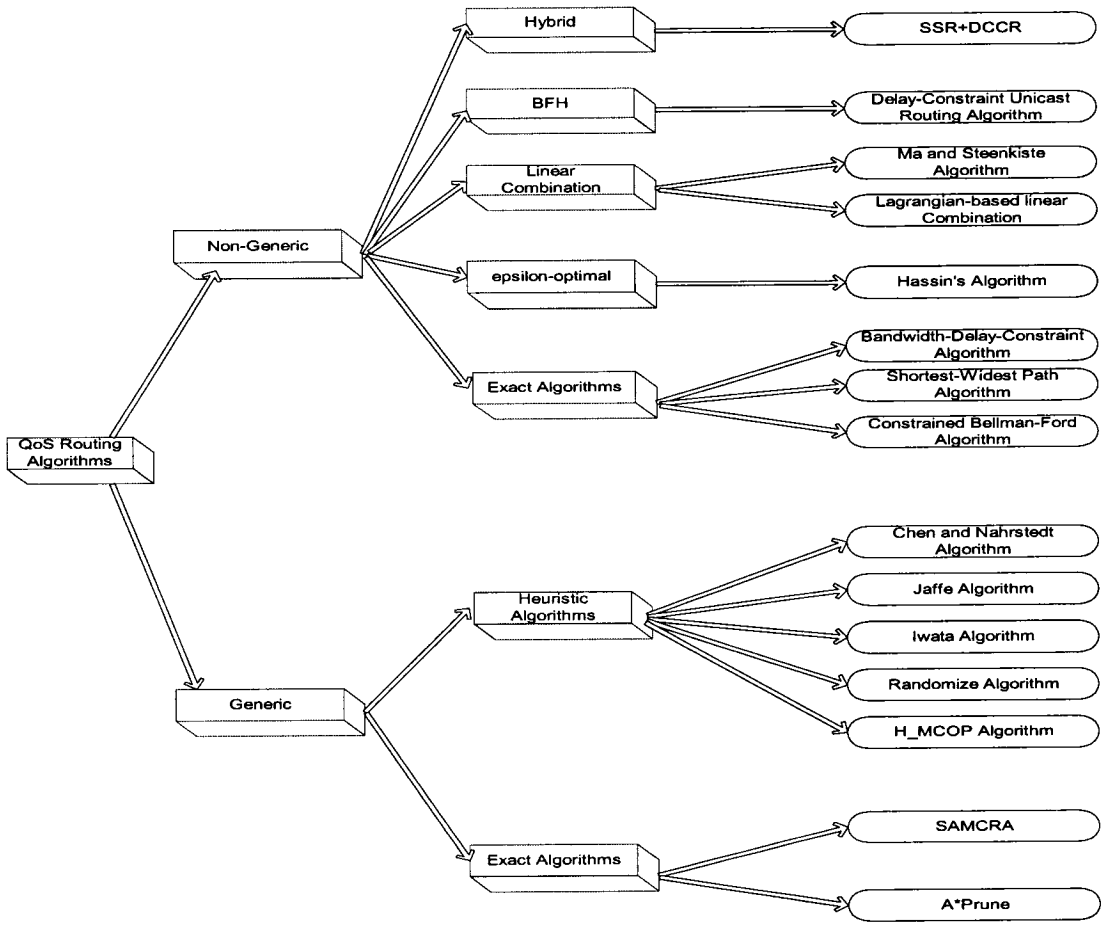


Figure 2.1: Summary Chart of QoS Routing Algorithms

2.3.1 Non-Generic Algorithms

Non-generic algorithms can be classified as exact, ϵ-optimal approximation, backward-forward heuristic, linear composition and hybrid algorithms.

2.3.1.1 Exact Algorithms

Exact Algorithms guarantee that a path satisfying the design goals of the algorithm would be returned, providing that at least one such path exists in the network. The following algorithms are non-generic exact algorithms.

Bandwidth-Delay-Constraint Algorithm This is a source routing algorithm which eliminates all links with a bandwidth less than the user requested bandwidth, resulting in a graph satisfying the bandwidth constraint [8]. The shortest path in terms of delay is then found by using the Dijkstra's algorithm. The path is feasible if and only if it satisfies the delay constraint.

This algorithm is not NP-complete since it has the same time complexity as Dijkstra's algorithm. However, if more than one path satisfies the user requested delay this algorithm can cause the path that has the least delay to be saturated. In addition, this algorithm assumes that the user is requesting a path that is optimal in terms of delay. However, this practice can result in the rapid depletion of the buffer resource in routers on the paths that have the lowest delay. What is needed is a routing strategy that will reserve the low delay paths for users that request optimal delay paths. The RDS algorithm solves this problem by finding a path that matches the user constraint and optimization requests as closely as possible. For example, if a network has two paths that satisfy the delay constraint between a source and a destination node, and the user does not request optimization of the delay constraint, the path with the largest delay will be selected by the RDS algorithm. However, for the same scenario the Bandwidth-Delay-Constraint path algorithm will select the path with the lowest delay.

Shortest-Widest Path Algorithm The Shortest-Widest Path Routing Algorithm also proposed in [8] is a distributed routing algorithm. The path search strategy is to find a path with maximum bottleneck bandwidth (a widest path), and to choose the link that would result in the shortest loop free propagation delay when there is more than one widest path.

This algorithm has polynomial time complexity. However, the resulting path of the algorithm is primarily based on the bandwidth metric. The reason is that, during path computation, the delay metric is only used as a tie breaker if more than one

widest path exists. Hence this selected path is the same as the one calculated with one routing metric. Therefore, a path that reflects the user optimization goals and constraints is not necessarily found by this algorithm. The RDS algorithm uses a preference function that has a preference scale for each metric on each constraint route connecting the source and a destination node in the network. This provides a framework where all metrics participate in the path selection process.

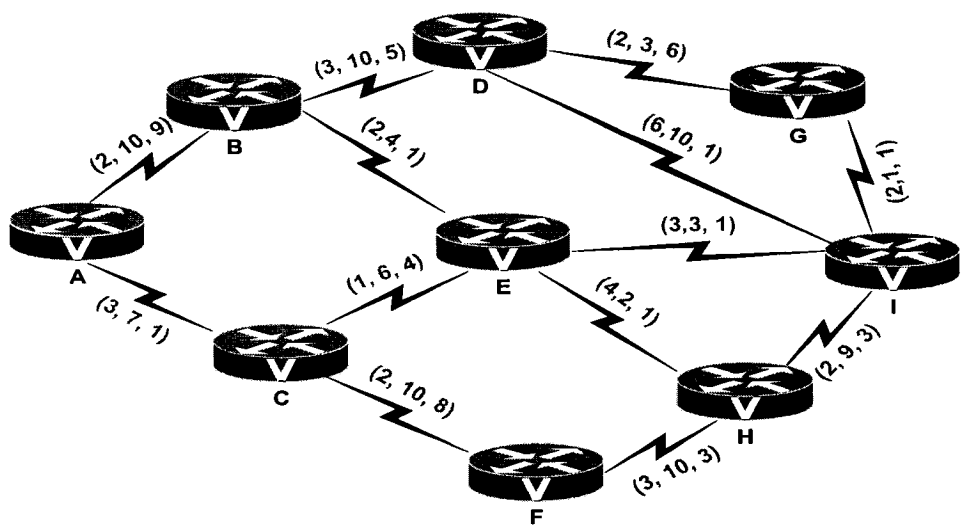


Figure 2.2: Communication network with links characterized by delay, bandwidth and cost.

Constrained Bellman-Ford Algorithm The Bandwidth-Delay-Constraint and the Shortest-Widest Path algorithms are simple polynomial time algorithms, however the problem becomes difficult when there is more than one additive constraint. A very common MCP problem is the delay-constrained least-cost (DCLC) problem. This problem is solved by the Constrained Bellman-Ford (CBF) algorithm proposed by Widyono [23]. To find the minimum cost path that does not violate the delay constraint the CBF algorithm performs a breadth-first-like search, discovering paths of increasing delay while recording and updating lowest cost paths to each node it

visits.

CBF has worst-case exponential complexity. The CP algorithm proposed in this thesis is similar to the CBF algorithm. However, the CP algorithm introduces tuning parameters and techniques that assist with the reduction of actual execution times involved in finding constraint paths. In addition, the CP algorithm is a generic algorithm.

Routing Scenarios involving Non-Generic Exact Algorithms Figure 2.2 is a communication network whose links are characterized by delay, bandwidth and cost. Table 2.2 shows a list of user requests for finding a feasible route between source node A and destination node B, and the corresponding paths produced by each algorithm.

Table 2.2: Examples of paths selected based on given user requests for non-generic exact algorithms.

User Requests	Bandwidth-Delay -Constraint	Shortest-Widest Path	Constrained Bellman-Ford
delay = 10 bandwidth = 3	A-B-E-I or A-C-E-I	Not Applicable	Not Applicable
bandwidth = 6 delay = ∞	A-B-D-I	A-B-D-I	Not Applicable
delay = 7 cost = 6	Not Applicable	Not Applicable	A-C-E-I

2.3.1.2 ϵ-Optimal approximation

Hassin's algorithm One general approach for dealing with NP-complete problems is to look for an approximation algorithm with polynomial time complexity. Hassin's algorithm [24] is an example of a ϵ-optimal approximation algorithm that solves the Restricted Shortest Path problem (RSP). The RSP is similar to the DCLC problem; however, both cost and delay are constrained. The algorithm returns a path whose

cost is at most $(1+\epsilon)$ times the cost of the optimal path, where $\epsilon > 0$. The general idea is to first devise an optimal (non-/pseudo-polynomial) algorithm, whose complexity is proportional to the largest possible value of the delay/cost. If the set of possible delay/cost values is scaled down to a small enough range, then the scaled problem can be solved optimally in polynomial time. The solution is then rounded back to the original delay/cost values with some bounded error. For the network given in Figure 2.2, Table 2.3 shows examples of selected paths based on given user constraints.

Table 2.3: Examples of paths selected based on given user requests for the Hassin's algorithm.

User Requests	Hassin's Algorithm
delay = 10 cost = 3	Not Feasible
delay = 20 cost = 7	A-C-E-I
delay = 20 cost = 20	A-C-E-I

The Hassin's algorithm is very elegant but is not scalable in terms of the number of additive metrics that can be used. In addition, the algorithm's complexity makes it difficult for use in large networks. Both the CP/RDS and the CP-H/RDS algorithm proposed in this thesis can handle a wide range of additive metrics. Results presented in Chapter 5 suggest that the CP-H/RDS algorithm is scalable under strict and loose user constraints [12].

2.3.1.3 Backward-forward heuristic

Delay-Constrained Unicast Routing (DCUR) algorithm The Backward-Forward Heuristic (BFH) first determines the Least-Delay Path (LDP) and Least-Cost Path (LCP) from source to destination. It then starts from the source node to build the

path in a node by node fashion that reduces the total cost from source to destination, while its approximated end-to-end delay obeys the delay constraint. The DCUR proposed by [25] to solve the DCLC problem is an example of a BFH. The DCUR is a heuristic source-based algorithm that maintains network cost and delay vectors at each node in the network. These vectors are similar to the distance vectors of existing distance-vector-based protocols such as RIP. The cost (delay) vector contains the next node on the least-cost (least-delay) path for every destination. The path is constructed one node at a time, from source to destination. Any node v at the head of the partially-constructed path can choose to add one of only two alternative outgoing links. One link is on the least cost path from v to the destination, while the other link is on the least delay path from v to the destination. This active node, v, checks if the link on the least cost path does not violate the delay constraint to the destination. If the link satisfies the constraint then a delay-constrained path from v to the destination exists and the link is selected for the path. If not, the link on the least delay path is selected.

When tight delay constraints are considered the DCUR algorithm may return paths that are high in cost because, in this case, the algorithm will rarely follow the unconstrained least-cost path. Hence the algorithm does not always return a least cost path. In Section 3.2, this thesis presents a routing paradigm that separates the path search mechanism from the optimization mechanism. The goal is for the path search process to find a subset of paths that match the user constraints, and then pass them to the optimization mechanism which decides the best path based on a set of optimization goals. Since the final path selection process is not graph dependent, the solution presented in this thesis will return paths that are based on flow optimization goals.

2.3.1.4 Linear Composition

Ma and Steenkiste Algorithm Ma and Steenkiste [26] show that if a rate proportional policy (for example, WFQ-like scheduling algorithms [27]) is used on routers within a network then delay and jitter can be expressed in terms of residue bandwidth. Hence the MCP-problem for bandwidth, delay and jitter can be solved in polynomial time by reducing the problem to a single parameter problem.

The main problem with the Ma and Steenkiste approach is that only user QoS constraints are considered in the linear combination. However, when QoS algorithms are used in TE the network administrator may wish to consider other metrics like a resource cost function in the path selection process. In addition, although queuing delay can be formulated as a function of bandwidth, this is not the case with propagation delay, which is the dominant delay component in high speed networks [28]. Furthermore, this approach does not provide the user with the option to specify optimization goals.

The CP/RDS algorithm proposed in this thesis supports both independent and dependent metrics which make it possible for the algorithm to handle other metrics apart from bandwidth, delay and jitter. In fact, the RDSS framework, Chapter 6, for implementing the RDS algorithm on the management plane [29] suggests that QoS metrics can be estimated using the Ma and Steenkiste algorithm. However, the flexibility of the RDS algorithm allows other metrics to be considered in the routing process.

Lagrangian-based linear composition Another form of linear composition is the Lagrangian-based linear composition that combines the delay and cost of each link into a link weight $w(u,v) = \alpha d(u,v) + \beta c(u,v)$ and finds the shortest path with respect to $w(u,v)$. Note that u and v are vertices in a graph, and $d(u,v)$ and $c(u,v)$ represent the link delay and cost for the directional link that connects u to

v respectively. Determining appropriate values for the multiplier constants α and β is important and can significantly affect the performance of algorithms that use this technique. The authors in [30] [31] use the concept of aggregated costs and they provide an efficient method to find the optimal multiplier based on Lagrangian relaxation. This method involves iteratively finding the shortest path with respect to the linear combination of cost and delay and adjusting the multipliers in the direction of the optimal path.

The idea of finding appropriate values for α and β for the link weight function is a very elegant solution. This is because once α and β and are known a simple Dijkstra algorithm can be used to compute the optimal path. However, the main problem of this approach is that the user optimization goals may not be reflected in the path selected by the algorithm since that path is entirely based on the values of α and β. As will be seen in Section 3.5.1 the RDS algorithm proposed allows the user to have a say in the type of optimization that the path search process uses.

2.3.1.5 Hybrid Algorithms

Hybrid algorithms use combinations of exact, ϵ-optimal approximation, backward-forward heuristic and linear composition techniques. One such hybrid algorithm is called Search Space Reduction plus Delay-Cost-Constrained routing (SSR + DCCR) [32]. This algorithm solves the delay-constrained minimum-cost path problem by using a variant of the k-shortest-path algorithm [33], an adaptive path weight function with an additional constraint imposed on the path cost, to restrict the search space.

The use of a k-shortest-path algorithm can potentially lead to high execution times [12]. However, based on the limited objectives of the algorithm it solves the MCP problem elegantly. However, what is needed is a generic solution that would allow the user and network policy makers to specify strict and loose constraints and

optimization goals. The RDS algorithm presented in Chapter 3 avoids the computational expense associated with the use of a k-shortest path algorithm and provides an environment where metrics can participate in the path selection process.

2.3.2 Generic Algorithms

As the MCP and MCOP problems are NP-complete and as a consequence very few exact solutions are proposed. An exact solution is guaranteed to give a feasible path for a specified set of user constraints. On the other hand, many heuristic solutions are proposed for MCP and MCOP problems but do not guarantee that a feasible path would be found that satisfies a set of user constraints.

2.3.2.1 Heuristic Algorithms

Chen and Nahrstedt Algorithm Chen and Nahrstedt [16] [34] proposed a technique for solving the MCP problem for 2 additive metrics. Suppose that

$MCP(G; s; t; w_1; w_2; C_1; C_2)$ represents the original problem, where G is the network, s and t are the source and destination nodes respectively, w_1 and w_2 are the metrics, and C_1 and C_2 are the user specified metric values for w_1 and w_2 respectively. The technique involves reducing the original problem to a simpler problem $MCP(G; s; t; w_1; w_2'; C_1; x)$ where x is some predetermined integer and

$$w_2' = \left\lceil \frac{w_2 \times x}{C_2} \right\rceil \tag{2.1}$$

Chen and Nahrstedt proved that a solution to the new problem is also a solution to the old problem and proposed extensions to Dijkstra's algorithm and the Bellman-Ford algorithm to solve the problem. This is a heuristic approach since finding a feasible path is a function of x. The solution to the actual problem might not be a solution to the simpler problem, which deems this algorithm to be a heuristic. The

success rates and running times of the algorithm are heavily influenced by the value of x. To obtain high success rates a high value of x is required. However, using high values for x can result in very high running times. However, the CP and the CP-H algorithms also depend on predetermined parameters which have an impact on the trade-off between execution times and success rates of the algorithms. For example, in the case of the CP-H algorithm high values of λ[1] result in high success rates but poor execution times. Results of simulations done in Section 5.2 indicate that when $\lambda = 3$ success rates of between 90-93 percent are obtained under strict user constraints. However, results in [16] suggest that the Chen and Nahrstedt algorithm does not exceed a success rate of 50% under strict user constraints.

Jaffe Algorithm Jaffe [16] proposed a method for solving the MCP problem for $m = 2$. Consider a graph $G(N; E)$, where N is a set of vertices and E is a set of edges, and $u, v \in E$. Suppose that link is characterized by weight functions $w_1(u, v)$ and $w_2(u, v)$. The Jaffe algorithm replaces the two link values of each link by a single link value which is a linear combination of the original link values as given in Equation 2.2, where d_1 and d_2 are constraints.

$$w(u, v) = d_1.w_1(u, v) + d_2.w_2(u, v) \tag{2.2}$$

The problem with the combined weight as a metric is solvable by a shortest path algorithm. The solution is then checked to determine whether the constraints are satisfied. However, the shortest path of the converted graph consisting of single weights may not satisfy user constraints L_1 or L_2. Jaffe argued that the values for d_1 and d_2 should be chosen based on Equation 2.3. This implies that in the worst-case scenario, one of the constraints can be violated by 100.

[1]Threshold used in the CP-H algorithm that determines the number of optimized paths considered for each metric see Section 5.1

$$\frac{d_2}{d_1} = \sqrt{\frac{L_1}{L_2}} \tag{2.3}$$

The key issue now is to determine the appropriate d_1 and d_2 such that an optimal path with respect to the new single weighted graph is likely to satisfy the individual constraints.

The Jaffe algorithm is heavily influenced by the nature of the user constraints and the link correlation structure of the network. Hence the performance of the algorithm in terms of success rates is unpredictable. In addition, the optimization strategy of the algorithm is fixed and cannot be influenced by individual user flows. The exact CP/RDS is guaranteed to give a success rate of 100% and the heuristic version high success rates when strict and loose constraints are considered. In addition, the routing paradigm proposed in this thesis is tailored to satisfy individual user QoS requirements. This is accomplished by separating the path search mechanism from the optimization mechanism.

Iwata Algorithm The Iwata algorithm presented in [35] attempts to find a path optimized for one of the metrics, instead of being optimized for a single cost of weighted QoS parameters. The algorithm finds a shortest path, or paths, based on one metric and then checks if all the constraints are met. If not, it finds a shortest path for another metric and again checks if the other constraints are met.

The Iwata algorithm provides a straightforward approach to solving the MCP problem. However, results in [2] suggest that the Iwata algorithm has very low success rates under strict user constraints. The CP-H introduced in Section 5.1 has similarities to the Iwata algorithm. However, rather than testing only one path for each metric, the CP-H algorithm tests λ paths for each metric. Simulation results indicate that this approach leads to a significant increase in success rates over the Iwata algorithm.

Randomize Algorithm The concept behind randomize algorithms proposed in [36] is to make random decisions during execution so that unforeseen traps can potentially be avoided when searching for a feasible path. The randomized algorithm proposed in [37] has an initialization phase and a randomized search. In the initialization phase, the algorithm computes the shortest paths from every node u to the destination node t first with respect to each QoS measure, and then with respect to a linear combination of all m measures. This information provides lower bounds for the path weight vectors of the paths from u to d. In the second phase of the algorithm a randomized Breadth-First Search (BFS) which discovers nodes that have a good chance to reach the destination d. In the original BFS all nodes that are attached to the root node of a sub-graph are discovered. However, in the case of a randomized BFS a check is done to see whether there is a chance of reaching the destination. If there is no chance of reaching the destination, the algorithm foresees the trap and avoids exploring such nodes any further.

Results in [2] show that the Randomize algorithm has relatively high success rates and relatively high running times. In addition, the optimization strategy of the algorithm is network based. Hence, there is no mechanism for individual user flows to communicate their optimization goals to the network. As stated earlier, the routing paradigm proposed in Section 3.2 is tailored to satisfy the individual user QoS requirements and so overcomes this disadvantage of the Randomize algorithm.

H_MCOP Algorithm Korkmaz and Krunz in [38] proposed the H_MCOP heuristic algorithm. This algorithm has two phases. The first phase of the algorithm computes the optimal path from every node u to the destination node t with regard to the linear combination metric of Equation 2.2, setting $d_i = \frac{1}{C_i}$. This first phase returns the optimal path for the linear combination metric. In the second phase, the algorithm uses the non-linear cost function given in Equation 2.4 to compute paths starting from source node s. The algorithm concatenates the non-linear path length

from s to u and the linear path length from u to t to approximate the length of the path from s to t. Since the algorithm considers complete paths before reaching the destination, it can foresee several infeasible paths during the search. If paths seem feasible, then the algorithm can switch to explore these feasible paths based on the minimization of the single measure.

The H_MCOP algorithm has a relatively high success rate and reasonable running times. However, the main difficulty is that the optimization strategy of the algorithm is fixed and cannot be influenced by individual user flows. The routing paradigm proposed in this thesis is tailored at satisfying the individual user QoS requirements and so overcomes this disadvantage of the H_MCOP algorithm.

2.3.2.2 Exact Algorithms

TAMCRA and SAMCRA The Tunable Accuracy Multiple Constraints Routing Algorithm (TAMCRA) algorithm [39] and its successor SAMCRA [9] are exact algorithms for the MCP problem and use a non-linear function to calculate path lengths. The combined path weight is given in Equation 2.4.

$$L(P) = \max_{i \leq j \leq m} \left(\frac{w_j(P)}{L_j} \right) \tag{2.4}$$

Both the TAMCRA and the SAMCRA use the k-shortest path algorithm [33]. In TAMCRA, k is pre-selected, while the SAMCRA algorithm controls the value of k self-adaptively. The selection of k in TAMCRA is a trade-off between performance and complexity; a small k implies that the algorithm will have polynomial running times but poor performance in terms of returning optimal paths while a large k means that the algorithm returns optimal paths at the expense of long running times. SAMCRA, on the other hand, guarantees to find a feasible path, if one exists, at the expense of exponential running times.

The SAMCRA algorithm offers many advantages in its approach to solving the MCP/MCOP problem. This thesis uses the SAMCRA algorithm extensively to gauge the performance of the proposed CP/RDS and CP-H/RDS algorithms. The major difficulty is that the optimization strategy of the algorithm is fixed for a given network and cannot be influenced by individual user flows. In addition, the use of a k-shortest path algorithm forces the algorithm to exhaustive searches in order to find an optimal path. This practice can lead to exponential running times.

A*Prune Liu and Ramakrishnan proposed the A*Prune algorithm [10] for the MCOP problem, which is an exact algorithm that uses a technique similar to the Randomize Algorithm. The algorithm has an initialization phase that finds the shortest path from each node u to destination node t and from source node s to each node u, with regard to each metric and the linear combination metric in Equation 2.2. Then starting from node s, the algorithm discovers each node based on the weight of the path P from s to t via u based on the linear weight function. All neighbor nodes are considered, and those that cause a loop, or would not meet the constraint, are pruned. The algorithm continues until k shortest paths are found or the heap is empty.

The A*prune algorithm always finds k shortest paths. This suggests that the algorithm cannot run in polynomial time according to [40]. Hence the A*prune algorithm is not suitable for use in large networks since running times of the algorithm may be too high. The RDS algorithm presented in this thesis avoids the computational expense associated with the use of a k-shortest path algorithm.

2.4 Complexity of QoS Algorithms

The complexity of an algorithm refers to the minimum amount of resources needed to solve a problem or execute an algorithm. Complexity can be classified into time and space complexity. The time complexity refers to the number of steps an algorithm

takes (in the worse-case) to return its answer, while the space complexity refers the amount of memory needed to run the algorithm. Tables 2.4 and 2.5 give the worse case time and space complexities for the non-generic and generic algorithms respectfully.

The symbols used in Tables 2.4 and 2.5 are explained as follows:

1. N is the number of nodes in the graph
2. m is the number of metrics
3. k is the number of shortest paths
4. E is the number of vertices in a graph
5. x used in the Ma and Steenkiste is the number of possible distinct residual bandwidth of a link
6. y used in the SSR + DCCR is a function of the configuration of the graph.

Table 2.4: Worst-case time and space complexities of some non-generic QoS algorithms [2].

Algorithm	Time	Space
CBF	$O(e^{\alpha N})$	$O(N)$
Shortest-widest Path	$O(NlogN + E)$	$O(N)$
Widest-shortest Path	$O(NlogN + E)$	$O(N)$
BFH	$O(NlogN + E)$	$O(N)$
Ma and Steenkiste	$O(xNE)$	$O(N)$
Lagrangian-based Linear Composition	$O(E^2 \log^2 E)$	$O(N)$
SSR + DCCR	$O(yE \log N + kE \log(kN) + k^2E)$	$O(kN)$

Table 2.5: Worst-case time and space complexities of some generic QoS algorithms [2].

Algorithm	Time	Space
Jaffe's algorithm	$O(NlogN + mE)$	$O(N)$
Iwata's algorithm	$O(mNlogN + mE)$	$O(N)$
Randomized algorithm	$O(N^3)$	$O(N)$
H_MCOP	$O(n\log(n) + km\log(kn) + (k^2+1)m)$	$O(N)$
SAMCRA, TAMCRA	$O(Nk_{min}log(Nk_{min}) + k^2mE)$	$O(kmN)$
A*Prune	$O(N!(m + N + NlogN))$	$O(mN!)$

2.5 Problems with QoS Routing Algorithms

Heuristic solutions for the MCP are very common and typically have polynomial-time and pseudo-polynomial time complexities, and suffer from excessive computational complexities or low performance, which rule them out as candidates for online routing. Moreover, most heuristic algorithms are highly specialized and cannot adapt easily to a wide range of user requirements.

An exact solution is guaranteed to give a feasible path based on specified user requests. These algorithms offer good performance at the expense of possible high complexities and running times growing exponentially in the worse case. To illustrate the low performance of heuristics algorithms and the potential high running times of exact algorithms the graphs of the simulation results done in [2][4] are reproduced. The simulation results involve the success rate and execution time. The success rate is the number of times a feasible path is found divided by the number of graphs examined. The execution time is a normalized execution time referring to the algorithms execution time over all graphs examined divided by the execution time that the Dijkstra algorithm would take computing shortest paths for the same graphs. Note that the authors in [2] found that Bellman-Ford based algorithms perform significantly poorer than Dijkstra-like algorithms as far as execution time is concerned.

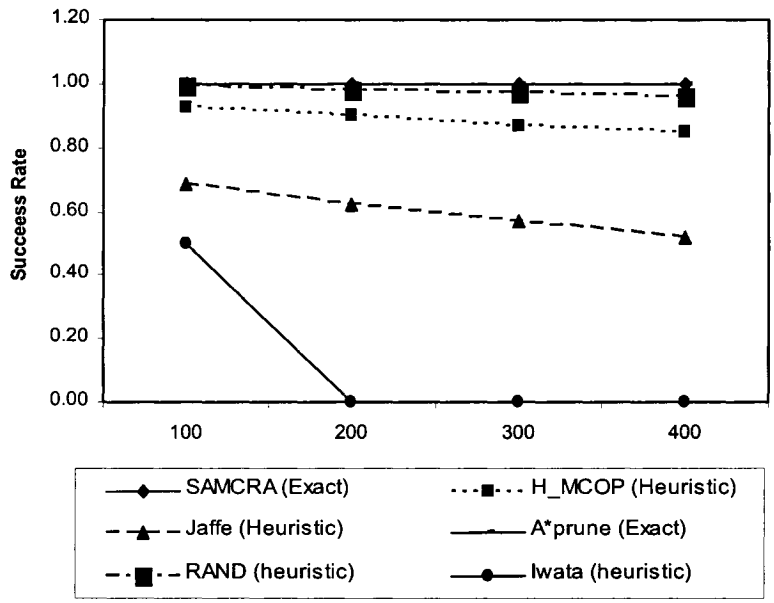


Figure 2.3: Success Rate as a function of the number of nodes ($m = 2$ and strict constraints) [2][4].

Figures 2.3 and 2.4 show the results given in [2] for strict user constraints under Waxman [41] network topologies. Figure 2.3 shows the success rates for some QoS routing algorithms as a function of network size while Figure 2.4 shows the normalized execution times as a function of network size.

The results indicate that although heuristic algorithms have relatively fast running times, their performance in terms of finding constraint paths that meet the user constraints is low. On the other hand, exact algorithms all have 100% success rates and higher running times than heuristic algorithms.

Finding a path that satisfies user optimization needs complicates the path finding process. Many algorithms like the SAMCRA and A*prune algorithms achieve user optimization goals by using a non-linear function. However, because the Dijkstra algorithm is designed to use a linear optimization function many QoS algorithms are

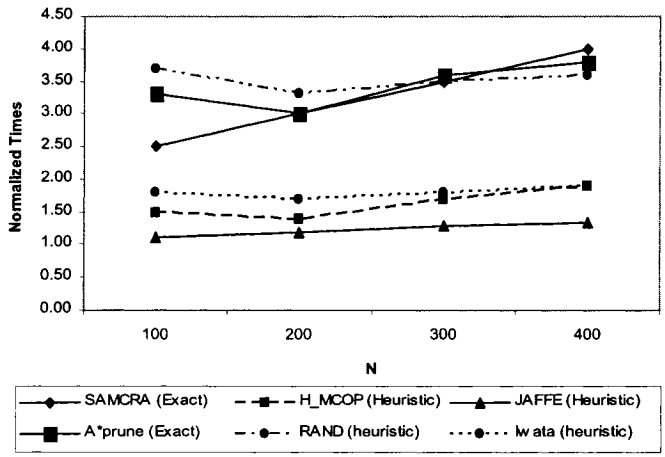


Figure 2.4: Normalized times as a function of the number of nodes ($m = 2$ and strict constraints). [2][4]

forced to use a modified version of the Dijkstra algorithm that finds the shortest path from a list of k-shortest paths. However, because k, the number of paths that exists between source and destination nodes can be large, this means that to find the shortest path among k paths can be time consuming This is one of the main problems with the use of a non-linear function.

Most QoS algorithms attempt to find optimal paths for only additive metrics. When a concave metric such as bandwidth is involved in the optimization process, no QoS algorithm, in the work reported here, offers a solution where bandwidth can be optimized. This is primarily because most QoS algorithms deal with the problem of meeting the requested bandwidth by pruning the network of all links that do not satisfy the bandwidth request. A cost function is typically then used with the Dijkstra algorithm to find an optimal path from the pruned network. However, this optimal path is based only on the cost function which is typically based on an additive metric. Although single metric algorithms such as the Constraint Shortest Path First (CSPF) provide a mechanism for finding optimal bandwidth paths, the existing MCP/MCOP

solutions do not provide a mechanism where a pareto optimal path [42] can be found that involves the bandwidth metric.

The routing paradigm presented in this thesis attempts to rectify the problems discussed above by separating the path finding mechanism from the optimization mechanism. This modular approach makes it possible to design more flexible path searching and optimization techniques. Since the search process is decoupled from the path search process, the use of a k-shortest path algorithm is less significant. Hence the associated expense that comes with the use of the k-shortest algorithm is eliminated. Instead, the designer of the path finding algorithm is free to use a wide range of techniques to find a set of constraint paths connecting source and destination nodes. In addition, the optimization mechanism is decoupled from the Mathematics of Graph Theory that makes it possible for optimization techniques from other areas of Mathematics such as Utility Theory to be used. Consequently, the results presented in this thesis will show that the CP/RDS and the CP-H/RDS provides the best success rate /execution time trade-off relative to current exact and heuristic algorithms.

2.6 Applications of QoS Algorithms to Traffic Engineering

Traffic Engineering [43] [44] [45] [46] [47] (See Appendix A) is the technique of mapping network traffic flows with a given set of constraints and optimization goals onto an existing physical topology in such a way that: 1) the goals of the traffic flows; and 2) the goal of getting maximum utilization out of the physical topology are met.

The key to a TE solution is how the traffic mappings are created. There are three strategies used for creating traffic mappings; online, offline or a combination of online and offline procedures and algorithms. Online TE solutions make dynamic routing decisions for customer flows in real time. On the other hand, offline TE solutions make routing decisions that are on a long time scale and do not allow routing decisions to be made in real time. This thesis is concerned with the application of QoS routing

algorithms to online TE with the goal of providing dynamic call admission services. Online TE is important because it reduces the need for manual configuration of network flows and provides a framework for the network to adapt more speedily to customer demands on the network.

Online TE uses a QoS routing algorithm for path computation. The ingress node of the core network determines the physical path for each flow by applying a QoS routing algorithm on the information in the Traffic Engineering Database (TED). This information may include:

1. Topology link-state information learned from the Interior Gateway Protocol (IGP);

2. Attributes associated with the state of network resources (such as total link bandwidth, reserved link bandwidth, available link bandwidth and link delay) that are carried by IGP extensions; and

3. Administrative policies.

Online TE is very important because it allows dynamic changes to network configuration based on changes in network policy, network traffic load changes, and network failures. Unfortunately, online optimization approaches take into account all traffic paths in the sequence that they are proposed in the network. This approach is not deterministic since the order in which explicit paths are calculated plays a critical role in determining their physical paths across the network. There is no way of ensuring that the order in which the traffic paths are set up will lead to an optimal solution. In fact, in many cases, much traffic is blocked because of the inefficient allocation of network resources that result from the order in which traffic flows request these resources. Other major disadvantages of using QoS routing algorithms in TE are:

1. Protection paths [45] are supported by QoS routing algorithms only in the 1:1

scheme. That is, no provision for the installation of a backup path is made. This is not an efficient protection policy.

2. Sharing protection capacity that is assumed to be exempt from simultaneous failure is a more efficient solution but requires a view of all trunks and their paths through the network.

3. QoS routing algorithms are not suitable in Diffserv networks since they cannot estimate the impact of a new trunk on the QoS characteristics of the existing trunks. This requires a view of all trunks and the load changes along these paths when a new trunk is introduced.

This thesis proposes a RDSS framework for using the CP-H/RDS algorithm in a TE environment that rectifies most of the above mentioned issues of adapting QoS algorithms for use in a TE environment. The use of the CP-H/RDS makes it easy for disjointed constraint paths to be found which in turn makes it easy to implement the concept of sharing protection capacity mentioned above in point 2. The RDSS framework addresses point 3 by implementing routing decisions on the management plane. This facilitates the use of CP-H/RDS and the Diffserv model since paths can be calculated based on a global view of all network trunks. However, this thesis does not provide a solution for the inefficient allocation of network resources that result from the order in which traffic flows request network resources.

2.6.1 QoS algorithms used in Traffic Engineering

QoS algorithms that are used in TE online routing algorithms include Minimum Interference Routing Algorithm (MIRA), Least Interference Optimization Algorithm (LIOA) and Minimal Hop Algorithm and Constraint Shortest Path First (CSPF).

2.6.1.1 Minimum Interference Routing Algorithm

The Minimum Interference Routing Algorithm (MIRA) [48] is a flow-based routing algorithm proposed in the context of MPLS networks to set up bandwidth-guaranteed LSPs. MIRA selects a path for a LSP request that maximizes the minimum available capacity between all other ingress-egress pairs. This routing strategy helps to reduce the blocking rate of LSP requests and prevents the creation of bottlenecks for flows.

There are many variations of MIRA which include Light Minimum Interference Routing Algorithm (LMIR) [5] which attempts to find K paths with the lowest capacities among all paths between two nodes, and then selects the path which minimizes the interference between the two nodes. Results in [5] show that the LMIR has significant improvements over other variations of the MIRA algorithm.

2.6.1.2 Least Interference Optimization Algorithm

The Least Interference Optimization Algorithm (LIOA) [49] reduces the interference among competing flows by balancing the number and quantity of flows carried by a link to achieve efficient routing of MPLS label switched paths (LSPs). The algorithm uses the current bandwidth availability and the traffic flow distribution to achieve traffic engineering in IP networks. The link cost function employed is given in Equation 2.5.

$$cost_l = \frac{I^\alpha}{(R_l - r_l)^{1-\alpha}} \tag{2.5}$$

where

I is the is the number of flows on link l

R_l is the maximum reservable bandwidth on link l

r_l is the total bandwidth reserved by the LSPs traversing the link

α is a parameter representing the trade-off between the number and the magnitude of the LSPs traversing the link l.

2.6.1.3 Constraint Shortest Path First

Constraint Shortest Path First (CSPF) [50] solves the problem of load balancing in the OSPF protocol by using link costs which reflect the current resource availability. These include costs which are inversely proportional to the residual link capacities. For example, the cost function for a link between two nodes is given in Equation 2.6. Thus, routing preference is given to paths that are shortest in terms of a combination of low total delay and high throughput.

$$C = \frac{1}{\min(bw)} + \sum_{e_i \in P^*} d_i \tag{2.6}$$

where

$\min(bw)$ = lowest link bandwidth in kbps

$\sum_{e_i \in P^*} d_i$ = total delay in seconds along path P^* connecting the nodes

2.6.1.4 Comparing CP-H/RDS to TE QoS Algorithms

The main advantage that the CP-H/RDS has over current TE QoS algorithms is that it is designed to use multiple QoS user constraints. All the current TE QoS algorithms only allow for the bandwidth user constraint to be used. Hence network carriers can only find paths that satisfy the bandwidth constraint which in turn forces bandwidth to be sold as a commodity [7]. This causes a situation where network carriers that have the lowest price per unit bandwidth get the most business. Some network carriers offer services tailored for each customer by using the DiffServ model that involves assigning a Per-Hop Behavior (PHB) [13] code to each customer. DiffServ routers in the core network use this code to sort packets into their corresponding treatment

classes, without having to know to which flows or what types of applications the packets belong. However, this approach has a major drawback in that the desired behavior of an application is typically specified by end-to-end parameters, while the DiffServ mechanisms defined per hop behaviour. The RDSS framework proposes to integrate the per-class approach of the DiffServ model and the intelligent path finding mechanism of the CP-H/RDS algorithm to offer a unique solution that meets the end-to-end QoS needs of the individual customer flows.

2.7 Problems associated with using QoS Routing in Core Networks

The problems associated with QoS routing in core networks include the problem of routing with imprecise state information and stability of paths found by the QoS routing algorithm. The following sections examine these problems and some possible solutions.

2.7.1 Routing algorithms with Imprecise State Information

In large networks, maintaining precise global network state information in the dynamic environment is extremely challenging. The three main factors that contribute to imprecise state information outlined in [51] are: infrequent link state update due to state update policy of the routing protocol; propagation delay of the link state update packets; and, hierarchical state aggregation. A number of QoS routing methods that tolerate imprecise state information have been proposed. These methods include safety-based routing [52], localized routing [53] and centralized server based QoS routing [54]. Each method of routing with imprecise state information is discussed below.

2.7.1.1 Safety-based Routing

The Safety-Based Routing (SBR) approach infers a range of the actual link state values from the link state updates, and finds the path that has the highest probability to satisfy a connection request. SBR assumes routing with bandwidth constraints and on-demand path computation. SBR computes the probability that a path can support an incoming bandwidth request. Assuming that links are mutually independent, the safety of a path can be determined as a product of the safety of the links in the path. Path safety can then be handled in exactly the same way as other path metrics.

In [52] two different routing algorithms are proposed for SBR. The first is the safest-shortest path algorithm that selects a path with the largest safety factor among the shortest path. The second algorithm, called the shortest-safest path selects paths with larger safety, and if more than one exists, the shortest one is chosen.

2.7.1.2 Localized Routing

Localized routing totally eliminates the impact of the imprecise global network state information by making routing decisions based on the information maintained locally at each router. Instead, source nodes infer the network QoS state based on flow blocking statistics collected locally, and perform flow routing using this localized view of the network state. In [53] a QoS routing model called proportional sticky routing (PSR), that assumes that the source node has network topology information from existing protocols like OSPF is used. The PSR algorithm proceeds in cycles of variable lengths. A number of cycles form an observation period. During the observation period, flows are routed, that is, paths are selected for connection requests, based on a parameter, called flow proportion, associated with each candidate path. In the meantime, the information of the flow blocking probability for each candidate path is collected. At the end of the observation period, a new flow proportion for each path is computed. The flow blocking probability of a path indicates the quality of the path

and is used to compute new flow proportions.

2.7.1.3 Centralized Server based QoS routing

A centralized server based QoS routing scheme is proposed in [54] that eliminates the concept of inaccurate network state information by changing the routing paradigm. In this routing scheme, routers are clients of the route server and send routing queries for each one of the incoming requests. The route server stores and maintains two data structures, the Network Topology Data Base (NTDB), which keeps the link state information for each link in the network, and the Routing Table Cache (RTC) that stores the computed path information.

This thesis proposes the RDSS framework that uses a centralized server based QoS routing model. In addition, SBR can be implemented by including an additional metric in the RDS algorithm that tracks the safety factor for each link.

2.7.2 Stability of QoS routing

QoS Routing is sensitive to changes in link state information which could lead to the problem of oscillation of traffic between paths. For example, if path selection relies on link state information about available bandwidth, it can easily lead to a situation where all the traffic is routed to a path with a lot of available bandwidth. This path will then be congested and the traffic will be routed to another path.

To help rectify this problem a reservation protocol can be used in conjunction with QoS routing to pin the network paths [55] [56]. In this way, even if some other path becomes a better choice, the connections already routed to a given path will not switch to the better path.

However, this thesis does not address the problem that QoS routing causes in terms of oscillating traffic paths. The RDSS framework uses route pinning indirectly since the MPLS protocol, which supports route pinning, is used as a means for setting

up explicit paths for flows.

2.7.3 Impact of QoS guaranteed traffic on best-effort traffic

A large percentage of the traffic flows in core networks is still best effort traffic. Since best effort traffic share paths with QoS traffic, whose priority is higher than best effort traffic, the performance of best effort traffic can be significantly affected.

The routing objectives in an environment with both QoS guaranteed traffic and best-effort traffic are: 1) Minimizing the call-blocking ratio of QoS flows; and, 2) Optimizing the throughput and fairness for best-effort flows. Since the first objective only considers QoS traffic and the second objective only best-effort traffic, this could lead to a contradiction.

The proposed RDS algorithm searches for a QoS path that optimizes individual traffic flow optimization requirements. As a result, this approach tends to reserve paths that have a lot of resources for QoS flows, while best effort flows are routed along paths that have less network resources. However, since both QoS guaranteed traffic and best-effort traffic compete for finite resources, this makes it difficult to formulate a solution that protects best effort traffic.

2.8 Summary

The metrics used in the QoS routing algorithms presented in this Chapter are summarized in Table 2.6. In addition, an organizational chart for QoS algorithms is provided in Figure 2.1.

Table 2.6: Routing Metrics for QoS Algorithms

Routing Algorithm	Metrics
Bandwidth-Delay-Constraint Algorithm	Bandwidth and Delay
Shortest-Widest Path Algorithm	Bandwidth and delay
Constraint Bellman-Ford Algorithm	Cost and Delay
Hassin's Algorithm	Cost and Delay
Delay-Constrained Unicast Routing (DCUR)	Cost and Delay
Ma and Steenkiste Algorithm	Bandwidth, loss, delay and jitter
Lagrange Relaxation	Cost and Delay
SSR + DCCR	Cost and Delay
Chen and Nahrstedt Algorithm	Multiple additive metrics
Jaffe Algorithm	Two additive metrics
Iwata Algorithm	Multiple additive metrics
Randomize Algorithm	Multiple additive metrics
H_MCOP Algorithm	Multiple additive metrics
SAMCRA	Multiple additive metrics
A*Prune	Multiple additive metrics

2.9 Conclusion

Several researchers have investigated the constraint-based path selection problem and proposed various algorithms, mostly heuristics. This chapter examined both generic and non-generic solutions to the MCP/MCOP problem. Many heuristic solutions are shown to have low performance in terms of finding a constraint path based on the user constraints. On the other hand, exact solutions have 100% success rates but perform poorly in terms of running times especially under loose constraints.

This chapter examined one possible application of QoS routing to online traffic engineering. The major difficulty of using QoS algorithms in a TE application is that global optimization is difficult to achieve since the flow sequence and therefore allocation of resources is random. In addition, most TE online algorithms do not consider user constraints apart from the concave bandwidth metric mainly due to the fact that routing with multiple additive metrics is an NP-complete problem. However,

if only the bandwidth metric is used in TE, then it will not be necessary to use QoS algorithms like the SAMCRA and A*prune since these algorithms were created to process multiple additive metrics. Chapter 6 introduces a new framework that allows both concave and additive metrics to be used in a TE environment.

Problems associated with the use of QoS routing in core networks were discussed and some of the current solutions examined.

Chapter 3

CP/RDS as a solution for MCP/MCOP problems

In [12] [4] it is argued that NP-complete behaviour of exact QoS routing algorithms only occurs in specially constructed graphs, which are unlikely to occur in realistic communication networks. This has significantly influenced the approach that this thesis takes in proposing an exact algorithm to solve the MCP/MCOP problem.

In this chapter the fundamental principles of exact algorithms are examined. A motivation is then presented for the CP and RDS algorithms developed. The pseudo code for the CP and RDS algorithms are presented. Finally, a mathematical analysis is presented for both the CP and RDS algorithms.

3.1 Principles of Exact Algorithms

An exact solution is guaranteed to give the correct feasible path based on the user request. Exact algorithms usually include the use of a nonlinear function, a k-shortest path algorithm and a mechanism to find non-dominant paths. These fundamental principles of exact algorithms are now presented.

3.1.1 Nonlinear Function

In [57] Jaffe uses a linear function search function. However, a non-linear function has a better chance of finding a feasible path because the solution space is larger than in the case of a linear function. This fact is illustrated in Figures 3.1 and 3.2. The circles represent constraint paths and each parallel line representing Equation 2.2, intercepts a solution. The first intercept represents the shortest path. As illustrated in Figure 3.1, the first intercepted line is outside the constraint area (defined by

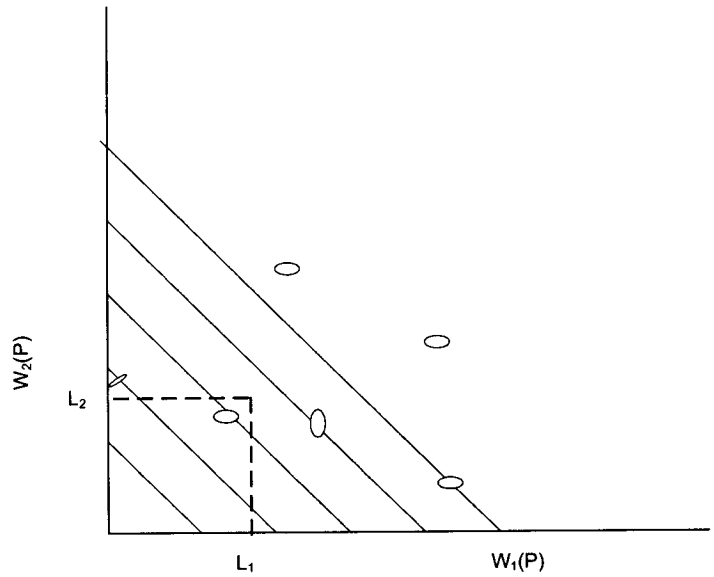


Figure 3.1: Linear function used in the search process. If the solution exists, the algorithm might not find it. Here the first path intercepted is not a feasible path.

rectangle formed by interception of L1 and L2 constraints) and is not a feasible path, and does not satisfy constraints L_1 and L_2. A non-linear function, as illustrated in Figure 3.2 can better intercept a feasible path because the solution space is larger than in the case of a linear function, covering the boundaries of the rectangular constraints. More importantly, the non-linear functions scan within the constraint area and so it is not possible to intercept a non-feasible path.

Dijkstra's algorithm relies on the fact that subsections of the shortest path are also shortest paths. Unfortunately, this property cannot be applied to non-linear functions and therefore using the length function in Equation 2.3 does not guarantee that the shortest path will be found. Hence most exact algorithms use the k-shortest path algorithm (discussed in 3.1.2) in an effort to find an optimal path satisfying the user constraints.

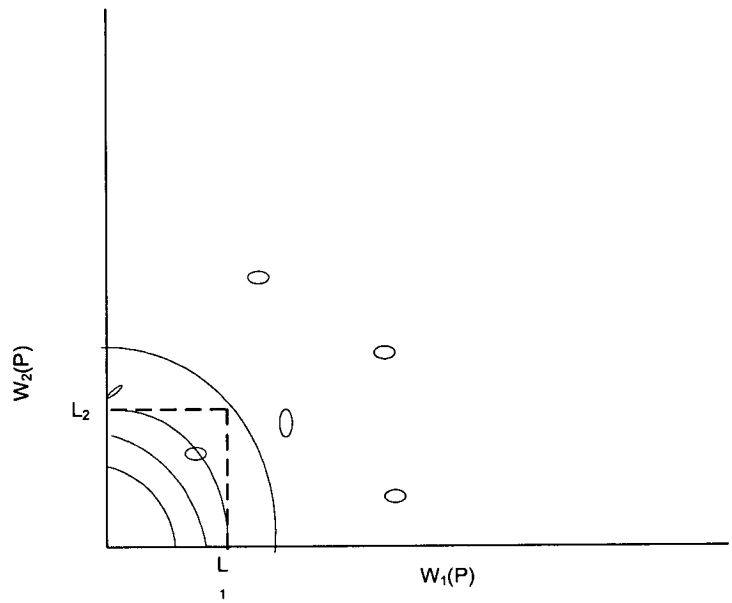


Figure 3.2: Non-linear function is used in the search process. Non-linear curves have a better chance at intercepting a feasible path.

3.1.2 The k-Shortest Path Algorithm

The k-shortest path algorithm [33] is essentially a version of Dijkstra's algorithm that stores the shortest, the second shortest, the third shortest path, and so on until the k-shortest path. Reference [9] states that the maximum number of paths between source and destination nodes on any graph is $e(N-2)!$ where $e = 2.72$. In addition, if the constraints or link weights have finite granularity (integers) a second maximum value of k is possible as given in Equation 3.1.

$$k_{\max} = \min \left[\frac{\prod_{i=1}^{m} L_i}{\max\limits_{1 \le i \le m} (L_i)}, \lfloor e(N-2)! \rfloor \right] \tag{3.1}$$

where,

$k_{\max}$ = maximum number of paths connecting source and destination nodes

L_i = value of the user constraints

N = number of nodes in the graph

The results of Equation 3.1 suggests that algorithms that use the k-shortest path algorithm can have very high worst case running times. This is because the number of paths that exists between two nodes in a graph can be very large.

3.1.3 Non-Dominant Paths

If the value of k in a k-shortest path based algorithm is unrestricted, then it is necessary to reduce the search space to increase the computational efficiency. One such search space reducing technique is that of finding non-dominant paths as introduced in [58]. A path Q is dominated by a path P if $Q_j \leq P_j$, $\forall\ j \in [1, m]$, where m is the number of metrics. If Q is not dominated by P then path P is said to be non-dominant. The SAMCRA algorithm only considers non-dominant paths in its search process which significantly reduces the search space used by the algorithm. However, this reduction of the algorithm's search space comes at the expense of additional processing time required to find non-dominant paths. Simulations in [59] show that finding non-dominant paths can reduce the overall running times of exact algorithms.

3.1.4 The Expense of Optimization in Exact Algorithms

The k-shortest path algorithm is responsible for the bulk of the computational and storage overhead of exact algorithms. Each node in the graph must have a queue size of k_{min}. This means that the maximum queue size is Nk_{min}. Therefore, to remove elements from the queue has a time complexity of $O(Nk_{min}log(Nk_{min}))$. In [4], the total worst case complexity for the SAMCRA exact algorithm is given as $O(Nk_{min}log(Nk_{min}) + k^2mE)$. Depending on the correlation link structure [12] and topology of the graph, the value of the queue size (k_{min}) can grow exponentially. This is illustrated in Figure 3.3 where a chain topology is shown. Each node of the graph

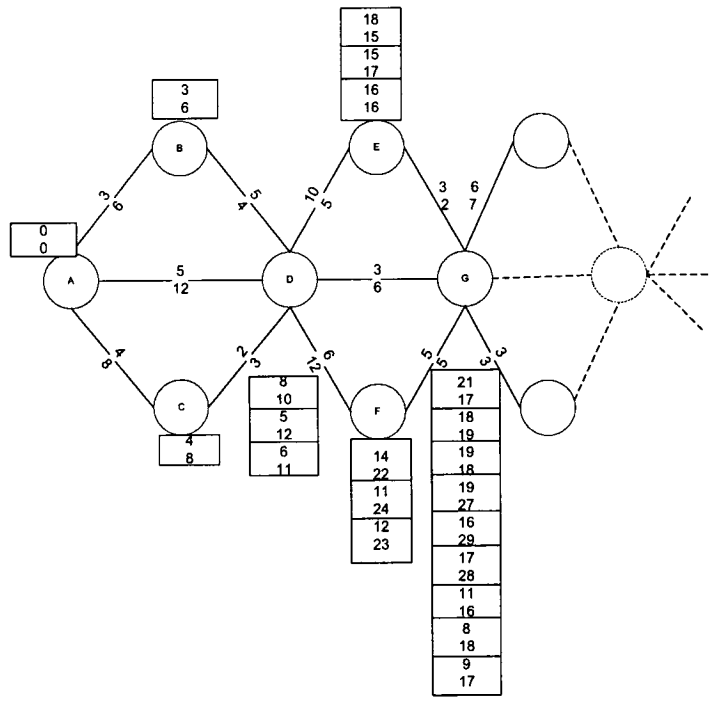


Figure 3.3: Chain Topology showing the exponential increase in a queue size at nodes along paths connecting A and G.

shows a queue in which elements store the values of metric 1 and metric 2 for all paths that can reach the node. Suppose that a route between source A and destination G is requested, then as illustrated, the storage capacity of the intermediate nodes required to store k-paths grows in an exponential fashion.

3.2 Motivation for a new Exact Algorithm

Most communication networks are designed for robustness and reliability, and as a consequence have redundant links resulting in chain like topologies. Although, the topology structure is not the only factor that determines NP-complete behavior, exact algorithms such as the SAMCRA and A*prune can experience NP-complete behavior when network topologies have chain like characteristics. SAMCRA uses

the k-shortest path algorithm, and therefore requires that all non-dominant paths between a source and a destination are determined before selecting the shortest path. An important point to note is that the SAMCRA algorithm attempts to find an optimal path whether or not the traffic flow requests optimization. That is, the optimization function is network based rather than flow based. However, since QoS routing is all about satisfying the constraints of the user, then the user should also have a say in the optimization preferences of the traffic flow.

The most important goal of a QoS algorithm is to find a path that satisfies all the user constraints. When the traffic flow does not request optimization and the QoS algorithm finds an optimal path optimization can be viewed as an extra unnecessary gift, which could waste network resources. In chain-like topologies where the time complexity is typically high, some users may wish to have a trade-off between an optimal path and any feasible path having a lower time complexity. In other words, users may wish to have an algorithm that is exact with respect to the MCP problem but heuristic with respect to the MCOP.

The Constraint Paths Routing Decision System (CP/RDS) presented in Sections 3.3 and 3.4 does not depend on a k-shortest path algorithm. To achieve this, the MCP solution mechanism is decoupled from the MCOP mechanism. Hence, there are two main components to this algorithm:

1. The CP Component

   This component finds a non-dominant subset of paths connecting source and destination nodes that meet the user network constraints.

2. The RDS Component

   This component uses a multi-criteria objective algorithm to decide which path from the set of paths resulting from component 1 best meet the user and network operator needs.

Figure 3.4 shows how the CP and RDS algorithms can be used to find a QoS route from a communication network (graph) given a set of user specified concave and additive metrics, and a set of policy based metrics from the network provider. The first step is to eliminate or prune all links not satisfying the specified concave metric(s). The second step is to find a set of constraint paths that are feasible with respect to the network additive metrics. The final step is to execute the RDS algorithm with the specified network and policy metrics, along with the constrained paths that resulted from step 2. The RDS algorithm calculates a preference ranking for each path based on the QoS requirements.

Remark 2 *The CP component is only concerned with network metrics such as delay, jitter and packet loss as opposed to policy metrics like security, safety factor and administrative color, while the RDS component is concerned with both network and policy metrics.*

Remark 3 *If the user flow does not require optimization there is no need to find a path that exceeds the user requirements and this practice could lead to the inefficient allocation of network resources which is similar to the problem that protocols like OSPF face with congestion in the shortest path. Thus even in the case of the MCP solution it is a waste of network resources to select any constraint path that satisfies the user constraints as the feasible path, a better solution is to select the path that is based on no optimization requirements or in the case where the flow has optimization requirements, select a path that optimizes these requirements.*

Separating the MCP solution from the MCOP solution makes it possible to have more control over finding a constraint path and the optimization process. For example, an objective function is not built into the path finding algorithm and this makes it possible for the RDS to accept flexible optimization requirements from user flows. This feature is not possible in other QoS algorithms without explicitly changing the

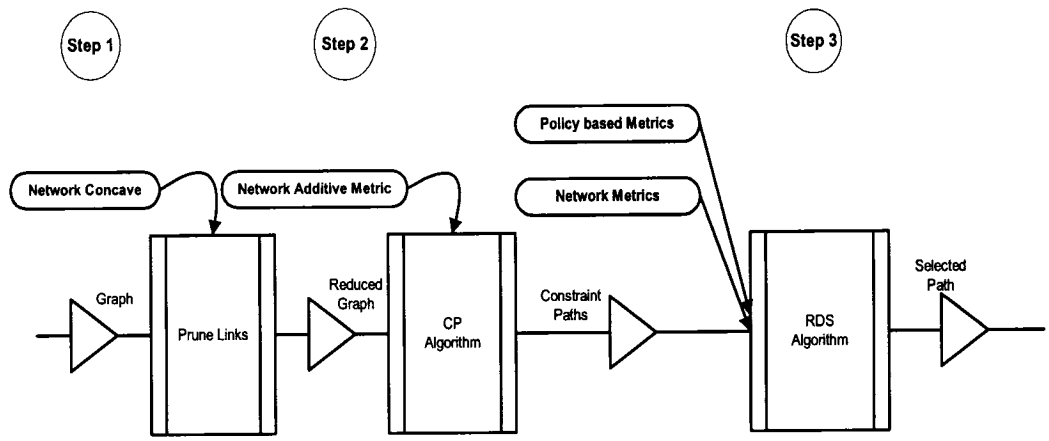


Figure 3.4: A pictorial view of the how CP/RDS algorithm processes network and policy metrics.

objective function. More significantly, both constraint-based and policy-based [60] measures can participate in the path selection process without involving policy-based link metrics in the graph search process. For example, the CP algorithm could find constraint paths based on delay and the RDS algorithm could optimize these paths in terms of jitter and security risks; this assumes that a security risk factor is administratively assigned to each link in the network.

The choice of user constraints can heavily influence how many feasible paths exist and therefore affect the execution time needed to find an optimal path. User constraints can be strict or loose [12]. The following definitions explain these terms further:

Definition 3 (Strict Constraints) *The set of strict user constraints from a set of paths P can be defined as follows:*

$L_j = w_j(P^*) + \varepsilon_j, \qquad j = 1, .., m$

where $P^*$is the path for which $\max_{1 \leq j \leq m} (w_j(P))$ is minimum and ϵ_j are small positive numbers relative to $w_j(P^*)$.

Definition 4 (Loose Constraints) *The set of loose user constraints from a set of*

paths P can be defined as follows:

$L_j = w_j(P^*) + \varepsilon_j, \qquad j = 1, .., m$

where P^* is the path for which $\max_{1 \leq j \leq m} (w_j(P))$ is maximum and ϵ_j are small positive numbers relative to $w_j(P^*)$.

The motivation for QoS Routing is rooted in the fact that multimedia applications require strict user constraints. Therefore the question is whether or not existing exact or heuristic QoS routing algorithms can address the needs of applications that have strict constraints. Based on the results in [2] it can be concluded that existing algorithms are inefficient especially under strict user constraints. Because the CP/RDS terminates in cases where it is computationally expensive to find a network optimal path the CP/RDS will lead to faster average running times than the SAMCRA and A*prune algorithms.

Most QoS algorithms focus on solving the MCP problem using additive metrics and hence optimize for such metrics. When a metric is concave, for example bandwidth, the first procedure [61] is to prune the network for all links that are smaller than the user requested bandwidth. If the user wishes to find a path that optimizes bandwidth from the network graph that remains after pruning, none of the current QoS routing algorithms addresses this issue. However, because the CP/RDS separates the MCP solution from the MCOP solution it is possible for the RDS to optimize any type or combination of metrics.

3.3 The Constraint-Path Algorithm

The Constraint Path (CP) algorithm finds a set or subset of constraint paths between source and destination nodes. The algorithm first finds a minimal hop constraint path between source and destination nodes by performing a Breadth First Search (BFS) that eliminates any sub-path that may violate any of the user constraints. This constrained path is then used as a bench mark to find other constraint paths that

are non-dominant with respect to it. The algorithm terminates when the queue size to find the minimal hop path reaches a specified threshold or when the queue is empty. Also, a heuristic is used to help reduce the time complexity that is involved with finding other constraint paths. In addition, the maximum number of constraint paths that can be returned by the algorithm is restricted.

The CP algorithm accepts a graph (G), source(s), destination (t), a set of user constraints (L) and returns a set of constraint paths (C). The data structure for the node consists of a path from the source to this node, the height of this node in the tree and a node identifier called the vertex. In addition, Q is a priority queue that first removes the node with the lowest height from the queue, and discoveredPaths is a counter that tracks the number of paths that have been found. The pseudo-code for the CP algorithm is given below.

Algorithm 1 *CP (G, s, t, C, L){*

```
1. node ←new Node(path ←Null, height← 0,vertex← s)
2. Q ←node
3. maxRatio ←1 {Global for CP and Relax}
4. while (Q) {
5.     u ← extract_min(Q)
6.     if (u = t) then
7.         C ← C ∪ u.Path
8.         discoveredPaths++
9.         if (C.size() > λ)
```

```
10.                 STOP
11.         end if
12.         else
13.  Relax (u)
14.  }
```

Algorithm 2 *Relax (closest) {*

```
1.  HEIGHT ← closest.height + 1
2.  for each neighbor u of closest {
3.          PATH ←closest.Path + u
4.          pathRatio ←max (PATH_j / L_j )
5.          if (pathRatio < maxRatio AND nonDominant (PATH)) {
6.                  node← new Node(path ←PATH, height← HEIGHT,vertex← u)
7.                  if (discoveredPaths = 0))
8.                          Q ←node
9.                  else
10.                 if (αN < queue.size())
11.                         Q ←node
12.     } (end if )
13.         if ( u = t)
```

14. $\quad\quad\quad$ *maxRatio* $\leftarrow$ *pathRatio*

15. $\quad\quad$ *if (Q.size()* $> \alpha N$ *and discoveredPaths = 0)*

16. $\quad\quad\quad\quad$ $\lambda \leftarrow 0$

17. } (for loop)

18. }

In the BFS source node is placed on the queue (Line 2). A node on the queue means that the node has been discovered. The CP algorithm then extracts the lowest height node from the queue, examines all its loop-free neighbors and places those that do not violate the user constraints on the queue (Relax algorithm). All nodes on the queue are reachable from the source without violating any user constraints.

The CP algorithm has a threshold value λ that is used to tune execution time performance of the algorithm at the expense of missing constraint paths that should be considered in the RDS optimization process. The λ threshold determines the maximum number of paths that the CP algorithm can pass to the RDS algorithm; λ is a small positive integer number (in our simulations $\lambda = 7$). There also exists another threshold value α that is a positive integer and controls the maximum number of queue entries allowed before λ is set to 0. The higher the value of α, the higher the probability that more paths will be in set C. However, higher values of α will mean that the CP algorithm will have longer running times.

The queue size, network size N, and α are used to determine whether or not the number of paths passed to the RDS algorithm should be restricted to 1 (lines 15-16 of the Relax algorithm). If the queue size required to find the first path exceeds αN then the CP algorithm returns only one path, say P_1, in an attempt to reduce long running times at the expense of finding an optimal path. If the queue size required to find the first path is less than αN, then only paths that are better than P_1 with respect to all

metrics would be considered as candidates for the next path to be searched (lines 13-14 of the Relax algorithm). Setting the maxRatio to the pathRatio of the constraint path P_1 guarantees that paths which are dominant are not considered (lines 5-12 of Relax algorithm). Similarly, after the second path, say P_2, is discovered only paths that are better than P_2 with respect to all metrics could be possible candidates for the next path P_3 to be searched, and so on. This is an important feature, also present in the SAMCRA algorithm, that assists with search space reduction. In addition, like the SAMCRA algorithm, the CP algorithm uses this search reduction technique of placing only non-dominant sub-paths on the queue.

If k_{min} is the number of constraint paths returned by the CP algorithm, then the queue can contain at most Nk_{min} nodes. When using a binary heap as the data structure for the priority queue, selecting the node with the minimum height among Nk_{min} different nodes takes at most a calculation time $O(1)$. Adding k_{min} nodes to the queue takes $O(Nk_{min}log(Nk_{min}))$. Calculating whether or not a path is a constraint path takes $O(m)$. The total worst-case complexity is $O(Nk_{min}log(Nk_{min}) + mE)$ for finding k_{min} feasible paths.

3.4 RDS algorithm

The RDS algorithm is concerned with finding an optimal path based on user requirements. The RDS algorithm does this by building preference functions [62] [11] for each metric from the constraint paths produced by the CP algorithm. In Goodridge et. al. ([63] [64] [65] [66]) introductions to this approach are presented. The basic idea is to use a preference function that accepts a network metric value say x as a parameter and returns a value, $s(x)$ between -1 and 1 that represents the preference value of x relative to all the network values for this metric. Figure 3.5 shows inputs and the output of the RDS algorithm. The peudocode for the RDS algorithm is given below.

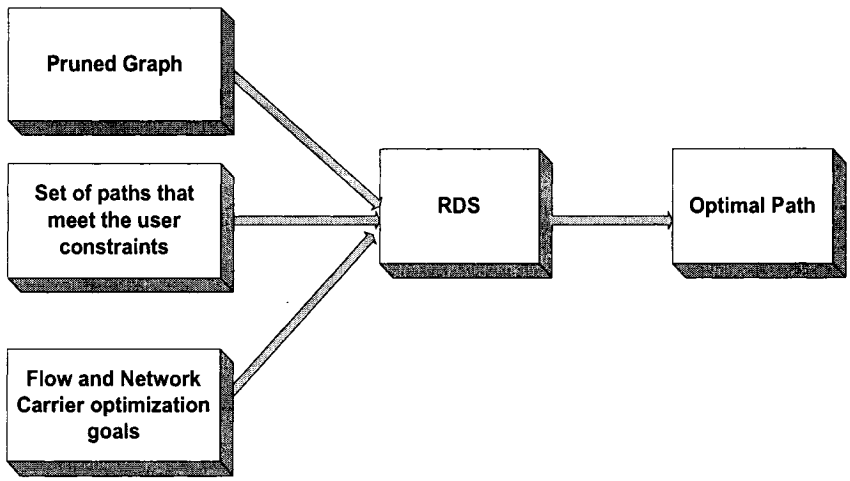


Figure 3.5: Shows the inputs and output of the RDS algorithm

Algorithm 3 *RDS (C, P, v) {*

1. $k \leftarrow$ *C.size()*
2. *if* $k = 1$ *then*
3. *Return* $P \leftarrow$ *C.get(0)*
4. *For* $j \leftarrow$ *0 to m Do {*
5. $a_j \leftarrow$ *BEST (C, j)*
6. $b_j \leftarrow$ *WORSE (C, j)*
7. *For* $i \leftarrow$ *0 to k Do {*
8. *if* $a_j \neq b_j$ *then*
9. $s_j(x_{ij}) \leftarrow 2\frac{(x_j - b_j)}{(a_j - b_j)} - 1$ *{Motivation given in Section 3.7.1}*
10. *else*

```
11.                 s_j(x_ij) ← 1
12.         }
13. }
14. Let s_j(d_j) ← −1, for all j ∈ [1, m]
15. Let sum_i ← 0, for all i ∈ [1, k]
16. For j ← 0 to m Do {
17.         If (v[j] = 1) then
18.         s_j(d_j) ← 1
19.         Else
20.         If (v[j] = 2) then
21.                 s_j(d_j) ← 0
22.         For i ← 0 to k Do
23.                 sum_i ← sum_i + s_j(d_j) × s_j(x_ij)
24. }
25. y ← LARGEST(sum)
26. return P ← C.get(y)
27. }
```

The algorithm accepts a list of constraint paths C, and a vector v of size m containing the user traffic optimization goals for each network metric. The RDS returns a path P that it calculates to meet the optimization goals of the user traffic.

In line 1 the $size()$ method of the list data structure returns the number of constraint paths and assigns the value to k. If k is 1 then the algorithm uses the $get()$ method of the list data structure to assign the path to P and halts the algorithm (line 3). Lines 4-13 of the algorithm build the preference function matrix $s_j(x_{ij})$ for $\forall\ j \in [1, m]$ and $\forall\ i \in [1, k]$.

On line 5, the function BEST accepts the list of constraint paths C and metric j, and returns the "best" value for metric j among all the paths in C. The best value for a concave metric is the maximum value among all paths, and the best value for an additive metric is the minimum value among all paths. Similarly, in line 6 the function worse returns the "worst" value for metric j among all the paths in C. Note that the worst value for a concave metric is the minimum value among all paths in C, and worst value for an additive metric is the maximum value among all paths. For multiplicative metrics the RDS will convert them to additive metrics as described in [67].

The array $s_j(d_j)$ stores scaled values for the user traffic flow demands. These values are set to -1 since the CP algorithm returns constraint paths better than or equal to the values of the user constraints. The elements of an array called sum of size k are set to 0. The elements of this array store $\sum_{j=1}^{m} s_j(x_{ij})s_j(d_j)$ for each path; lines 16-24 calculate this sum. Note that lines 17-21 are used to set the user demand array $s_j(d_j)$ to the optimization goals of the network traffic. An optimization goal of 1 means that user traffic wishes to optimize the given metric and a 2 means that no optimization is required.

The function LARGEST accepts the sum array and returns the path $y \in [1, k]$ that has the largest value. The algorithm uses the $get(y)$ method of the list data structure to assign the path located at y to P and the algorithm halts.

Remark 4 *The total worst case time complexity of the CP/RDS algorithm is* $O(N\lambda \log(N\lambda) + mE + m(\lambda + 1))$.

The space complexity of the CP/RDS is $O(\lambda mN)$

A formal definition for a pareto optimal point [42][68] is now examined.

Definition 5 (Pareto Optimal Point) *A point* $x^* \in C$ *is said to be (globally) pareto optimal or a globally efficient solution or a non-dominated or a non-inferior point for multiple objective optimization (MOP) if and only if there is no* $x \in C$ *such that* $f_j(x) \leq f_j(x^*)$ *for all,* $j \in \{1, 2, .., m\}$ *with at least one strict inequality. (Note the set* C *contains all constraint paths and* x *represents a vector containing path weights for each metric).*

Definition 6 (Global Pareto Optimal Set (GPO)) *If there exists no solution in the search space which dominates any member in the set* $\bar{O}$*, then the solutions belonging to the set* $\bar{O}$ *constitute a global pareto-optimal set.*

For a multi-criterion optimization problem, there is more than one objective function. This gives rise to a set of optimal solutions instead of one optimal solution. Each point "x" in Figure 3.6 represents a path in a communication network characterized by delay and jitter. The points that are circled are members of the GPO set and are said to be non-dominant paths (see Section 3.1.3). Note that the paths that the CP algorithm returns may or may not include a pareto point. However, it will be shown in Chapter 4, that under strict constraints the CP algorithm has a very high success rate of including a pareto optimal point in the set of paths that is passed to the RDS algorithm.

3.5 CP/RDS Demonstration

This section gives three examples of how the CP/RDS work. The first example involves only additive metrics while the second example involves bandwidth and two additive metrics. The third example is a variation of the second example that involves dynamic allocation of the bandwidth resource.

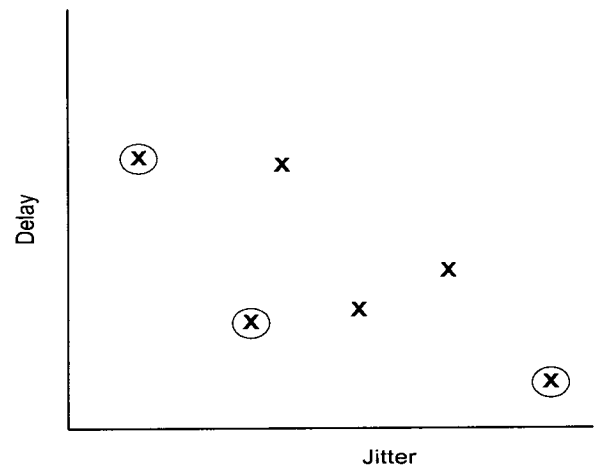


Figure 3.6: Each point marked x represents a path in a given graph G(V;E) and is characterized by delay and jitter. The circled points are pareto optimal and are members of the GPO set.

3.5.1 Additive metrics Example

To illustrate how the CP/RDS works, consider the network shown in Figure 3.7. Each link is characterized by three independent weights that correspond to delay, cost and jitter respectively. Table 3.1 shows a number of objectives that network traffic could request of the network under loose user constraints for delay and jitter.

The first step uses the CP algorithm to find a set of possible constraint paths based on the given user constraints. These paths are A-D-G, A-B-D-G, A-D-E-G, A-C-D-G, A-B-D-E-G, and A-C-B-E-G. Note that in practice the CP algorithm would only return a subset of the list given above due to the effect of lines 13-14 (Relax algorithm) which guarantees that only non-dominant paths with respect to the minimal hop path are considered. However, for simplicity and for illustration purposes of how the RDS works, it is assumed that all the paths given in the above list are returned. These paths are then passed to the RDS algorithm, which addresses goals 1 to 5. For the given constraint paths, the matrix in Figure 3.8 shows the value of each metric on each

Table 3.1: Example of goals user flows may request

Goal No.	User request description
1	A cost of no more than 22 units
2	A cost of no more than 22 units and minimizes hop count
3	A cost of no more than 22 units and minimizes delay and hop count
4	A cost of no more than 22 units and minimizes jitter and hop count
5	A cost of no more than 22 units and minimizes delay, jitter and hop count

of the six routes that connect A and G. Figure 3.9 shows the matrix after preference function scaling.

Table 3.2 shows a summary of the values for vector v, the product of $A\overline{v}$ and the path that the algorithm selected. Note that $s_j(d) = 1$ means that optimization of the jth metric is desired, $s_j(d) = 0$ means that optimization for the jth is not an important goal of the user traffic, and $s_j(d) = -1$ means that optimization of the jth metric is not desirable or that the worst path for the jth metric is desirable. Since the CP algorithm only returns constraint paths based on specified users constraints it follows that all user requests would result in $s_j(x) = -1$ (no path value for a given constraint can be worse than the specified user constraints).

When examining the results of the product $A\overline{v}$, it is seen that path A-D-E-G is the best path for a user demanding at most 22 units of cost. This is because the value for cost on this path is 19 units, which is closest to 22 units. The RDS does not over allocate the cost resource so path A-B-D-G, which has a cost of 16 units,

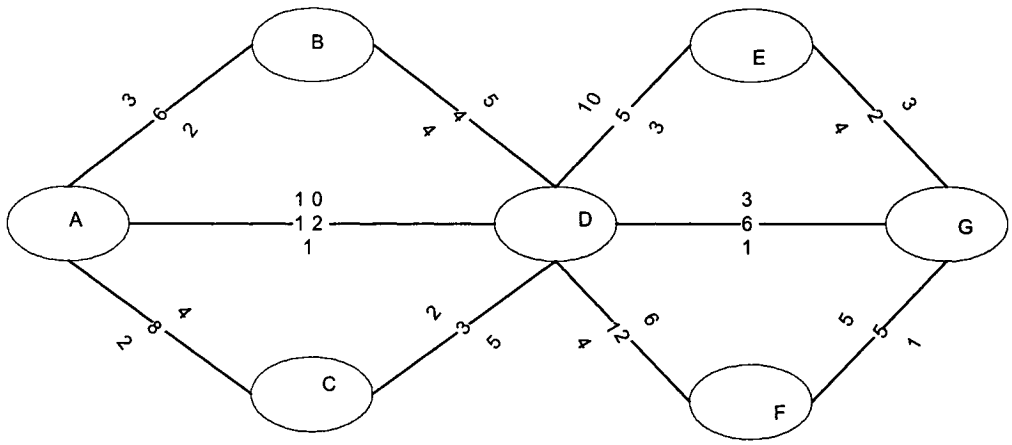
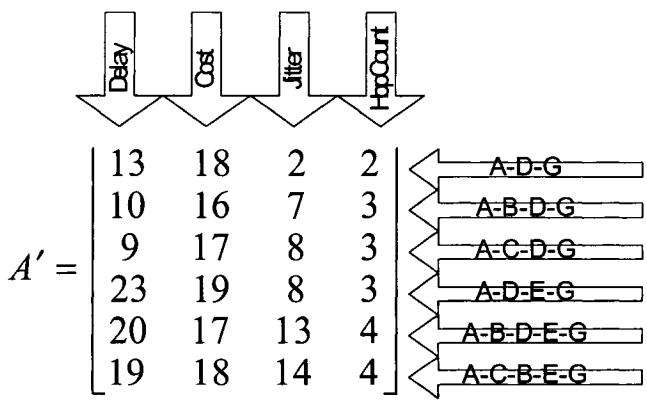


Figure 3.7: Example to illustrate how the CP/RDS works. Each link of the graph has delay, cost and jitter values



$$A' = \begin{bmatrix} 13 & 18 & 2 & 2 \\ 10 & 16 & 7 & 3 \\ 9 & 17 & 8 & 3 \\ 23 & 19 & 8 & 3 \\ 20 & 17 & 13 & 4 \\ 19 & 18 & 14 & 4 \end{bmatrix}$$

Figure 3.8: Matrix containing columns of metric values for each network path (rows)

$$A = \begin{bmatrix} 0.43 & -0.33 & 1.00 & 1.00 \\ 0.86 & 1.00 & 0.17 & 0.00 \\ 1.00 & 0.33 & 0.00 & 0.00 \\ -1.00 & -1.00 & 0.00 & 0.00 \\ -0.57 & 0.33 & -0.83 & -1.00 \\ -0.43 & -0.33 & -1.00 & -1.00 \end{bmatrix}$$

Figure 3.9: Matrix A after preference function scaling has been performed.

Table 3.2: Paths selected based on user goals

#	$\overline{v}$	$A\overline{v}$
1	$[0, -1, 0, 0]$	$[0.33, -1.0, -0.33, 1.0, -0.33, 0.33]$ A-D-E-G (y = 4)
2	$[0, -1, 0, 1]$	$[1.33, -1, -0.33, 1, -1.33, -0.67]$ A-D-G (y = 1)
3	$[1, -1, 0, 1]$	$[1.76, -0.14, 0.67, 0, -1.9, -1.1]$ A-D-G (y = 1)
4	$[0, -1, 1, 1]$	$[2.33, -0.83, -0.33, 1.0, -2.16, -1.67]$ A-D-G (y = 1)
5	$[1, -1, 1, 1]$	$[2.76, 0.03, 0.67, 0, -2.73, -2.1]$ A-D-G (y = 1)

is not selected. For goal 2, path A-D-G is the closest path to 22 units that satisfy a minimum hop count of 2. Goals 3, 4 and 5 involves multiple optimization and path A-D-G is a pareto optimal path since it is the only path for which at least one metric, hop count, is minimum with all the other metric values being relatively small. It is noted that the RDS is concerned with finding the best path from among the paths received from the CP algorithm based on the user optimization needs. For naturally conflicting metrics like delay and jitter the RDS returns the path that is best suited for the user optimization goals.

Table 3.3 shows the paths that the CP/RDS, SAMCRA and Iwata algorithms produce for given optimization goals. It is clear that the CP/RDS path selection process is flow based since different flow optimization goals resulted in different paths being selected. This is not the case with the SAMCRA and Iwata algorithms.

3.5.2 Bandwidth and two Additive Metrics

Figure 3.10 shows the topology of the network used in this example. Each link is characterized by three weights that correspond to bandwidth, delay and cost respectively. Table 3.4 shows a number of objectives that network traffic could request of

Table 3.3: Comparing paths selected for CP/RDS, SAMCRA and Iwata algorithms based on given optimization goals. The network in Figure 3.7 is used.

Flow	Optimization		CP/RDS	SAMCRA	Iwata
1	Optimize delay Optimize cost Optimize jitter Optimize hop count	no no no no	Any path	A-D-G	A-C-D-G
2	Optimize delay Optimize cost Optimize jitter Optimize hop count	yes no no no	A-C-D-G	A-D-G	A-C-D-G
3	Optimize delay Optimize cost Optimize jitter Optimize hop count	yes yes no no	A-B-D-G	A-D-G	A-C-D-G
4	Optimize delay Optimize cost Optimize jitter Optimize hop count	yes yes yes no	A-B-D-G	A-D-G	A-C-D-G
5	Optimize delay Optimize cost Optimize jitter Optimize hop count	yes yes yes yes	A-D-G	A-D-G	A-C-D-G

the network under loose user constraints for delay and cost. In addition, for simplicity we assume that each request has a bandwidth of 7 Mbps.

The first step prunes the network of all links that are less than 7 Mbps. The CP-H algorithm is then used to find a set of possible constraint paths based on the given user constraints. These paths are A-D-G, A-C-F-G, and A-B-C-F-G. These paths are then passed to the RDS algorithm, which addresses goals 1 to 5. For the given constraint paths, the matrix in Figure 3.11 shows the value of each metric on each of the three routes that connect A and G. Figure 3.12 shows the matrix after preference function scaling.

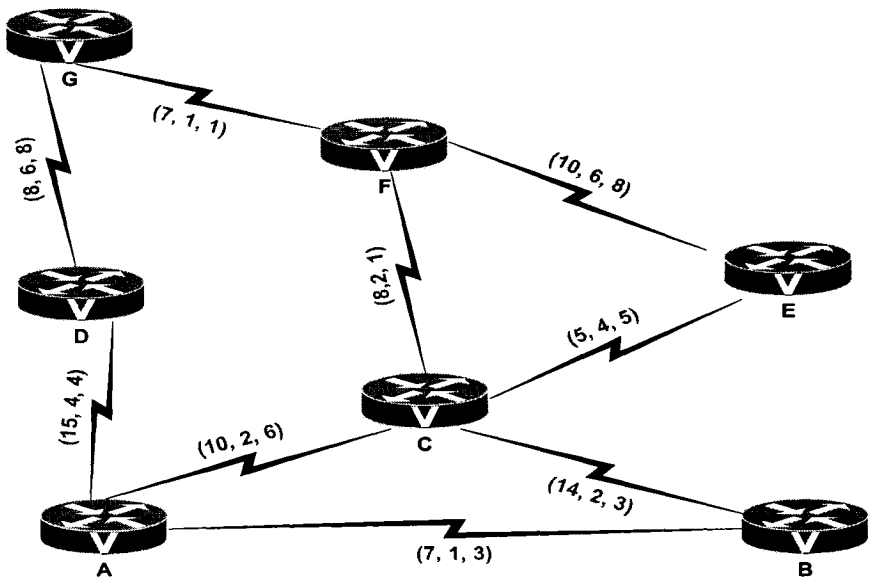


Figure 3.10: Example to illustrate how the CP/RDS works. Each link of the graph has delay, cost and jitter values

$$A' = \begin{array}{c} \begin{array}{ccc} \text{Bandwidth} & \text{Delay} & \text{Cost} \end{array} \\ \begin{bmatrix} 8 & 10 & 12 \\ 7 & 5 & 8 \\ 7 & 6 & 8 \end{bmatrix} \end{array} \begin{array}{l} \\ \text{ADG} \\ \text{ACFG} \\ \text{ABCFG} \end{array}$$

Figure 3.11: Matrix containing columns of metric values for each network path (rows)

$$A = \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ -1 & 0.6 & -1 \end{bmatrix}$$

Figure 3.12: Matrix A after preference function scaling has been performed.

Table 3.4: Example of goals user flows may request

Goal No.	User request description
1	Delay of 13 ms, no optimization
2	Delay of 13 ms, delay optimized, no cost optimized
3	Delay of 13 ms, bandwidth optimized
4	Delay of 13 ms, delay optimized
5	Delay of 13 ms, all optimized

Table 3.5: Paths selected based on user goals

#	$\overline{v}$	$A\overline{v}$	SAMCRA path
1	$[0, -1, 0]$	$[1, -1, -0.6]$ (y = 1) A-D-G	A-C-F-G
2	$[0, -1, 0]$	$[-1, 1, 0.6]$ (y = 2) A-C-F-G	A-C-F-G
3	$[0, -1, 1]$	$[0, -2, -1.6]$ (y = 3) A-B-C-F-G	A-C-F-G
4	$[1, -1, 0]$	$[2, -2, 0.4]$ (y = 1) A-D-G	A-C-F-G
5	$[1, -1, 1]$	$[-1, -1, 0.6]$ (y = 3) A-B-C-F-G	A-C-F-G

Table 3.5 shows a summary of the values for vector v, the product of $A\overline{v}$ and the path that the algorithm selected. When examining the results of the product $A\overline{v}$, it is seen that path A-D-G is the best path for a user demanding a delay of 13 ms and no optimization. This is because the value for delay on this path is 10 ms, which is closest to 13 ms. The RDS does not over allocate the delay resource, and therefore the other paths that have better delay are ignored. For goal 2, path A-C-F-G is selected which optimizes the delay metric. Goal 3 requests that bandwidth be optimized and the algorithm selects path A-D-G which has the highest bandwidth. The path selected for goal 4 is A-D-G which minimizes delay. Finally, the path selected for goal 5 is A-B-C-F-G which is a global pareto optimal path.

Table 3.5 also shows the corresponding paths that the SAMCRA algorithm produces for the given optimization goals. It is clear from these results that the SAMCRA algorithm is not flow given.

3.5.3 Dynamic Bandwidth Example

This example uses the network topolopy in Figure 3.10. However, as the network allocates flows into the network, each link bandwidth is adjusted to reflect the new link bandwidth. Table 3.6 shows requests for two flows. The columns of the table include matric A, matic $A'$and the output for the CP/RDS and SAMCRA algorithms.

Table 3.6: Paths selected based on dynamic bandwidth allocation scheme

#	Optimization	A	A'	CP/RDS	SAMCRA
1	Bandwidth 6 Optimize delay yes Optimize cost yes	$\begin{bmatrix} 8 & 10 & 12 \\ 7 & 5 & 8 \\ 7 & 6 & 8 \end{bmatrix}$	$\begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ -1 & 0.6 & -1 \end{bmatrix}$	A-D-G	A-C-F-G
2	Bandwidth 1 Optimize delay yes Optimize cost yes	$\begin{bmatrix} 1 & 10 & 12 \\ 1 & 5 & 8 \\ 1 & 6 & 8 \\ 1 & 13 & 20 \end{bmatrix}$	$\begin{bmatrix} 1 & 0.25 & 0.34 \\ 1 & 1 & 1 \\ 1 & 0.75 & 1 \\ 1 & -1 & -1 \end{bmatrix}$	A-C-F-G	A-C-F-G

3.6 Mathematical Analysis of the CP algorithm

In this section it is shown that the CP algorithm is an exact algorithm with respect to the MCP problem.

To prove that the CP algorithm is an exact algorithm it is neccessary to appreciate that the CP algorithm is a modification of the All-Paths algorithm. Consider the All-Paths algorithm below.

Algorithm 4 *All-paths(G, s, t, L)*

```
1. node ←new Node(path ←Null, height← 0,vertex← s)
2. Q ←node
3. while (Q) {
4.     closest ← extract_min(Q)
5.     if (closest.vertex == t)
6.         PRINT (node.path)
7.     for each neighbor u of closest {
8.         PATH ←closest.Path + u
9.         node← new Node(path ←PATH, height←closest.height + 1,vertex←
   u)
10.        if(PATH is loop-free)
11.            Q ←node
12.    }
13. }
```

Remark 5 *The All-paths algorithm uses the same data structures and variable names as used in the CP algorithm.*

Theorem 1 *The All-paths algorithm finds all the paths that exist between source s and destination t.*

Proof. In [69] it is shown that the BFS discovers every node that is reachable from the source node s. To discover a node[1] in the context of the all-path algorithm and

[1]The term node in this context does not refer to the Node data structure used in the CP algorithm. It is simply a vertex in a graph.

CP algorithms means that the node is placed on the queue. Without modifications the BFS algorithm works as follows: All links are characterized by a single metric that has the value 1. Initially all nodes are colored white except for the source node, which is colored grey. A grey color means that the node has been discovered, whereas white means that it has not. The algorithm now extracts a grey-colored node with smallest length and discovers all its white neighbors, which are then colored grey. Once all neighbors of a node are discovered the node is colored black. The BFS algorithm continues until all nodes are black.

The all-paths algorithm behaves like the BFS-algorithm except that instead of examining only the white neighbors of a grey-colored node, it examines all of its loop-free neighbors (line 7). Given a destination t, the BFS algorithm is guaranteed to find a path connecting s and t provided that t is reachable. Because the all-paths algorithm is an extension of the BFS there is at least one path to each node from s. Since each node examines all its loop-free neighbors, all possible combinations are examined, which corresponds to finding all loop-free paths (line 6). ■

Theorem 2 *The CP algorithm finds at least one constraint path between source s and destination t provided that there exist such a path.*

Proof. Let us assume that at least one path between source s and destination t exists that satisfy the user constraints. The CP algorithm can be seen as a modified all-paths algorithm. However, the CP algorithm only considers paths that are within the user constraints (line 5 of Relax algorithm). Given the result of the previous theorem, this means that the CP algorithm returns at most the number of constraint paths between s and t.

If the queue size grows beyond the size αN [2] before any path between s and t is found, the threshold value of λ is set to 0. After the first path is found, the CP

[2]see Relax algorithm

algorithm checks to see if $\lambda = 0$ and if it is the algorithm is halted. Hence, the CP algorithm finds at least one constraint path between source s and destination t provided that there exist such a path. ■

A consequence of the Theorem 2 is that the CP algorithm always solves the MCP problem, since at least one path satisfying user constraints is guaranteed to be found.

3.7 Mathematical Analysis of the RDS algorithm

Many decision analysis problems involve the trade-off of conflicting decision criteria. Multi-Attribute Utility Theory (MAUT) [70] is a commonly used methodology in decision analysis for solving conflicting decision criteria. However, MAUT is very difficult to use because it utilizes non-intuitive concepts of subjective probabilities which are difficult to measure. This thesis uses concepts of Preference Function Modelling (PFM) proposed in [71] for reconciling competing decision criteria.

Section 3.7.1 gives a background of scales and preference functions followed by Section 3.7.2 which describes the use of scales and preference functions in the RDS algorithm. Finally, Section 3.7.3 outlines a set of conditions under which multiple optimization can be achieved using the RDS algorithm.

3.7.1 Preference Functions

A scale s is a function that maps elements from a set A to a set F where F is a field [72]. A field is a set with two operations of addition and multiplication. Four different types of scales are discussed in [73]. These include Ratio Scales, Interval Scales, Ordinal Scales and Implications. The purpose of measurement is to enable mathematical operations on scale values. In [3] three types of measurement models are defined which include weak, proper and strong models. The three models are summarized in Table 3.7.

The following definitions are very important to the development of the technique

Table 3.7: Measurement Models. Adopted from [3]

Type	Applicability	Requirements
Weak	None	No straight line is defined
Proper	Linear Algebra	Equality, addition and multiplication
Strong	Order and calculus	Equality, addition and multiplication, order, completeness

used in the RDS algorithm for evaluation of multi-criteria objective functions.

Definition 7 (Affine Space) *[74]*

An affine space is a triplet of sets (P, V, F) together with associated operations as follows. The pair (V, F) is a vector space. The elements of P are termed points and two functions are defined on P: a one-to-one and onto function $h : P \rightarrow V$ and the "difference" function $\Delta : P^2 \rightarrow V$ that is defined by $\Delta(a, b) : h(a) - h(b)$.

Remark 6 *A homogenous field is a one-dimensional affine space.*

Remark 7 *If an affine space is one-dimensional and $a, b, c, d \in P$, then there exists a scalar α such that $\Delta(a, b) = \alpha\Delta(c, d)$ and we denote:*

$$\frac{\Delta(a, b)}{\Delta(c, d)} \in F$$

Remark 8 *In a homogenous field the set P is termed a straight line.*

Remark 9 *Formally the operations of addition and multiplication are not defined on P.*

Before a formal definition of a preference function is given, it is helpful to give the definition of a value function [71] which is used to represent a decision maker's preference.

Definition 8 (Value Function) *Let $A \subseteq R^m$ be a the evaluation space i.e. the set of alternatives under consideration. A function $s : A \rightarrow R$ which assigns a real number $s(x)$ to each alternative x in A is called a value function if:*

1. x_1 "preferred to or indifferent to" $x_2 \iff s(x_1) \geq s(x_2)$
2. x_1 "indifferent to" $x_2 \iff s(x_1) = s(x_2)$

Definition 9 (Preference Function) *[71] [75]*

A preference function has a domain set D consisting of multiple criteria $x = (x_1, x_2, ., x_m)$ where x_j represents the jth criteria. Each alternative is a point $q = (q_1, q_2, .., q_k)$ in the set D. Building a preference function involves assessment of numerical values of the coordinates $(q_1, q_2, .., q_k)$ for each alternative. The overall (multi-criteria) preference function f for x:

$$f(x_1, x_2, ..., x_m) \in \mathbb{R} \tag{3.2}$$

The following must hold:

1. x "is preferred to" $y \iff f(x) \geq f(y)$
2. $x - z$ "is preferred to" $y - z \iff f(x) - f(z) \geq f(y) - f(z)$

Remark 10 *A preference function is conceptually similar to a value function. However, a preference function is concerned with ordering both preference and difference.*

Definition 10 (Proper Scale) *A proper scale must satisfy the fundamental equation of measurement.*

Definition 11 (Fundamental Equation of Measurement (FEM)) *[3] Let $s : A \rightarrow F$ be a scale. The Fundamental Equation of Measurement that states:*

$$\frac{\Delta(a,b)}{\Delta(c,d)} = \frac{\Delta(s(a), s(b)))}{\Delta(s(c), s(d))} \tag{3.3}$$

where $a, b, c, d \in P$ and $\Delta(s(a), s(b))) = s(a) - s(b)$ denotes scalar subtraction in set F.

Definition 12 (Strong Scales) *A strong scale must be a proper scale and must satisfy the following conditions [3]:*

1. F must be an ordered set.

2. F must be complete.

Preference must be modeled by affine scales. "Proper affine scales map points in an empirical homogeneous field over the scalar field F into a mathematical homogeneous field. Each scale $s : A \rightarrow L$ is a function that maps a subset A of an empirical straight line into a mathematical straight line L (these scales are the coordinate functions of objects)" [3].

3.7.2 Preference Function Formulation of the RDS Algorithm

In the RDS algorithm let x_{ij} be the value for metric j on route i, and d_j be the value of the traffic flow demanded for metric j. The algorithm uses a proper scale $s_j(x)$, given in Equations 3.4 and 3.5 to map values of a given metric j into the domain $[-1, 1]$, where x_j is the metric value for metric j, a_j is the best value for metric j, and b_j is the worst value for metric j. Note that Equation 3.4 is used for additive/multiplicative metics and Equation 3.5 is used for concave metrics. The RDS algorithm creates a set of points y_i (network routes) by applying the function given in 3.6.

$$s_j(x) = \left\{ \begin{array}{cc} 2\frac{(x-b_j)}{(a_j-b_j)} - 1 & a_j \neq b_j \\ 1 & a_j = b_j \\ -1 & x < b_j \\ 1 & x > a_j \end{array} \right\} \text{(Addtive Metrics)} \tag{3.4}$$

$$s_j(x) = \left\{ \begin{array}{ll} 2\frac{(x-b_j)}{(a_j-b_j)} - 1 & a_j \neq b_j \\ 1 & a_j = b_j \\ 1 & x > a_j \\ -1 & x < b_j \end{array} \right\} \text{(Concave Metrics)} \tag{3.5}$$

$$y_i = \sum_{j=1}^{m} s_j(x_{ij}) s_j(d_j) \tag{3.6}$$

Note: $s_j(x_{ij}) \in [-1, 1] \subseteq \mathbb{R}$ and $s_j(d_j) \in [-1, 1] \subseteq \mathbb{R}$.

Theorem 3 *The scale $s_j(x)$ given in 3.4 is a strong scale where $x \in [b, a]$, a is the best value for metric j and b is the worst value for metric j and $j \in [1, m]$.*

Proof. All proper models must satisfy the FEM given in Equation 3.3. For the left hand side where $a \neq b$:

$$\frac{\Delta(p,q)}{\Delta(u,v)} = \frac{p-q}{u-v}$$

for $p, q, u, v \in [b, a]$

For the right hand side:

$$\frac{\Delta(s(p), s(q))}{\Delta(s(u), s(v))} = \frac{\left[\frac{2(p-b)}{a-b} - 1\right] - \left[\frac{2(q-b)}{a-b} - 1\right]}{\left[\frac{2(u-b)}{a-b} - 1\right] - \left[\frac{2(v-b)}{a-b} - 1\right]}$$

$$= \frac{p-q}{u-v}$$

Since the right hand side evaluates to the same expression as the left hand side this implies that the FEM is obeyed. This means that s is a proper scale. Since the range of s range is $[-1, 1] \in \mathbb{R}$, it follows that s is a strong scale. ■

The RDS algorithm outlined in Section 3.4, gives the formulation of the preference ranking for a given set of feasible paths by building a scaled network matrix x_{ij}. Let A be the matrix x_{ij} and $\overline{d}$ be a $1 \times m$ vector that stores the scaled requested user demands $d_1, d_2, .., d_m$. The product of A and $\overline{d}$ produces a vector $\overline{y}$.

$$\overline{y} = A\overline{d} = \begin{pmatrix} s_1(x_{11}) & s_1(x_{12}) & .. & s_1(x_{1m}) \\ s_1(x_{21}) & s_1(x_{22}) & .. & s_1(x_{2m}) \\ .. & .. & .. & .. \\ s_1(x_{k1}) & s_1(x_{k2}) & .. & s_1(x_{km}) \end{pmatrix} \begin{pmatrix} s_1(d_1) \\ s_2(d_2) \\ ... \\ s_m(d_m) \end{pmatrix}$$

Ordering of the elements of $\overline{y}$ results in a preference ordering of the set of feasible paths based on the user demands and what the network can offer at the time.

Remark 11 *Let y_h, y_l be the largest and smallest entries of the vector $\overline{y}$ respectively. Now $s_j(x_{ij})s_j(d_j)$ represents route's i preference for metric j multiplied by the flow demand preference for metric j. Since this product can never be greater than 1, then $y_h = \sum_{j=1}^{m} s_j(x_{ij})s_j(d_j) \leq m$, and $y_l = \sum_{j=1}^{m} s_j(x_{ij})s_j(d_j) \geq -m$. When $y_h = m$ this intuitively means that all metrics fully satisfy both the network preference and the flow demand preference. However, when $y_l = -m$ then this means that when $s_j(x_{ij}) = -1$ then $s_j(d_j) = 1$ or vice versa for all metrics.*

The intuitive meaning for $s_j(x_{ij}) < 0$ multiplied by $s_j(d_j) < 0$ is that the flow demand preference for metric j is very low as well as the network preference for metric j on route i. Hence the resulting positive number is an indication that both the network and flow demand preference agree that route i is not good for metric j.

Hence a $[-m, m]$ bound exists for $\sum_{j=1}^{m} s_j(x_{ij})s_j(d_j)$ with $-m$ satisfying no demands and m satisfying all demands. In addition the elements of vector $\overline{y}$ are members of $\mathbb{R}$ which implies that order can be applied to the elements. Hence the closer the value of $\sum_{j=1}^{m} s_j(x_{ij})s_j(d_j)$ is to m means more demands of the $\overline{d}$ vector are satisfied. That is, the route position y_h where $y_h \in [-m, m]$, corresponding to the highest value of the vector $\overline{y}$ will satisfy most of the user constraints of the flow demand $\overline{d}$. Similarly, the route position y_l, corresponding to the lowest value of the vector $\overline{y}$ will be the worst choice in terms of satisfying the requirements of the flow demand $\overline{d}$.

A key question that this thesis has not addressed is: why does $s : A \rightarrow F$ have a range of $[-1, 1]$? The answer to this question is that s is chosen so that the range is

symmetrical. However, the range of s could be $[-k, k]$ where $k \in Z^+$. That is, the magnitude of scale value for the best metric is the same as the magnitude of the scale value for the worst metric. For example, if the path P_1 has the best bandwidth of 50 Mbps and path P_2 has the worst bandwidth of 20 Mbps then the absolute scale value for both paths are the same, and path P_1 is the additive inverse of path P_2 with respect to bandwidth. This is necessary so that meaningful multiplication and addition can be performed on the set $[-k, k]$. For example, $s_j(x_{ij}) < 0$ multiplied by $s_j(d_j) < 0$ results in a positive product.

Suppose that $s : A \rightarrow [0, k]$. Recall that RDS returns a route that matches the scaled user request demand as closely as possible. However, when a flow request d containing values such that $s_j(d_j) \leftarrow 0$, then it follows that $\sum_{j=1}^{m} s_j(x_{ij})s_j(d_j) \leftarrow 0$. This means that highest value in the output vector $\overline{y}$ will most likely not correspond to the route that matches the flow demand. Hence the RDS algorithm will not hold if the range of s is $[0, k]$.

3.7.3 Optimization Process in the RDS Algorithm

A set of theorems relating to the optimization process of the RDS algorithm is now presented.

Theorem 4 *From the set of constraint paths passed to the RDS algorithm, finding a path that optimizes a single metric is possible by setting $s_a(d_a) = 1$ for metric $a \in [1, m]$, the desired metric to be optimized.*

Proof. Let A be a $k \times m$ matrix define as follows:

$$A = \begin{pmatrix} s_1(x_{11}) & s_1(x_{12}) & .. & s_1(x_{1m}) \\ s_1(x_{21}) & s_1(x_{22}) & .. & s_1(x_{2m}) \\ .. & .. & .. & .. \\ s_1(x_{k1}) & s_1(x_{k2}) & .. & s_1(x_{km}) \end{pmatrix},$$

If the metric to be optimzed is $a \in [1, m]$ then all other metrics concerned are set to $s_j(d_j) = 0$, that is $j \neq a$.

Let $\overline{d}$ be defined such that $s_j(d_j) = 0$ for all $j \neq m$,and $s_j(d_j) = 1$ for $j = m$:

$$A\overline{d} = \begin{pmatrix} s_1(x_{11}) & s_1(x_{12}) & .. & s_1(x_{1m}) \\ s_1(x_{21}) & s_1(x_{22}) & .. & s_1(x_{2m}) \\ .. & .. & .. & .. \\ s_1(x_{k1}) & s_1(x_{k2}) & .. & s_1(x_{km}) \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ ... \\ 1 \end{pmatrix} = \begin{pmatrix} s_1(x_{1m}) \\ s_1(x_{2m}) \\ ... \\ s_1(x_{km}) \end{pmatrix}$$

Now if,

$y = \max\limits_{1 \leq j \leq m} (\overline{d})$

It is clear that the route associated with y contributes the most to the mth metric. Hence setting $s_j(d_j) = 1$ for any metric $j \in [1, m]$ would result in a path that optimizes the set of constraint paths received from the CP algorithm for the given metric. ■

Theorem 5 *If a pareto optimal path with respect to a given set of metrics say A, exists in the set of paths produced by the CP algorithm, then the RDS algorithm will find this path if all the desired user metrics in set A are set as follows* $s_j(d_j) = 1$.

Proof. Let $s_j(d_j) = 1$ for all $j \in A$ and y_h be the maximum of $y_i = \sum_{j=1}^{m} s_j(x_{ij}) s_j(d_j) = \sum_{j=1}^{m} s_j(x_{ij})$ for all i.

If route y_h is not pareto optimal then there is another route i for which $s_j(x_{ij}) \geq s_j(x_{hj})$ for all j and the inequality is strict for at least one j then $y_i > y_h$ which is a contradiction. ■

3.8 Conclusion

In this chapter the motivation for introducing CP/RDS as a new QoS Routing algorithm was examined. One of the advantages is that the algorithm adapts to user traffic optimization needs without requiring a change to its core logic or its length function. This routing paradigm searches for feasible paths satisfying multiple QoS

requirements and implements a routing decision system that separates the constraint path finding mechanism from the optimization mechanism. The use of preference functions makes it possible for the CP/RDS algorithm to return an optimal path for flows with optimization requirements. In addition, this mechanism allows concave and additive metrics to be optimized simultaneously.

The later sections of this chapter focused on the design of the CP and RDS algorithms. A comprehensive Mathematical analysis of the CP is given and it was shown that the CP algorithm is an exact algorithm. This result is significant since a user can be guaranteed that a path that meets the specified constraints will be returned, provided that such a path exists. Another significant result that was verified is that the CP algorithm only returns 1 constraint path connecting source and a destination node in cases where the algorithm encounters high space and time complexities.

The RDS algorithm was shown to always find a pareto optimal path provided such a path exists in the set of constraint paths. This result is significant since the purpose of the RDS is to find optimal paths based on user optimization goals.

Chapter 4

Performance Analysis of the CP/RDS Algorithm

In this chapter simulations to investigate answers to the following questions relating to the CP/RDS algorithm are reported:

1. What values of α should be used in the CP algorithm?
2. What is the success rate of the CP/RDS algorithm?
3. What are the relative normalized execution times of the CP/RDS with respect to both strict and loose user constraints under varying network topologies and sizes?
4. What are the normalized execution times of the CP/RDS as a function of the number of additive metrics?
5. What is the average number of paths returned by the CP algorithm?
6. What is the CP/RDS success rate at returning a pareto optimal path for flows requesting optimization goals?

All algorithms used in simulations have been implemented in Java and have used the Binary Heap [76] data structure. The BRITE [77] graph generator tool to generate Waxman graphs [41] (See Appendix B.1) is used. The configuration parameters used for generating the Waxman graphs are given in Table 4.1. The value for both the α and β are in $[0.0, 1.0]$ range. Note that the larger the value of α the greater the number of edges in the graph, while the larger the value of β the larger the ratio of long edges to short edges. The value of α and β are chosen so that graphs containing chain like topologies (see Section 3.1.4) are likely to be generated.

Table 4.1: Configuration of Brite tool to generate Waxman graphs used in simulations. Appendix B.3 gives further explanation of Brite parameters.

Parameter Name	Value
Network Size	N
Size of main plane	1000
Size of inner plane	100
Node placement	1
Growth type	1
Number of neighboring nodes each new node connects to	3
α (Waxman parameter)	0.10
β (Waxman parameter)	0.7

4.1 Appropriate values for α in the CP algorithm

The CP algorithm is designed to return only one feasible path when the number of nodes on the priority queue grows beyond αN (lines 15-16 of CP algorithm). The objective is to select an α that would result in the smallest execution times possible while at the same time ensuring that a reasonable number of paths k ($k < \lambda$) are passed to the RDS algorithm for ranking. To study the effect that α has on algorithm execution times and the number of paths returned by the CP algorithm, 3 sets of graphs are generated, each set containing 200 Waxman graphs consisting of 20, 60 and 100 nodes. For each graph set, the normalized execution time[1] and number of paths produced by the algorithm are recorded for each α in the set $\{1, 2, 3, 4, 5\}$. Table 4.2 shows the simulation results.

For $N = 20$, as the value of α increases the execution times of the algorithm remained constant, while the average number of paths returned by the algorithm increased. Hence for graphs of size 20, setting $\alpha = 5$ is more desirable than setting $\alpha = 1$ since more paths are passed to the RDS algorithm for best path ranking

[1]See Section 4.3.1 for explanation of normalized execution time.

Table 4.2: Normalized execution times and number of paths for given network sizes and alpha values

N	α	Normalized execution time	Average number of paths returned
20	1	1.30	1.69
	2	1.34	2.31
	3	1.34	2.62
	4	1.35	2.70
	5	1.35	2.70
60	1	1.90	1.69
	2	1.91	2.32
	3	1.92	2.6
	4	2.3	2.7
	5	2.4	2.7
100	1	3.14	2.36
	2	3.83	2.36
	3	9.98	2.36
	4	18.0	2.39
	5	21.9	2.39

without any additional increase in algorithm execution time.

For $N = 60$, as the value of α increases the execution times of the algorithm gradually increased until $\alpha = 3$, while the average number of paths returned by the algorithm increased. For $\alpha > 3$ execution times of the algorithm are higher than for $\alpha = 3$, while the average number of paths returned by the algorithm remained unchanged. Hence for graphs of size 60, setting $\alpha = 3$ would result in the best trade-off between execution time and number of paths produced by the algorithm.

For $N = 100$, as the value of α increases the execution times of the algorithm increased, while the average number of paths returned by the algorithm remained constant for $1 \leq \alpha \leq 3$ and increased slightly for $\alpha > 4$. Hence for graphs of size 100, setting $\alpha = 1$ is very desirable since execution times are reduced while at the same time returning a high number of paths to the RDS algorithm for best path ranking.

Remark 12 *For simulations that involve graphs with 100 or more nodes $\alpha = 1$ will*

be used in the CP algorithm. If the graph size is less than 100, an $\alpha = 3$ will be used..

4.2 Success Rate of the CP/RDS

The success rate is defined as the ratio of the number of user requests satisfied using a given algorithm divided by the total number of requests generated [2]. To study the success rate of the CP/RDS algorithms 4 sets of graphs were generated, each set containing 2000 Waxman graphs consisting of 100, 200, 300, and 400 nodes. Two independently correlated link weights ($m = 2$) were considered. Also 500 random sets of strict user requests for each graph in each set were generated and used for simulations involving the CP/RDS, SAMCRA and Jaffe algorithms. Figure 4.1 shows the success rate plotted as a function of the network size for strict user constraints. The simulation was repeated for loose user constraints and Figure 4.2 shows the results.

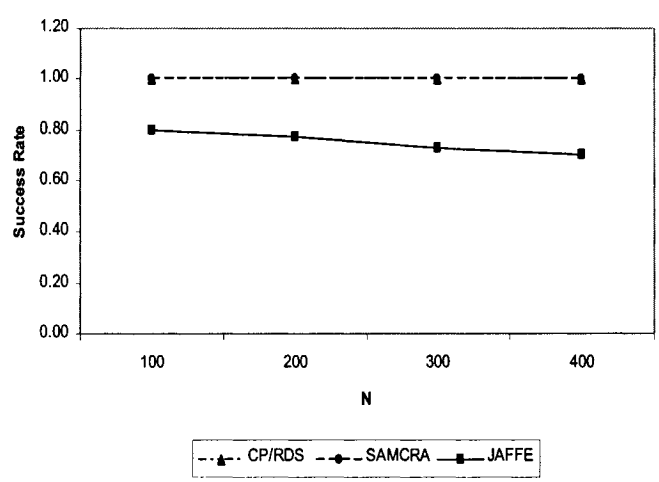


Figure 4.1: Success rate for the class of Waxman graphs as a function of nodes (N) ($m = 2$ and strict constraints)

The CP/RDS algorithm is guaranteed to return a path that meets the user constraints. This is demonstrated in Figure 4.1 where 100% success rates were recorded for large graphs under strict user constraints. These results confirm that the CP/RDS algorithm, like the SAMCRA algorithm, is exact with respect to solving the MCP problem. Under loose constraints all algorithms have 100% success rates. This is because the definition of loose constraints implies that many feasible paths exist in the given algorithm's search space, and therefore the chance of the algorithm finding a feasible path is very high.

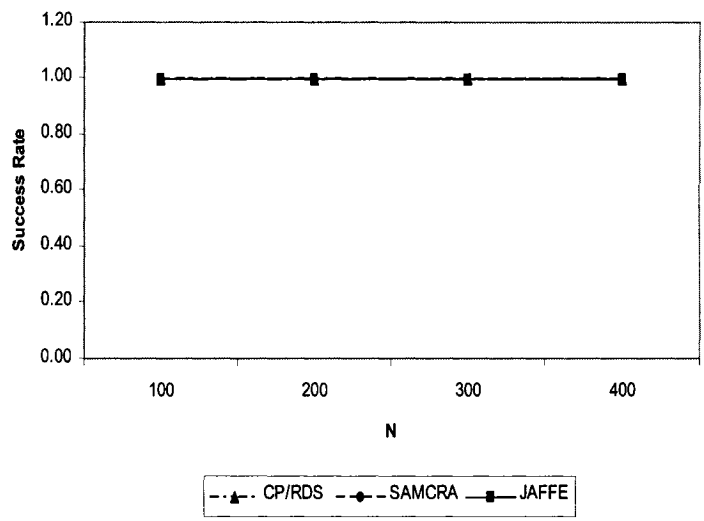


Figure 4.2: Success rate for the class of waxman graphs as a function of nodes (N) ($m = 2$ and loose constraints)

4.3 Normalized Execution times of the CP/RDS

4.3.1 Waxman and Barabasi Topologies

The normalized execution time of an algorithm is defined as the execution time of the algorithm over all iterations divided by the execution time of the least-delay path

Dijkstra's algorithm [2]. The normalized execution time is used to obtain machine independence. To study the normalized execution time of the CP/RDS algorithm, 5 sets of graphs, each set containing 2000 Waxman graphs containing 50, 100, 200, 300 and 400 nodes were generated. Each link in all graphs had two independent link weights uniformly distributed in the range [0, 100]. Figures 4.3 and 4.4 show the normalized execution times plotted as a function of the network size for strict and loose user constraints respectively. This similation was repeated for the Barabasi class of graphs and Figures 4.5 and 4.6 show the results for the Barabasi class of graphs for strict and loose user constraints respectively. Each normalized time plotted for a given graph size and algorithm was derived from the average normalized time over 2000 trials. The 95% confidence intervals associated with the data for Figures 4.3, 4.4, 4.5 and 4.6 are given in Tables 4.3, 4.4, 4.5 and 4.6 respectively. The confidence levels with the other graphs are given in Appendix D.

Remark 13 *The strict constraints are set by choosing the range of* $\varepsilon_j = [0, 0.5w_j(P^*)]$, *for all,* $j \in \{1, .., m\}$.

The execution times of both the SAMCRA and the CP/RDS are high relative to the Jaffe algorithm. This is attributed to the fact that SAMCRA and CP/RDS are exact with respect to finding a constraint path and therefore would naturally have higher execution times since a larger search space is involved. In addition, the BRITE configuration parameters are selected in such a way that the generated Waxman graphs have a good chance of having chain-like topologies similar to real networks. As a consequence, large algorithm queue sizes are likely to be associated with finding a feasible path. As illustrated in Figures 4.3/4.4 and Figures 4.5/4.6 the CP/RDS algorithm has lower running times than the SAMCRA algorithm under strict and loose user constraints for both the Waxman and Barabasi graph topologies. This difference can be accounted for by the fact that the CP/RDS applies a heuristic to halt the algorithm if the first minimum-hop constraint path is expensive to find in

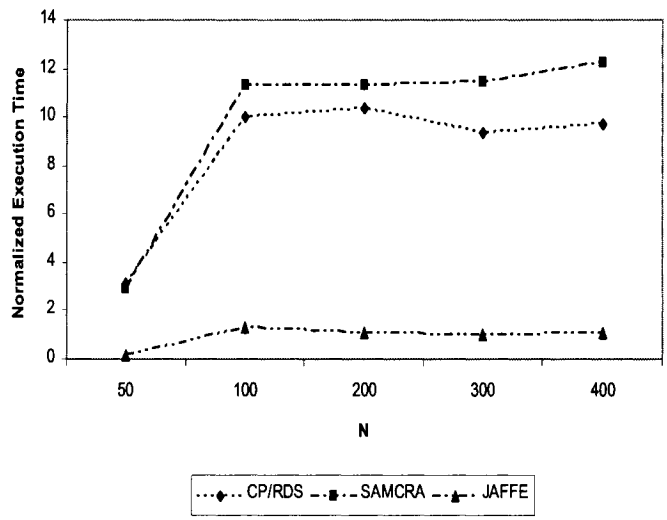


Figure 4.3: Normalized execution time for the class of Waxman graphs as a function of number of nodes for strict constraints ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)).

Table 4.3: 95% Confidence Intervals for mean execution times of Figure 4.3.

N	Algorithm	Mean time	95% Confidence intervals
50	CP/RDS	3.09	(2.75, 3.43)
	SAMCRA	2.93	(2.58, 3.27)
	Jaffe	0.18	(0.059, 0.215)
100	CP/RDS	10.02	(9.38, 10.65)
	SAMCRA	11.33	(10.98, 12.49)
	Jaffe	1.30	(1.042, 1.55)
200	CP/RDS	10.37	(9.85, 10.99)
	SAMCRA	11.32	(12.35, 13.91)
	Jaffe	1.12	(0.99, 1.25)
300	CP/RDS	9.38	(8.92, 9.83)
	SAMCRA	11.43	(10.75, 12.11)
	Jaffe	1.02	(0.93, 1.11)
400	CP/RDS	9.75	(9.19, 10.30)
	SAMCRA	12.24	(11.33, 13.14)
	Jaffe	1.124	(1.03, 1.23)

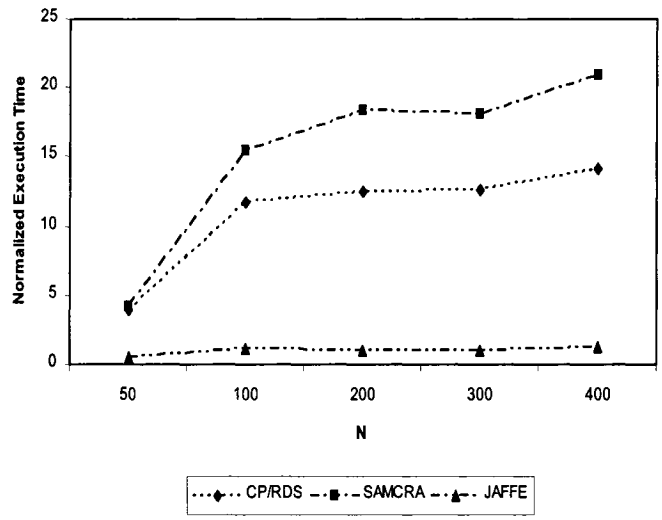


Figure 4.4: Normalized execution time for the class of Waxman graphs as a function of number of nodes for loose constraints ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)). 95% confidence intervals for each mean normalized time is given in Appendix D.

Table 4.4: 95% Confidence Intervals for mean execution times of Figure 4.4.

N	Algorithm	Mean time	95% Confidence intervals
50	CP/RDS	4.00	(3.62, 4.40)
	SAMCRA	4.19	(3.78, 4.59)
	Jaffe	0.47	(0.32, 0.61)
100	CP/RDS	11.73	(11.12, 12.33)
	SAMCRA	15.43	(14.63, 16.23)
	Jaffe	1.21	(0.97, 1.44)
200	CP/RDS	12.54	(12.10, 12.98)
	SAMCRA	18.32	(17.42, 19.21)
	Jaffe	1.08	(0.94, 1.21)
300	CP/RDS	12.56	(12.14, 12.97)
	SAMCRA	18.13	(17.15, 19.10)
	Jaffe	1.05	(0.97, 1.14)
400	CP/RDS	14.15	(13.68, 14.61)
	SAMCRA	20.90	(19.20, 22.58)
	Jaffe	1.22	(1.14, 1.30)

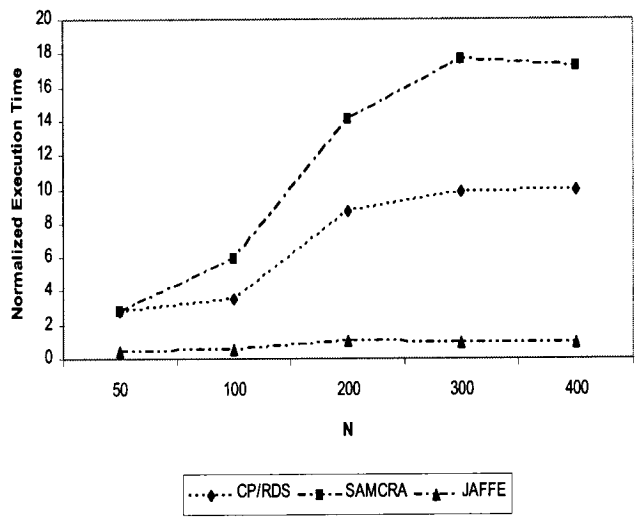


Figure 4.5: Normalized execution time for the class of Barabasi graphs as a function of number of nodes for strict constraints ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)).

Table 4.5: 95% Confidence Intervals for mean execution times of Figure 4.5.

N	Algorithm	Mean time	95% Confidence intervals
50	CP/RDS	2.78	(2.48, 3.08)
	SAMCRA	2.80	(2.50, 3.11)
	Jaffe	0.46	(0.33, 0.59)
100	CP/RDS	3.55	(3.28, 3.82)
	SAMCRA	5.92	(5.49, 6.34)
	Jaffe	0.55	(0.44, 0.67)
200	CP/RDS	8.71	(8.24, 9.17)
	SAMCRA	14.06	(13.12, 15.00)
	Jaffe	1.00	(0.86, 1.13)
300	CP/RDS	9.87	(9.33,10.42)
	SAMCRA	17.61	(16.27, 18.94)
	Jaffe	0.98	(0.88, 1.08)
400	CP/RDS	9.91	(9.39, 10.43)
	SAMCRA	17.18	(15.96, 18.40)
	Jaffe	0.95	(0.87, 1.03)

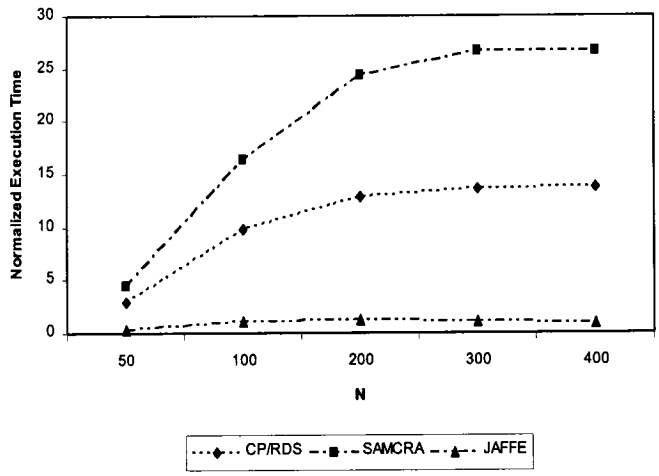


Figure 4.6: Normalized execution time for the class of Barabasi graphs as a function of number of nodes for loose constraints ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)).

Table 4.6: 95% Confidence Intervals for mean execution times of Figure 4.6.

N	Algorithm	Mean time	95% Confidence intervals
50	CP/RDS	2.96	(2.66, 3.26)
	SAMCRA	4.49	(4.12, 4.86)
	Jaffe	0.34	(0.23, 0.45)
100	CP/RDS	9.76	(9.20, 10.33)
	SAMCRA	16.35	(15.48, 17.23)
	Jaffe	1.13	(0.90, 1.35)
200	CP/RDS	12.80	(12.29, 13.30)
	SAMCRA	24.39	(23.09, 25.69)
	Jaffe	1.12	(1.05, 1.35)
300	CP/RDS	13.70	(13.17, 14.21)
	SAMCRA	26.65	(25.12, 28.19)
	Jaffe	1.08	(0.98, 1.19)
400	CP/RDS	13.70	(13.17, 14.23)
	SAMCRA	26.59	(25.12, 28.05)
	Jaffe	0.86	(0.79, 0.94)

terms of the number of entries in the queue. The SAMCRA algorithm does not halt after reaching the destination node the first time but continues in search of a pareto optimal path.

Remark 14 *The 95% confidence intervals for strict and loose constraints suggest that when $N \geq 100$ the execution times for the CP/RDS algorithm and SAMCRA are statistically distinguishable. This is because there is no overlapping of confidence intervals. However, when $N = 50$ the CP/RDS and the SAMCRA algorithms are statistically similar in terms of execution time.*

The simulation execution times of the SAMCRA and CP/RDS algorithms under strict constraints are high relative to the results in [2]. This is attributed to the fact that the authors of [2] considered strict paths with $\varepsilon_j = 0$ for all j. This means that given a set of strict user constraints only one path in the entire search space can satisfy these constraints. The authors reasoned that by forcing $k = 1$ allowed them to fairly compare MCP and MCOP algorithms. However, in reality strict user constraints are hardly precise enough that only one feasible path is possible. Hence by generating strict constraints in the range $[w_j(P^*), w_j(P^*) + \varepsilon_j]$ for $\varepsilon_j \geq 0$ for all j, this thesis presents a more realistic simulation model than the model presented in [2] with respect to assessing the true execution times associated with exact algorithms. The choice of setting $\varepsilon_j = 0$ for all j has a significant impact on the results in [2] and to illustrate this impact, the simulation for strict user constraints under the Waxman graph topology was repeated with $\varepsilon_j = 0$ for all j, rather than $\varepsilon_j > 0$. Figure 4.7 shows the normalized execution times plotted as a function of the network size, N, for strict user constraints when $\varepsilon_j = 0$ for all j. From the graph it can be seen that the SAMCRA and the CP/RDS have lower normalized execution times that are polynomial with respect to an the increase in network size. This result for the SAMCRA execution times is now consistent with results in [2].

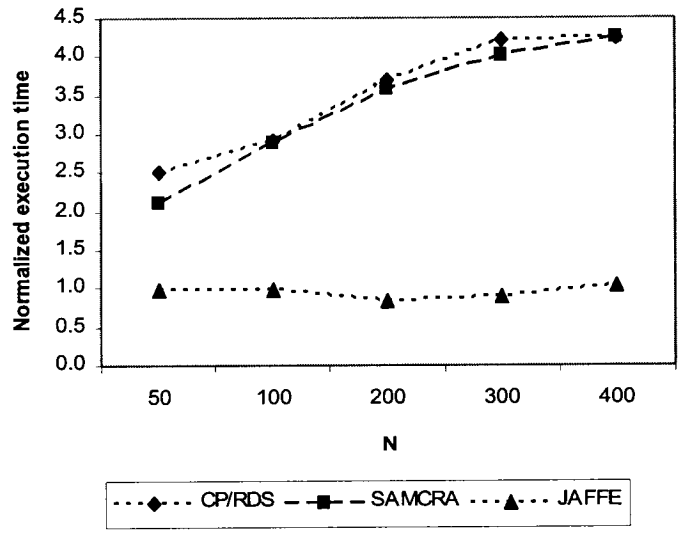


Figure 4.7: Normalized execution time for the class of Waxman graphs as a function of number of nodes for strict constraints ($m = 2$ and ($\varepsilon_j = 0.0$))

4.3.2 Lattice Topology

Lattice graph topologies are inherently difficult[12]. This is because a lattice graph structure has dense chain like topologies. To study the normalized execution time of the CP/RDS algorithm in lattice topologies, 4 sets of graphs, each set containing 200 lattice graphs containing 16, 25, 36 and 49 nodes was generated. Each link in all graphs has two positively correlated link weights uniformly distributed in the range [0, 100]. Figures 4.8 and 4.9 shows the normalized execution times plotted as a function of the network size for strict and loose user constraints respectively, when the link metrics are independent.

Remark 15 *Two link weights are positively correlated if they share the same mean weight from a given uniform distribution. On the other hand, two links are negatively correlated if one of the weights is selected from a uniform distribution with a small*

mean while the other is selected from another uniform distribution with large mean. In addition, if there is no specific correlation structure, the link weights are assumed to be independent.

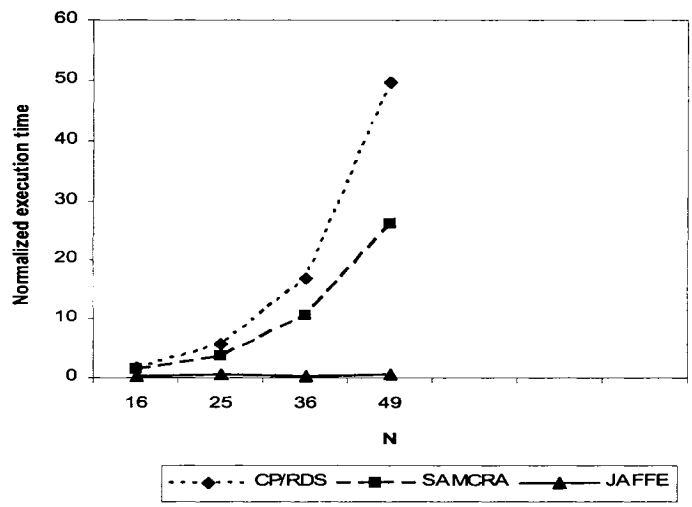


Figure 4.8: Normalized execution time for the class of Lattice graphs as a function of number of nodes for strict user constraints. Here an independent positive corelated link structure is used. ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$))

For the lattice graph topology the SAMCRA algorithm has lower running times than the CP/RDS algorithm for both strict and loose constraints as illustrated in Figures 4.8 and 4.9. These results can be explained by the fact that the lengths of paths connecting a given pair of nodes in a lattice topology are long and roughly the same size. Hence a BFS requires a large number of entries in the queue before reaching the first minimum hop path. On the other hand, the SAMCRA algorithm search process is driven by a length function which is topology independent and therefore more likely to find the destination node quicker than a BFS algorithm.

To illustrate the impact that link correlation structure has on the execution times of algorithms the previous simulation to study the execution times of the CP/RDS in

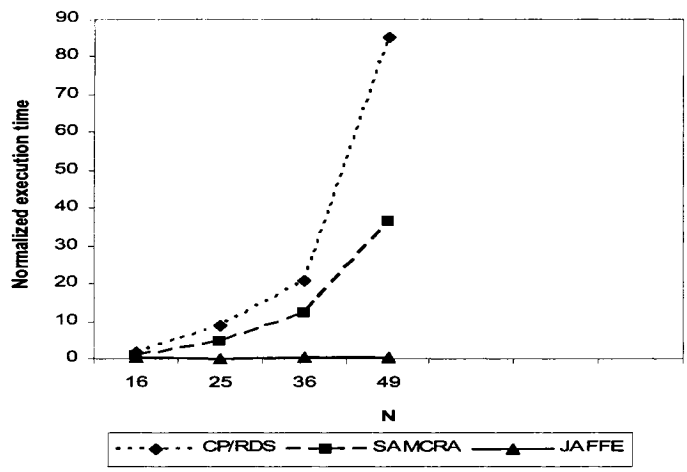


Figure 4.9: Normalized execution time for the class of Lattice graphs as a function of number of nodes for loose constraints. Here an independent positive correlated link structure is used. ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$))

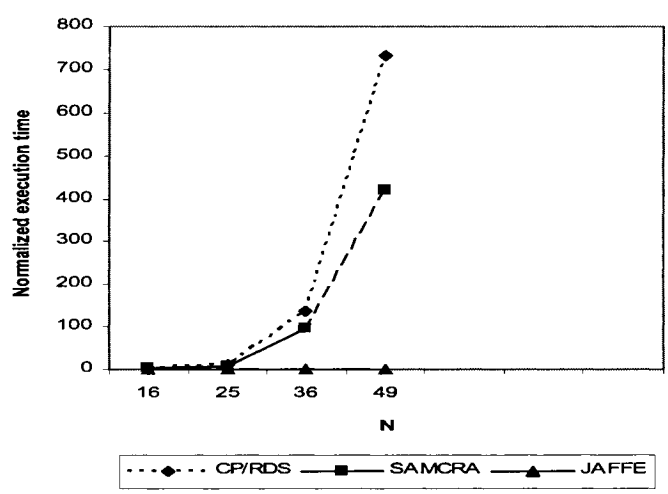


Figure 4.10: Normalized execution time for the class of Lattice graphs as a function of number of nodes for strict constraints. A negative correlated link structure is used. ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$))

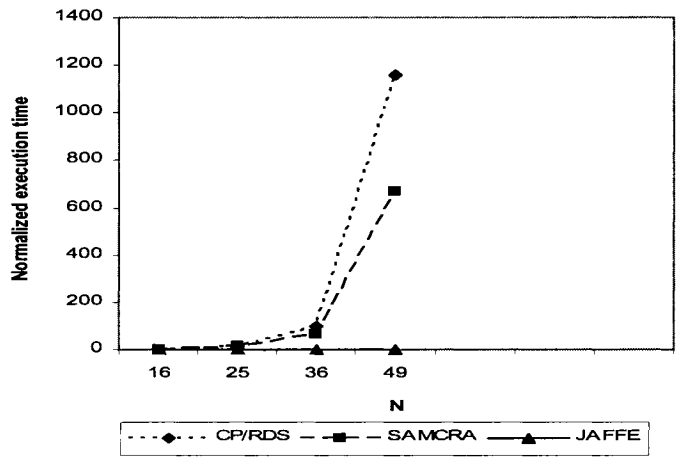


Figure 4.11: Normalized execution time for the class of Lattice graphs as a function of number of nodes for loose constraints. A negative correlated link structure is used. ($m = 2$ and ($\varepsilon_j = [0, 0.5w_j(P^*)]$))

a lattice graph was repeated with negatively correlated link weights. The correlation link structure of metrics has a significant impact on the running times of both the SAMCRA and CP/RDS algorithm. The fact is illustrated in Figures 4.10 and 4.11 where running times are shown to be much higher under negatively correlated link structures than positively link correlated graphs (Figures 4.8 and 4.9).

4.4 Execution times as a function of number of metrics

The execution times of the CP/RDS with respect to the number of constraints m (m = 2, 4, 6, and 8) under independent link weights each in the range $[0, 100]$ was simulated. In this simulation 2000 Waxman graphs each containing 200 nodes were generated.

Remark 16 *The Jaffe algorithm is omitted from the analysis due to the fact that it is designed to work with only two metrics.*

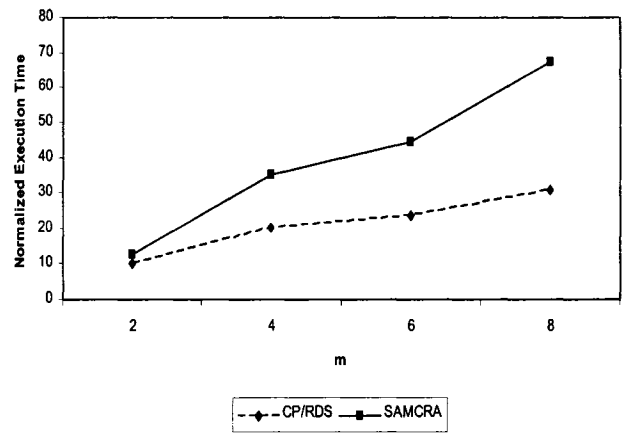


Figure 4.12: Normalized execution time for the class of Waxman graphs as a function of number of additive metrics for strict constraints($N = 200$, and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)). 95% confidence intervals for each mean normalized time is given in Appendix D.

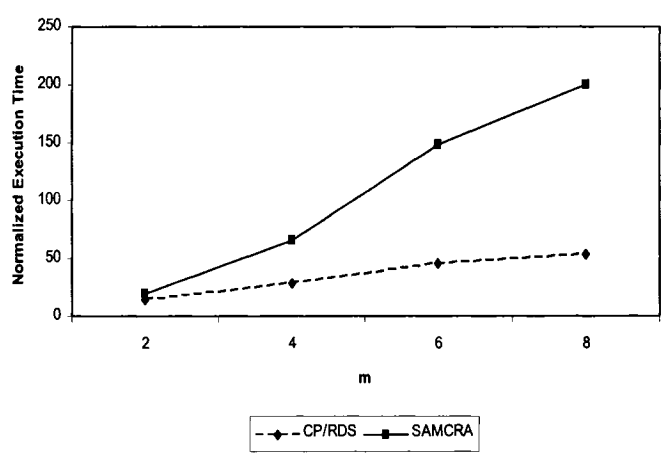


Figure 4.13: Normalized execution time for the class of Waxman graphs as a function of number of additive metrics for loose constraints($N = 200$, and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)). 95% confidence intervals for each mean normalized time is given in Appendix D.

The results of simulations for the class of Waxman and Barabasi graphs are shown in Figures 4.12/4.13 and 4.14/4.15 respectively. The class of Waxman graphs under strict user constraints showed a proportionate increase in the execution times of both the SAMCRA and CP/RDS algorithms for values of m ranging from 2 to 8 as illustrated in Figure 4.12. The increase in execution time as m increases from 2 to 8 can be accounted for by the fact that both algorithms require additional time and memory to process additional metrics in the dense edge structures present in Waxman graphs. The lower running execution times for the CP/RDS can be accounted for by the fact that the algorithm applies a heuristic to restrict the number of feasible paths to 1. In other words, the average number of paths being passed to the RDS algorithm from the CP algorithm decreases to 1 as the number of metrics used in the algorithm increases - this is illustrated in Figure 4.18. As a consequence, the computational time associated with finding additional paths is avoided. Under loose constraints there was also a proportionate increase in the execution times for both the SAMCRA and CP/RDS algorithms for values of m ranging from 2 to 8 as illustrated in Figure 4.13. However, for all values of m, under loose constraints the execution times associated with the SAMCRA algorithm is significantly higher than the CP/RDS algorithm.

The simulation results for the class of Barabasi graphs under strict user constraints showed an initial increase in execution times as m is increased from 2 to 4. However, for $m > 4$ there is proportionate decrease in the execution times of both the SAMCRA and CP/RDS algorithms as illustrated in Figure 4.14. This result is not expected since the increase in the number of metrics is typically associated with an increase in computational time of algorithms. The explanation for this anomalous behavior lies in the way the Barabasi graph is constructed with edges preferentially attached to nodes with high degrees. That is, in a Barabasi graph there exists a few nodes with high out-degrees and many nodes with low out-degrees. As a consequence, paths are generally short and have a common set of core nodes. As the number of metrics

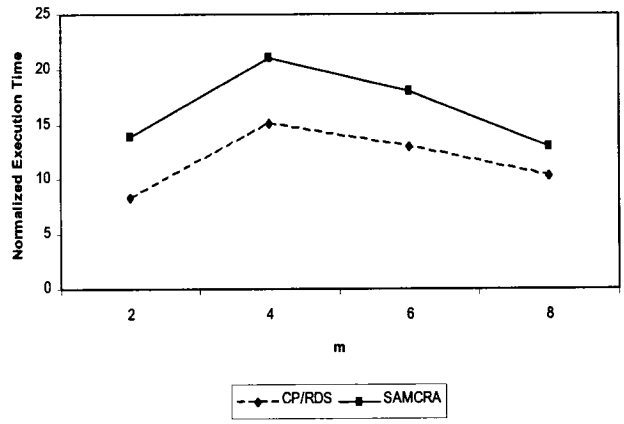


Figure 4.14: Normalized execution time for the class of Barabasi graphs as a function of the number of additive metrics for strict constraints ($N = 200$, and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)). 95% confidence intervals for each mean normalized time is given in Appendix D.

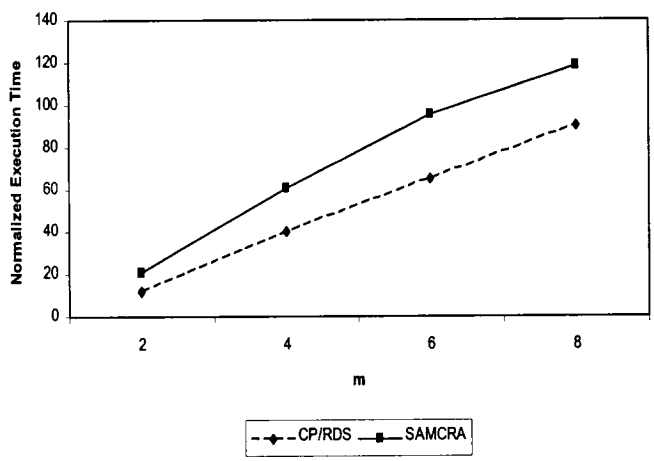


Figure 4.15: Normalized execution time for the class of Barabasi graphs as a function of the number of additive metrics for loose constraints ($N = 200$, and ($\varepsilon_j = [0, 0.5w_j(P^*)]$)). 95% confidence intervals for each mean normalized time is given in Appendix D.

increases, the possibility of eliminating sub-paths from source to destination that violate any one of the user constraints increases. Eliminating sub-paths means that the search space is reduced which in turn means that execution times of the SAMCRA and CP/RDS will decrease. This is not true for Waxman graphs because all nodes have the same average out-degree and no preferential policy is used to create edges, resulting in Waxman graphs having dense search spaces.

Remark 17 *The decrease in execution time is only evident if the increase in computational time associated with additional time and memory to process additional metrics is less than the gain in execution time associated with a reduced search space. This explains the increase in execution time for* $2 < m \leq 4$.

The simulation results for the class of Barabasi graphs under loose user constraints conformed to the theoretical expectations of an increase in execution times for both the SAMCRA and CP/RDS algorithm as illustrated in Figures 4.14 and 4.15. Of course, CP/RDS have lower running times than the SAMCRA for the same reason that was advanced for the case when Waxman graphs were used. However, the reason why there is no decrease in execution times with respect to an increase in the number of metrics, as in the case with strict user constraints, is that only a small number of sub-paths under loose constraints violate the user constraints thus making the search space of the algorithm large.

4.5 Number of paths CP produces

To examine the number of constraint paths (k) that the CP algorithm produces, the simulations results from Section 4.3.1 that include data on the number of constraint paths produced by the CP algorithm are used. Figures 4.16 and 4.17 show the probability of the CP algorithm returning a certain number of constraint paths for Waxman graphs with strict and loose constraints respectively.

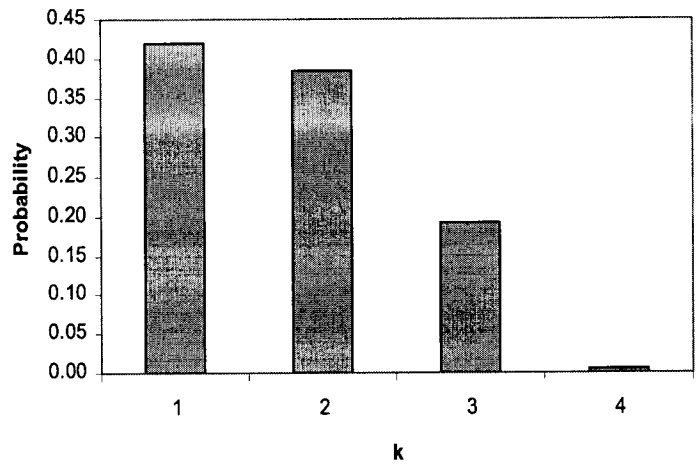


Figure 4.16: Probability versus the number of paths returned by the CP algorithm for class of Waxman graphs ($m = 2$, strict constraints, graph size 100 to 200 nodes)

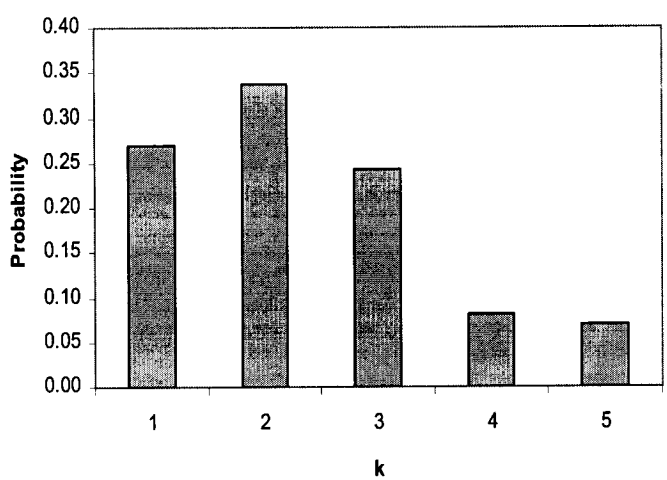


Figure 4.17: Probability versus the number of paths returned by the CP algorithm for class of Waxman graphs ($m = 2$, loose constraints, graph size 100 to 200 nodes)

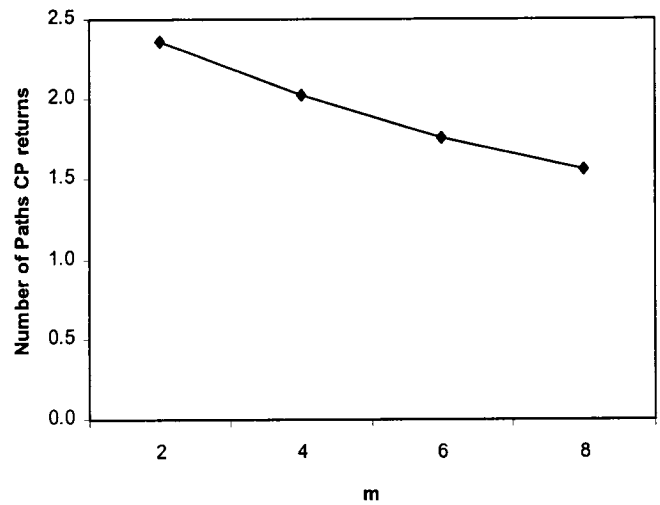


Figure 4.18: Number of paths the CP algorithm returns as a function of the number of metrics used in the algorithm for the class of Waxman graphs. Strict user constraints are used with $N = 100$, and $\varepsilon_j = [0, 0.5w_j(P^*)]$

The number of paths returned by the CP algorithm under strict constraints is between 1 and 4. The probability that one path is returned is 0.40, and therefore the probability of at least two paths being returned is 0.60. When only one path is returned this could be an indication that the CP/RDS algorithm reduced the search space due to a large queue size. As a consequence, the RDS algorithm does not need to perform any preference ranking which could result in low pareto success rates[2]. Under loose constraints the number of paths returned by the CP algorithm increases. The probability of at least two paths being returned is greater than 0.73 and therefore the RDS algorithm under loose constraints plays a greater role in deciding which constraint path is selected. Additionally, an important point to note from these results is that the choice of $\lambda = 7$ is justified since the probability of 7 paths being returned by the CP algorithm is zero.

[2]See Section 4.6 for explanation of pareto success rates.

Figure 4.18 shows that the number of paths returned by the CP algorithm under strict constraints decreases as the number of additive metrics used in the algorithm increases where each point on the curve represents the average number of paths returned by 2000 Waxman graphs. As the number of metrics increases, the possibility of eliminating sub-paths from source to destination that violate any one of the user constraints increases. This means that as m increases the search space decreases. A decrease in search space means that fewer constraint paths would be found. Hence as m increases the number of paths the CP returns decreases.

4.6 RDS Optimization

To investigate the success ratio that the CP/RDS algorithm has at returning a path that is a member of the global pareto optimal set (GPO) the term pareto success rate is defined as follows:

Definition 13 *Pareto success rate*

The pareto success rate of QoS algorithm is the number of times the algorithm produces a path that is a member of GPO set divided by the total number of times the algorithm is executed.

Using the results from simulations in Section 4.3.1, the pareto success rate for graph sizes 50, 100, 200, 300 and 400 for strict and loose constraints was calculated. Figure 4.19 shows the results of this simulation.

The CP/RDS algorithm is not guaranteed to find an optimal path. However, if a user flow requests optimization under strict user constraints, the CP/RDS algorithm has a 94 to 98 percent chance of finding an optimal path as illustrated in Figure 4.19. However, under loose constraints the CP/RDS algorithm has a 60 to 80 percent chance of finding an optimal path. These results confirm that the CP/RDS algorithm is heuristic with respect to solving the MCOP problem. Since strict constraints are

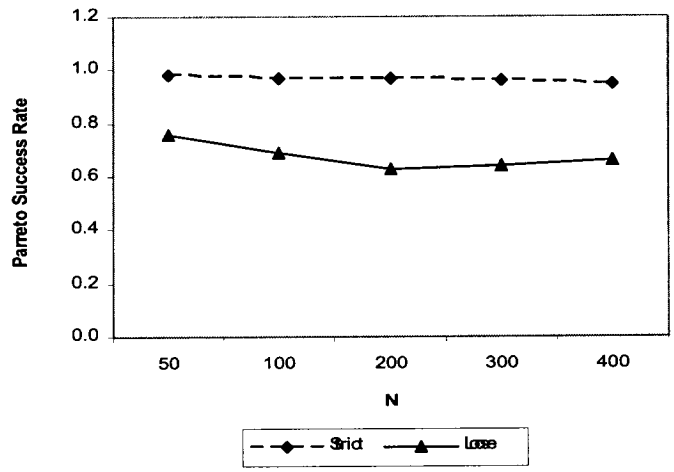


Figure 4.19: Pareto success rate as a function of the size of the network (N) under strict and loose constraints.

more important for real time applications the CP/RDS can still be considered a good algorithm for finding an optimal path for QoS flows.

4.7 Summary

Tables 4.7, 4.8, 4.9 and 4.10 summarize the simulations results presented in this chapter.

Table 4.7: Summary results for success rates of algorithms under strict and loose constraints

Algorithm	Success rate under strict constraints	Success rate under loose constraints
CP/RDS	100%	100%
SAMCRA	100%	100%
JAFFE	60%-80%	100%

Table 4.8: Algorithms under strict and loose constraints for given graph topologies

Graph Type	Strict user constraints	Loose user constraints
Waxman	The CP/RDS and SAMCRA algorithm have polynomial time complexity with the CP/RDS algorithm having lower running times.	The CP/RDS and SAMCRA algorithm have high running times. However, the CP/RDS algorithm has significantly lower running times.
Barabasi	Same as for Waxman graphs. However, execution times are generally higher than for Waxman graphs.	Same as for Waxman graphs. However, execution times are generally higher than for Waxman graphs.
Lattice	The SAMCRA has much lower execution times. However, both the SAMCRA and the CP/RDS has expotentail time complexity with respect to network size.	Same as for strict constraints. However, execution times are generally higher than for Waxman graphs.

Table 4.9: Running times of the CP/RDS and SAMCRA under strict and loose constraints for the Waxman and Barabasi topologies as the number of additive metrics used in the algorithms increases.

Graph Type	Strict user constraints	Loose user constraints
Waxman	Both the CP/RDS and SAMCRA algorithms have increase running times with respect to an increase in the number of additive metrics used in the algorithms.	Same as for strict user constraints with higher running times.
Barabasi	Both the CP/RDS and SAMCRA algorithms have decrease running times with respect to an increase in the number of additive metrics for $m > 2$	Both the CP/RDS and SAMCRA have increase running times with to an increase in additive metrics.

Table 4.10: Pareto success rates for strict and loose user constraints when a Waxman graph topology is used.

Algorithm	Type of constraint	Pareto success rate
SAMCRA	Strict	100%
	Loose	100%
CP/RDS	Strict	94%-98%
	Loose	60%-100%

4.8 Conclusion

The CP/RDS algorithm is guaranteed to find a path that satisfies the QoS goals of the user if such a path exists in the network. However, the CP/RDS does not guarantee that the chosen path would be optimized in terms of the user goals. The CP/RDS algorithm has faster execution times than the SAMCRA algorithm under strict and loose constraints for Waxman and Barabasi graph topologies, while for the lattice topology both the algorithms have exponential running times with the SAMCRA algorithm having much better execution times then the CP/RDS algorithm.

The simulations results in this chapter suggest that exact algorithms, although very accurate, are unrealistic for use in commercial communication networks due to the unpredictable high running times associated with them. In addition, there are many factors that contribute to an algorithm's NP-complete behavior including the link weight correlation structure of the network, requested user constraints and the topology of the network. Therefore, when exact algorithms are used to find constraint paths in communication networks there always exists the possibility that long running times will occur.

The number of additive metrics used by an exact algorithm can severely affect the algorithm's running time depending on the network topology and type of user constraints requested. Under strict constraints an increase in the number of metrics leads to gradual increases in running times of exact algorithms while under loose

constraints the increase is more significant. The one exception to this increase in running times with respect to an increase in the number of additive metrics is when strict constraints are used in a Barabasi graph topology. In this case, decreases in an exact algorithm's running time are evident after a given number of additive metrics is added to the algorithm.

The fact that the number of paths returned by the CP algorithm is one 40% of the time under strict user constraints, defeats the rationale for introducing a powerful multiple objective algorithm like the RDS, which requires at least two paths to be passed to it by the CP algorithm to justify its use. Hence although the rationale for separating the path finding mechanism from the optimization mechanism has obvious advantages, the exact CP algorithm is not adequate enough to be used as the path finding algorithm in all circumstances.

Chapter 5

Heuristic CP/RDS Algorithm

It was stated in Chapter 3 that exact algorithms offer a more realistic approach to solving the MCP/MCOP problem. Although this is obviously theoretically true, results of simulations presented in the Chapter 4 suggest that the price of exactness is high algorithm execution times. Especially in the context of using QoS algorithms to implement aspects of TE, where it may be necessary to route thousands of traffic flows each minute.

To address this speed issue a heuristic version of the CP/RDS algorithm that can be used in TE applications has been developed. This approach, and simulations to support its performance are presented in this chapter.

5.1 Heuristic Approach

One of the biggest challenges with using QoS algorithms in communication networks is finding a mechanism for user applications to communicate metric constraints to internetworking devices. Currently, the Reservation Resource Protocol (RSVP) is the most widely used signaling protocol that applications use to advertise application QoS constraints. However, RSVP only supports the bandwidth, delay and jitter constraints [13]. That is, constraint values for only 2 additive and 1 concave metric are supported by the RSVP protocol. Furthermore, the number of network metrics that can be communicated from a user to a network is limited to bandwidth, delay, jitter and packet loss rate. That is, only 3 additive and 1 concave metric can be communicated from the application to the network. The CP portion of the CP/RDS algorithm is only concerned with network metrics as stated in Remark 2. A heuristic

version of the CP algorithm that has high success rates and guaranteed fast running times for 3 or less additive metrics is presented in this section. In addition, if the user or network carrier requires a network path that is optimized for additional policy based metrics, the RDS algorithm can accommodate the processing of these metrics.

An algorithm that can be viewed as an improvement to the Iwata algorithm [35] is proposed. The Iwata algorithm attempts to find a path optimized for one of the metrics and then evaluates whether this can guarantee other user specified QoS requirements. If it can, the algorithm is halted and this path is selected. On the other hand, if the path optimized for the first metric cannot meet the QoS requirements of all the other user constraints, a path optimized for the second metric is selected and the procedure repeated until an appropriate path is found or until all attempts are exhausted. As seen in Figure 2.3 and Figure 2.4 this approach under strict constraints can lead to relatively good running times but poor performance in terms of success rates.

The proposed heuristic finds $\lambda \times m$ paths between source and destination nodes, where λ is a small positive integer and m is the number of QoS metrics under consideration. The algorithm is called the Constraint Path Heuristic (CP-H) algorithm and it is outlined below.

Algorithm 5 $CP - H(G, s, t, C, L, \lambda, m)\{$

1. $j \leftarrow 1$

2. while $(j \leq m)$ $\{$

3. $\quad i \leftarrow 1$

4. $\quad$ *while* $(i \leq \lambda)$ $\{$

5. $\quad\quad P_i \leftarrow Dijkstra(G, s, t, j)$

6. $PrunePath(P_i, j)$

7. $C \leftarrow C \cup P_i$

8. $i \leftarrow i + 1$

9. }

10. $j \leftarrow j + 1$

11. }

12. }

Algorithm 6 *PrunePath(P, j){*

For each link l in P{

$\tau = \max_j(P)$

$l_j \leftarrow \tau + l_j$ *{sets the link weight for the jth metric, where τ is a largest value for the jth metric on path P.}*

}

The algorithm accepts a graph G, source node s, destination node t, an empty set C for storage of $\lambda \times m$ paths between s and t, the set of user constraints L, the value of λ, and the number of additive metrics m. The first while loop, lines 2-11 is concerned with ensuring that each metric $j \in [1, m]$ is considered. The second while loop, lines 4-9 is concerned with ensuring that λ paths are added to set C for each metric. For a given metric, say m_1, the Dijkstra's algorithm (line 5) is first executed and produces a shortest path P_1 with respect to metric m_1. The pruning procedure (line 6) is then executed and attempts to prevent path P_1 from being selected the next time the Dijkstra's algorithm is executed for metric m_1 by incrementing the link values of path P_1 by τ, where τ is set to the largest link value for m_1 on path P_1. In other words, pruning attempts to ensure that distinct paths result from successive

runs of the Dijkstra's algorithm which in turn increases the chance that one of the λ paths would satisfy all QoS constraints. The paths are stored in set C and are passed to the RDS algorithm which decides which path in set C best satisfies a desired set of objectives.

The value of τ in the pruning procedure can impact the types of paths that are searched. For example, suppose τ is fixed to 20 and the metric delay is under considerarion. After the first Dijkstra's algorithm is executed and a shortest path say P_1 is computed, all the link values for delay on path P_1 would be increased by 20. Because paths can share links, this can prevent the next run of the Dijkstra's algorithm from finding a feasible path, say P_2, since this path may have a common link or links with P_1 which causes it to lose its shortest path status due to an increase in the delay value of one or more of it links . On the other hand, setting τ to a very small value can lead to successive runs of the Dijkstra's algorithm producing the same path.

The value of τ this thesis uses is the maximum of the link values of path P_1 with respect to a given metric. It is difficult to predict a good value for τ since it is a function of the graph topology, correlation link structure and user requested constraints, which are all random variables. However, simulation results show that a high number of distinct feasible paths in set C is obtained if τ is set to the maximum of link values of path P_1 with respect to a given metric.

Remark 18 *The total worst case time complexity of the CP-H/RDS algorithm is* $O(\lambda(mN\log(N) + mE) + m(\lambda + 1))$.

The space complexity of the CP-H/RDS is $O(\lambda N)$

5.2 Evaluation of the CP-H/RDS Approach

To evaluate the CP-H/RDS algorithm Java is used to implement the Binary Heap data structure and the CP-H/RDS algorithm. The simulations of in Sections 4.2,

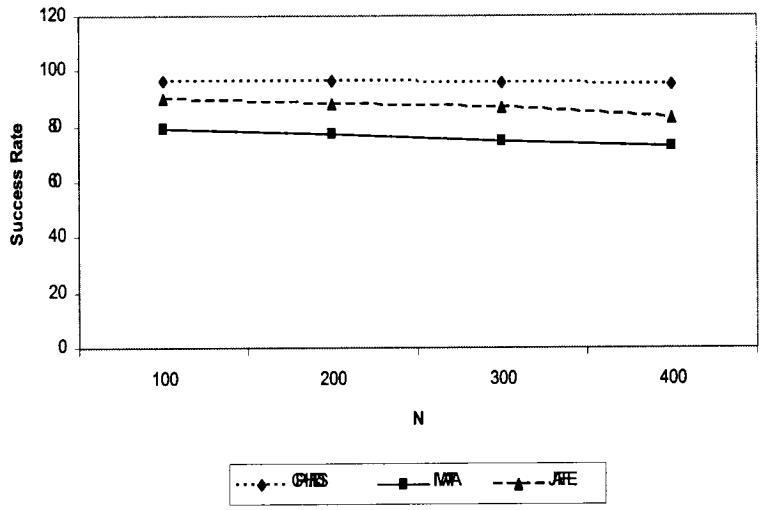


Figure 5.1: Success rates for the class of Waxman graphs as a function of number of nodes for strict constraints ($m = 2$ and $(\varepsilon_j = [0, 0.5w_j(P^*)])$)

4.3.1, 4.4, and 4.6 were repeated for the Waxman graph class using CP-H/RDS. Figure 5.1 shows the success rate for the CP-H/RDS, Iwata and Jaffe algorithms as a function of the size of the network under strict user constraints. These results indicate that the CP-H/RDS algorithm performs consistently better than the Jaffe and Iwata algorithms in terms of success rates. The success rate of the CP-H/RDS ranges between 93% and 96% when strict constraints and 2 additive metrics are used with a Waxman graph topology.

Remark 19 *λ is set to 3 for all simulations.*

Figures 5.2 and 5.3 show the normalized execution times for the CP-H/RDS, CP/RDS, SAMCRA, Iwata and Jaffe algorithms as a function of the size of the network for strict and loose constraints respectively. For both strict and loose constraints the CP-H/RDS algorithm has significantly lower running times than the CP/RDS and SAMCRA algorithms. However, the CP-H/RDS algorithm has higher running

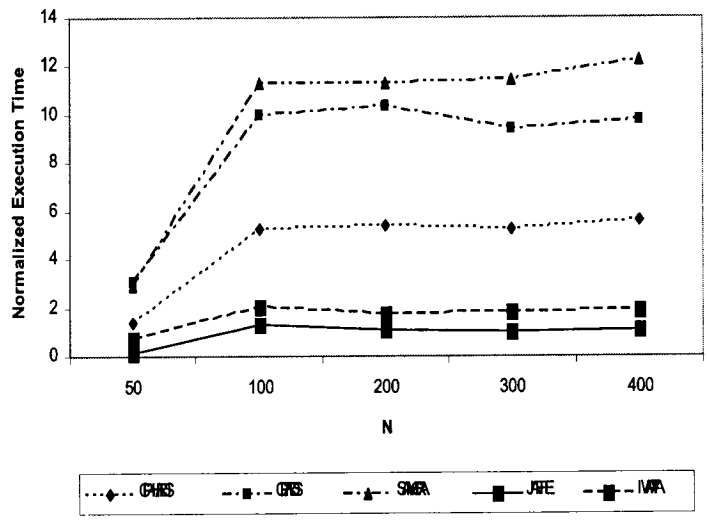


Figure 5.2: Normalized execution time for the class of Waxman graphs as a function of number of nodes for strict constraints ($m = 2$ and $(\varepsilon_j = [0, 0.5w_j(P^*)])$)

times than its heuristic counterparts. The average execution time of the CP-H/RDS algorithm is about 5 times the Dijkstra's algorithm.

Remark 20 *The 95% confidence intervals for strict and loose constraints do not overlap when $N \geq 100$ and therefore suggests that the execution times for the CP-H/RDS algorithm, CP/RDS, SAMCRA and Iwata algorithms are statistically distinguishable.*

Figures 5.4 and 5.5 show the normalized execution times for the CP-H/RDS, CP/RDS, Iwata and SAMCRA algorithms as a function of the number of additive metrics for strict and loose constraints respectively. For both constraints it is clear that CP-H/RDS execution times increase proportionally with the number of additive constraints. However, the CP-H/RDS algorithm has lower running times than the SAMCRA and CP/RDS algorithms as the number of metrics used by the algorithm increases. Additionally, under loose constraints it is clear that the SAMCRA

Table 5.1: 95% Confidence Intervals for mean execution times of Figure 5.2.

N	Algorithm	Mean time	95% Confidence intervals
50	CP-H/RDS	1.42	(1.18, 1.66)
	Iwata	0.72	(0.55, 0.90)
	CP/RDS	3.09	(2.75, 3.43)
	SAMCRA	2.93	(2.58, 3.27)
	Jaffe	0.18	(0.059, 0.215)
100	CP-H/RDS	5.28	(4.81, 5.75)
	Iwata	2.05	(1.74, 2.36)
	CP/RDS	10.02	(9.38, 10.65)
	SAMCRA	11.33	(10.98, 12.49)
	Jaffe	1.30	(1.042, 1.55)
200	CP-H/RDS	5.38	(5.15, 5.60)
	Iwata	1.77	(1.61, 1.94)
	CP/RDS	10.37	(9.85, 10.99)
	SAMCRA	11.32	(12.35, 13.91)
	Jaffe	1.12	(0.99, 1.25)
300	CP-H/RDS	5.26	(5.12, 5.40)
	Iwata	1.81	(1.71, 1.92)
	CP/RDS	9.38	(8.92, 9.83)
	SAMCRA	11.43	(10.75, 12.11)
	Jaffe	1.02	(0.93, 1.11)
400	CP-H/RDS	5.64	(5.45, 5.83)
	Iwata	1.92	(1.81, 2.03)
	CP/RDS	9.75	(9.19, 10.30)
	SAMCRA	12.24	(11.33, 13.14)
	Jaffe	1.124	(1.03, 1.23)

Table 5.2: 95% Confidence Intervals for mean execution times of Figure 5.2.

N	Algorithm	Mean time	95% Confidence intervals
50	CP-H/RDS	2.59	(2.25, 2.91)
	Iwata	0.90	(0.70, 1.10)
	CP/RDS	4.01	(3.62, 4.40)
	SAMCRA	4.19	(3.78, 4.59)
	Jaffe	0.47	(0.32, 0.61)
100	CP-H/RDS	5.46	(5.00, 5.92)
	Iwata	2.12	(1.81, 2.43)
	CP/RDS	11.73	(11.13, 12.34)
	SAMCRA	15.43	(14.63, 16.23)
	Jaffe	1.21	(0.96, 1.44)
200	CP-H/RDS	5.36	(5.13, 5.58)
	Iwata	1.91	(1.743, 2.08)
	CP/RDS	12.54	(12.09, 12.98)
	SAMCRA	18.32	(17.43, 19.21)
	Jaffe	1.08	(0.95, 1.21)
300	CP-H/RDS	5.62	(5.48, 5.77)
	Iwata	18.13	(17.15, 19.10)
	CP/RDS	12.57	(12.14, 12.97)
	SAMCRA	18.13	(17.15, 19.10)
	Jaffe	1.10	(0.96, 1.14)
400	CP-H/RDS	6.45	(6.31, 6.59)
	Iwata	2.24	(2.15, 2.33)
	CP/RDS	14.15	(13.68, 14.61)
	SAMCRA	20.89	(19.21, 22.58)
	Jaffe	1.22	(1.14, 1.30)

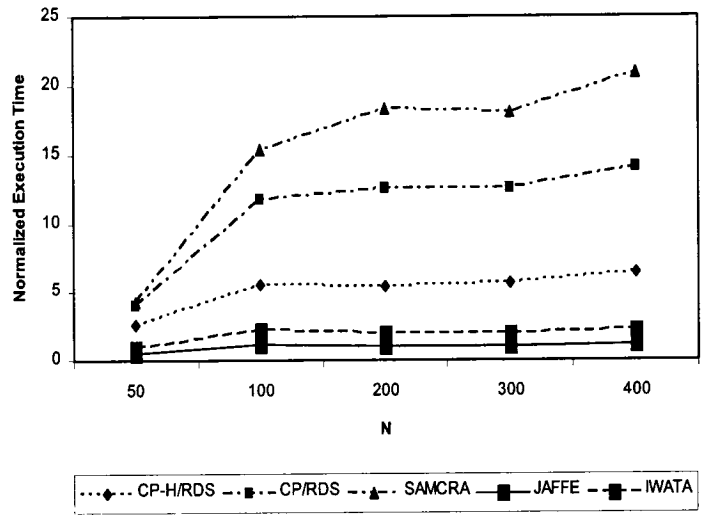


Figure 5.3: Normalized execution time for the class of Waxmax graphs as a function of number of nodes for loose constraints ($m = 2$ and $(\varepsilon_j = [0, 0.5w_j(P^*)])$).

algorithm performs significantly worse than the other algorithms as the number of additive metrics increases.

When the number of additive metrics used in the CP-H/RDS increases, the running times of the algorithm is expected to increase proportionally ($\lambda \times m$). This is because the number of times that the Dijkstra's algorithm needs to be executed is ($\lambda \times m$). At first glance this may seem very high. However, Figure 5.5 shows that for all values of m, under loose constraints, the execution times of the CP-H/RDS are better than both the SAMCRA and the CP/RDS algorithms. In addition, Figure 5.4 shows that under strict constraints and values of $m \leq 7$ the execution times of the CP-H/RDS are better than both the SAMCRA and the CP/RDS algorithms and when $5 \leq m \leq 7$ the execution times of the CP-H/RDS are comparable to the CP/RDS algorithm.

Remark 21 *Since the execution time of the CP-H/RDS algorithm is significantly increased as the number of metrics increases, it is recommended that no more than 3*

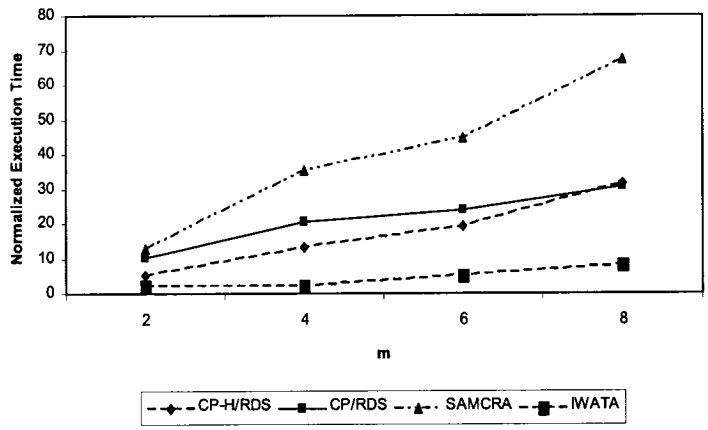


Figure 5.4: Normalized execution time for the class of Waxman graphs as a function of number of additive metrics for strict constraints($N = 200$, and $(\varepsilon_j = [0, 0.5w_j(P^*)])$). 95% confidence intervals for each mean normalized time is given in Appendix D.

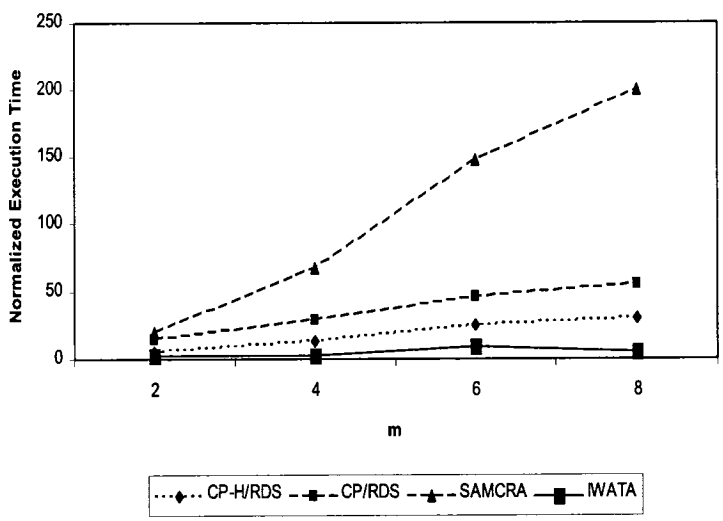


Figure 5.5: Normalized execution time for the class of Waxman graphs as a function of number of additive metrics for loose constraints($N = 200$, and $(\varepsilon_j = [0, 0.5w_j(P^*)])$). 95% confidence intervals for each mean normalized time is given in Appendix D.

additive metrics be used provided that $\lambda \leq 3$. The CP-H algorithm is concerned only with network metrics which include delay, jitter, packet loss and bandwidth. Hence the maximum number of metrics that will be used in the CP-H algorithm is 3 namely: delay, jitter and packet loss. Links that violate the bandwidth constraint are pruned from the network before the CP-H algorithm is executed. Any number of policy based metrics can still be processed by the CP-H/RDS algorithm via the RDS algorithm.

Figure 5.6 shows an interesting result. As the number of additive metrics increase the success rate of the CP-H/RDS remains constant. The Iwata algorithm success rate decreases with respect to an increase in additive metrics. The key reason for this can be seen in Figure 5.7. As the number of additive metrics increases the paths considered in the search process for both the CP-H/RDS and Iwata increase. However, in the case of the CP-H/RDS algorithm, the number of paths drastically increases as the number of additive metrics increases. A larger path search space means a better chance of finding a path satisfying all metrics. Hence the CP-H/RDS algorithm has higher success rates than the Iwata algorithm as the number of additive metrics increases. In the case of the Iwata algorithm, although the search space increases, only one path per metric is being examined as a candidate for satisfying all metrics which can lead to an overall reduction in success rate since that one path must satisfy an increased number of constraints. However, in the case of the CP-H/RDS algorithm λ paths are considered for each metric.

Figure 5.8 shows the pareto success rate as a function of the network size under loose user constraints for the CP-H/RDS and CP/RDS algorithms. The pareto success rate for the CP-H/RDS is surprisingly higher than the CP/RDS algorithm. This result is not expected since the CP algorithm places considerable emphasis on finding non-dominant paths. However, a random search for the most optimal paths seems to be more successful at finding paths that are pareto optimal. One possible explanation for the high pareto success rates of the CP-H/RDS is due to the fact that

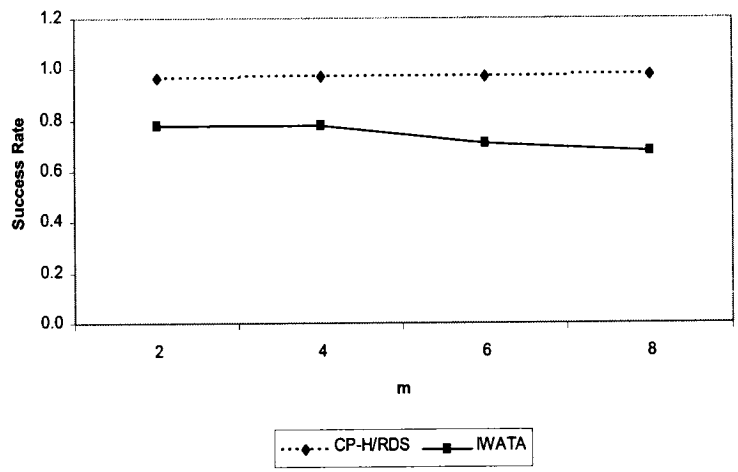


Figure 5.6: Success rate as a function of the number of additive metrics under strict user constraints for the Iwata and CP-H/RDS algorithms.

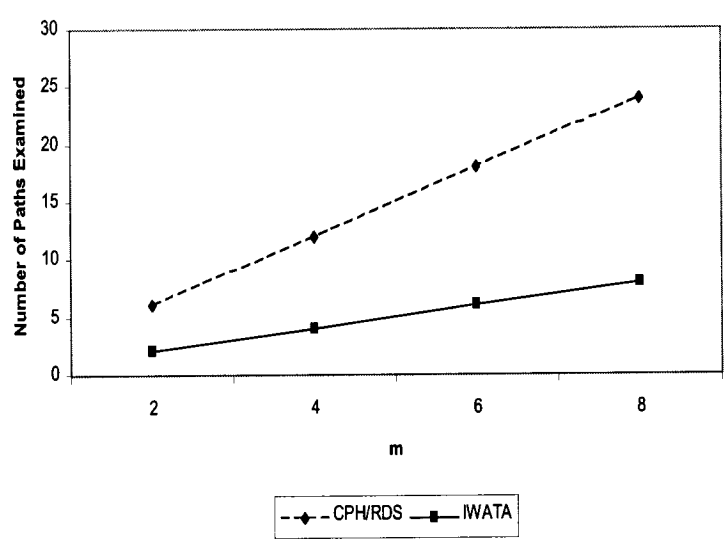


Figure 5.7: Number of paths involved in the routing decision for the class of Waxman graphs as a function of number of additive metrics for strict constraints ($N = 200$, and ($\varepsilon_j = [0, 0.5w_j(P^*)]$))

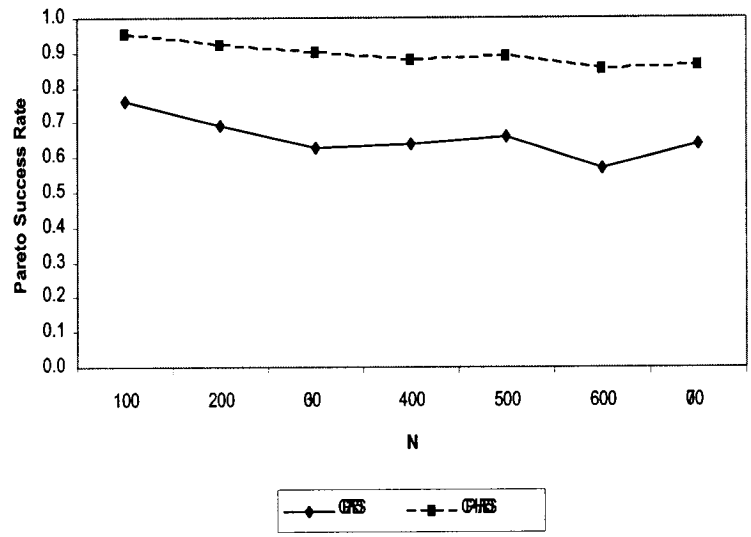


Figure 5.8: Pareto success rate versus the size of the network (N) under strict constraints

the Dijkstra algorithm finds the shortest path with respect to a given metric. This means a pareto optimal path would always be found provided that a feasible path is among the paths found by the Dijkstra's algorithm. Consequently, the pareto success rate of the CP-H/RDS should be equivalent to the success rate of the algorithm.

Figure 5.9 shows the probability of the CP-H algorithm returning a number of distinct feasible paths for Waxman graphs under strict user constraints. The probability that at least two distinct feasible paths are returned by the CP-H algorithm is 0.96. In addition, the probability that at least five distinct feasible paths are returned by the CP-H algorithm is 0.60. As a consequence, the RDS algorithm when used with the CP-H algorithm in place of the CP algorithm better serves the role of ranking feasible paths based on optimization goals since more feasible paths are present.

The probability of returning 6 distinct paths in a Waxman topology with $\lambda = 3$ and $m = 2$ is relatively high. This demonstrates that the pruning algorithm is doing

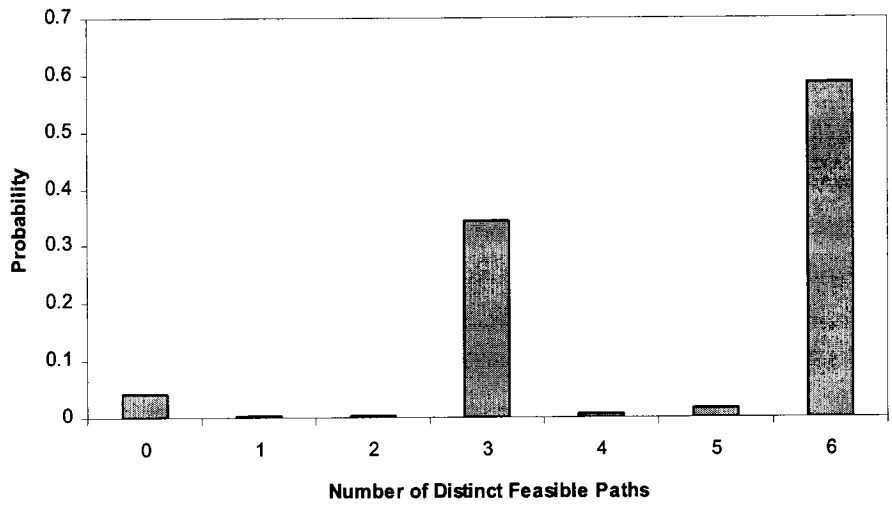


Figure 5.9: Probability versus the number of distinct feasible paths returned by the CP-H algorithm for class of Waxman graphs ($m = 2$, strict constraints, graph size 100 to 400 nodes)

its job. The choice of incrementing the links of the shortest path with the maximum of the link value for a given metric proves to be effective since 6 distinct paths are produced by the CP-H algorithm 60% of the times. Since the links of the Waxman graphs are randomly generated, there exists cases where the strategy used by the pruning algorithm will return less that 6 distinct paths. Figure 5.9 shows that the next highest amount of distinct paths returned is 3, which has a probabilityof 0.35. This is because $m = 2$ means that there will be at least two 2 distinct paths provided that the shortest path with respect to each metric is different. Since the pruning algorithm runs 3 times ($\lambda = 3$) for each metric, there is a good chance of finding at least one more distinct path. Consequently, the probabilty of returning 0, 1, 2, 4 or 5 paths is low.

5.3 Analysis of Call Blocking rate of the CP-H/RDS

The main goal of TE is to reduce call blocking rates. This objective can also be seen in terms of maximizing network resource utilization. In this section the performance of CP-H/RDS in the context of call blocking rate, is compared with the Minimum Hop Algorithm (MHA), the Least Interference Optimization routing Algorithm (LIOA) and the SAMCRA algorithm. Note that the popularly used Minimum Interference Routing Algorithm (MIRA) and its improvement L-MIRA [5] are not considered since results in [49] suggests that the LIOA has significantly reduced blocking rates and resource utilization over these algorithms. Bandwidth (concave metric) is the only user constraint that MHA and LIOA employed. The CP-H and SAMCRA algorithms primarily use additive metrics after pruning links not meeting the requested bandwidth. Therefore, for comparison purposes only delay is used as a metric for the CP-H algorithm while delay, cost, bandwidth and hop count are used as metrics for the RDS algorithm. The CP-H/RDS is implemented to address user and network carrier goals (described in Section 6.3) with the elimination of jitter from the model. Note that in the case of the SAMCRA algorithm, delay and cost (same as LIOA) metrics are used.

To study the call blocking rate of the CP-H/RDS algorithm, simulations done in [5] were reproduced with the main goal of comparing the CP-H/RDS existing TE QoS algorithms. Figure 5.10 shows the test network employed which consists of 15 nodes and 28 links. Links are bi-directional, representing two links with the same capacity in opposite directions. Lighter links have a capacity of 1200 bandwidth units, whilst the darker ones have 4800 bandwidth units, representing OC-12 and OC-48 rates respectively. In addition, each link has a delay value uniformly generated in the interval [0.02, 0.04]. Bandwidth requests are generated using a uniform distribution in the interval [1, 4] and a delay request is derived by first generating a value in the range [0.02, 0.04] and then multiplying this value by a hop count value generated

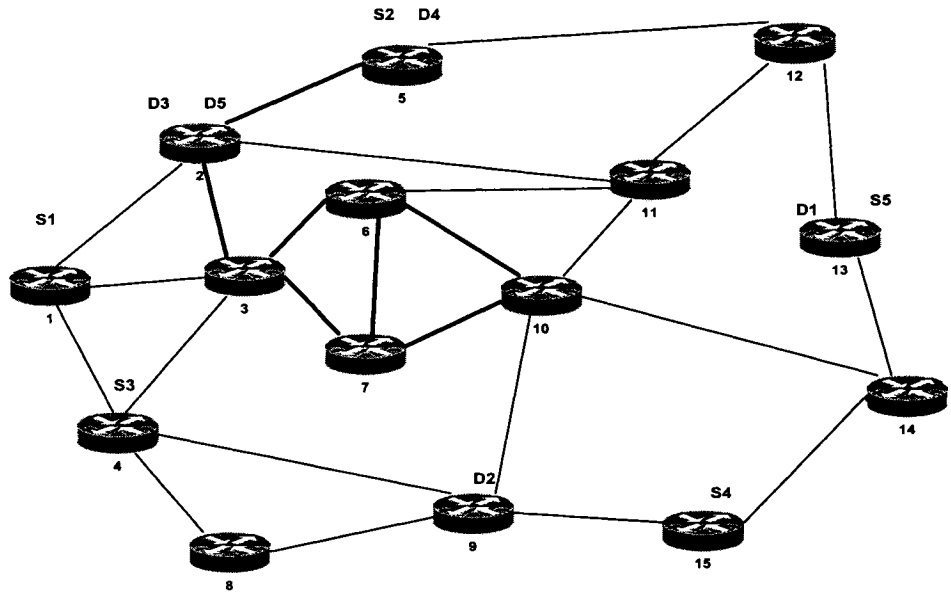


Figure 5.10: Topology used in the simulation taken from [5].

using a uniform distribution in the interval [6, 8].

Remark 22 *The main goal of the simulations is to study the impact that CP-H/RDS has on call blocking rate. Hence to compare CP-H/RDS fairly with other algorithms, the delay request value is set in such away as to eliminate blocks due to paths not satisfying the delay request.*

Three simulations A, B and C were performed corresponding to permanent, short and long lived flows respectively. Parameter values associated with each simulation are given in Table 5.3. The mean request rate is μ_1 and the mean service rate is μ_2. The requests are assumed to follow a possion distribution and the departures an exponential distribution. T denotes the number of simulation trials and N the number of requests per trial.

The simulation results for permanent flows (no link failures) are shown in Figure 5.11. The number of requests rejected is plotted as a function of the number of

Table 5.3: Simulation Parameters

	Simulations		
Parameter	A	B_1	B_2
α (LIOA and CP/RDS)	0.5	0.5	0.5
Connection Type	Permanent	Long	Short
μ_1	3	3	3
μ_2	∞	5000	50
T	20	20	20
N	8000	8000	8000

requests arriving at the network. The SAMCRA algorithm rejects highest number of requests followed by the MHA and then the CP-H/RDS. The SAMCRA algorithm chooses an optimal path based on a length function (see Section 2.3.2.2) that involves both delay and cost.. This strategy proves very expensive in terms of wastage of of the bandwidth resource. On the other hand, the MHA algorithm always chooses the shortest paths which causes links of this path to rapidly saturate. SAMCRA started rejecting requests from around 2800 requests, MHA around 3200 requests, CP/RDS around 3300 requests and LIOA started around 3600 requests. However, the rate of rejected requests for SAMCRA is higher than the MHA, LIOA and CP-H/RDS algorithms.

The simulation results for short lived and long lived flows (no link failures) are shown in Figure 5.12 and Figure 5.13 respectively. Both graphs indicate a significant improvement in the number of requests allowed into the network. This is expected since the bandwidth resource is constantly being consumed and returned to the network. The CP-H/RDS performs very well in terms of maintaining a low call blocking rate considering that its path selection process considers more than one user constraint as well as network resources. The SAMCRA algorithm like the CP-H/RDS algorithm finds optimal paths based on delay and cost. However, the important advantage that the CP-H/RDS has over the SAMCRA algorithm is that it is able find optimal paths

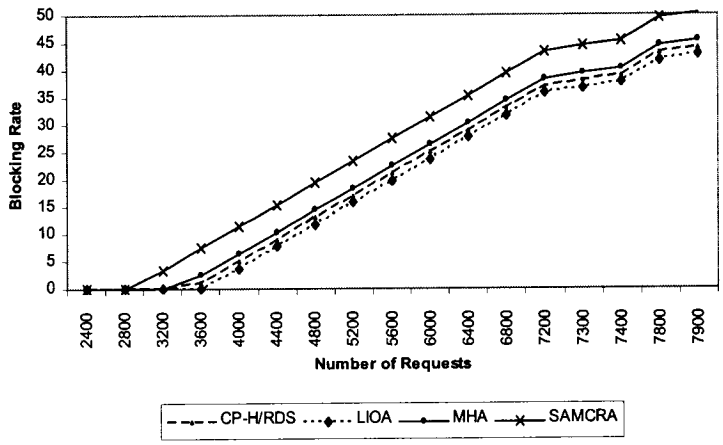


Figure 5.11: Number of requests in the case of permanent flows

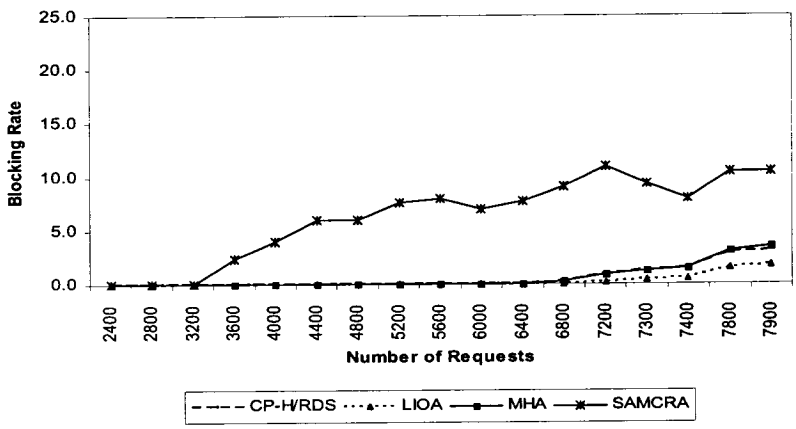


Figure 5.12: Number of requests in the case of short lived flows

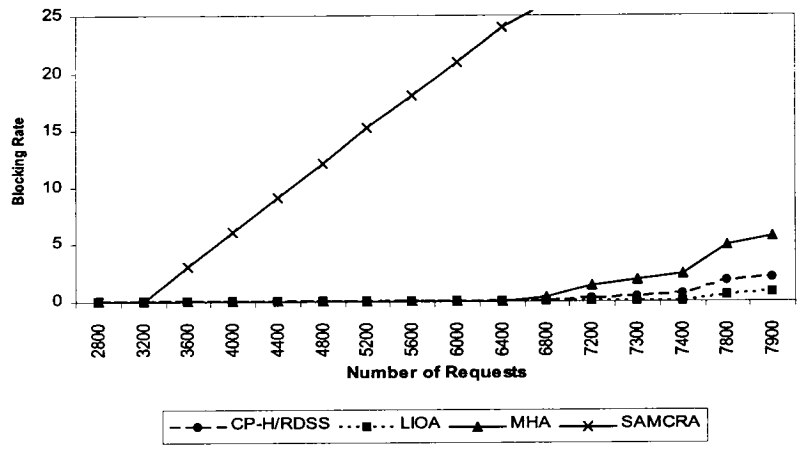


Figure 5.13: Number of requests in the case of long lived flows

is able find optimal paths that minimize the bandwidth resource as well the number of hops between source and destination nodes.

5.4 Conclusion

The CP-H/RDS algorithm is fast relative to exact algorithms and has higher success rates than the heuristic algorithms discussed in this chapter. The results in this chapter suggest that there is a trade-off between execution time of algorithms and their ability to find feasible paths. However, from the set of algorithms considered the CP-H/RDS algorithm offers the best trade-off between execution time and success rates. This makes it a good candidate for use in online routing where there are high demands for fast processing of multiple network and policy criteria.

The RDS algorithm depends on distinct feasible paths being passed from the constraint path algorithm in order to find the most suitable paths that match the optimization goals of both the user application and policy objectives of the network carrier. The CP-H algorithm has a very high probability of returning three or more distinct paths and therefore is very suitable for use with the RDS algorithm.

The call blocking rate of the CP-H/RDS algorithm is comparable to TE algorithms like MHA and LIOA. This means that the network carrier can achieve high resource usage as a result of using the CP-H/RDS algorithm.

Chapter 6

Framework for using CP-H/RDS in TE

Online Traffic Engineering provides a means of controlling traffic through a network, offering services tailored for traffic flow requirements while ensuring economical use of network resources. Keeping this in mind, the objective of QoS routing is for the routing algorithm to find paths that satisfy traffic flow requirements. Therefore, at first glance TE seems to be a good place to use the CP-H/RDS algorithm. However, a closer look may reveal that TE and QoS routing have contradicting objectives. On one hand, TE is concerned with optimizing network resources, particularly bandwidth, and on the other hand, QoS routing is concerned with satisfying traffic flow requirements. Hence, it is not straightforward to use CP-H/RDS in a TE environment. This chapter provides a rationale for using an intelligent path finding algorithm in TE. A framework that examines the potential use of the CP-H/RDS algorithm in a TE environment is presented. The framework describes how the CP-H/RDS algorithm could work with existing protocols to achieve QoS routing. The major contradiction is that a provider need not over-provision if the CP-H/RDS algorithm is used.

6.1 Rationale for using path finding algorithm

A network carrier makes money by delivering traffic from one node to another for its clients. Hence, network carriers sell bandwidth as a commodity [7] where standardized bandwidth pipes such as DS3 circuits are used. However, in this setup there is little differentiation of services among vendors, and therefore the carriers that provide the lowest prices per unit bandwidth attract the most clients.

The flexibility of IP networks has encouraged the integration of diverse traffic flows

on carrier networks. For example, applications like web browsing, e-mail, telephony, voice and video distribution, data mining, and e-commerce transaction processing have different requirements for bandwidth, delay, jitter and security. The existence of an intelligent path selection based on multiple user constraints such as the CP-H/RDS will allow network carriers to offer differentiated services on a retail market [7]. Consequently, network carriers will have the flexibility of customizing a customer's traffic flow and pricing it accordingly.

The constraints that are typically supported by online TE algorithms include available bandwidth and number of hops [48]. The power of the RDS part of the CP-H/RDS algorithm is invaluable in a TE environment since it is designed to process multiple objective criteria. Since the RDS algorithm allows individual configuration of traffic flows, network carriers will be able to offer highly specialized services.

To date, network carriers have introduced the concept of differentiated services using the DiffServ model. Assigning applications to different classes of service and marking the traffic appropriately allows for scheduling, queuing, and drop behavior based on the application type. However, to receive strict scheduling guarantees, it is not enough to mark traffic appropriately. If the traffic follows a path with inadequate resources to meet performance characteristics such as jitter or delay requirements, the user QoS requirements cannot be met. Of course, this problem could be solved by over provisioning the network to avoid congestion altogether. However, this approach wastes network resources and cannot provide any guarantees when congestion is caused by link and/or node failures. Many models are proposed [78][79][80][81][82] that integrate TE, MPLS and DiffServ. In these models the concept of class type is used to control the maximum and minimum bandwidth guarantees for traffic classes. However, there is no maximum or minimum bandwidth requirement to be enforced at the level of an individual flow within the class type. Thus traffic engineering is performed on a per class-type level which limits the flexibility that a network carrier

has to provide services at the customer level.

6.2 Routing in the Management Plane

The network layer has changed little over the past seven years. This is true in spite of the fact that applications have dramatically changed from simple email and file transfer to complex multimedia applications such as telemedicine and interactive games. The main reason for the little change is that any changes would mean updating potentially every router on the Internet. However, there is an increasing demand for complex router services to meet the needs of interactive and multimedia applications. One possible solution is to move the routing functionality as far as routing decisions are concerned to the management plane [29]. In this way QoS objectives as well as improvements to technology can be made without the need for updating routers' software.

One of the key advantages for making routing decisions in the management plane is that the complexity introduced by QoS awareness remains outside the network layer. This makes it possible to use state-of-the-art hardware that have lots of memory and CPU power for QoS routing. In Chapter 5 the average time for the CP-H/RDS algorithm was shown to be 5 times that of Dijkstra's algorithm. Since the Dijkstra's algorithm is already computationally expensive this would significantly increase the time it takes for routing protocols to converge if routing decisions were made on the control plane [29] (layer 3).

6.3 Using CP-H/RDS in online TE

The CP portion of the algorithm CP/RDS is responsible for finding a subset of non-dominant paths between two nodes in a network that satisfies a set of user constraints. When additive metrics are involved finding such paths could lead to long running times, making CP/RDS unsuitable for use as an online TE algorithm. Since the

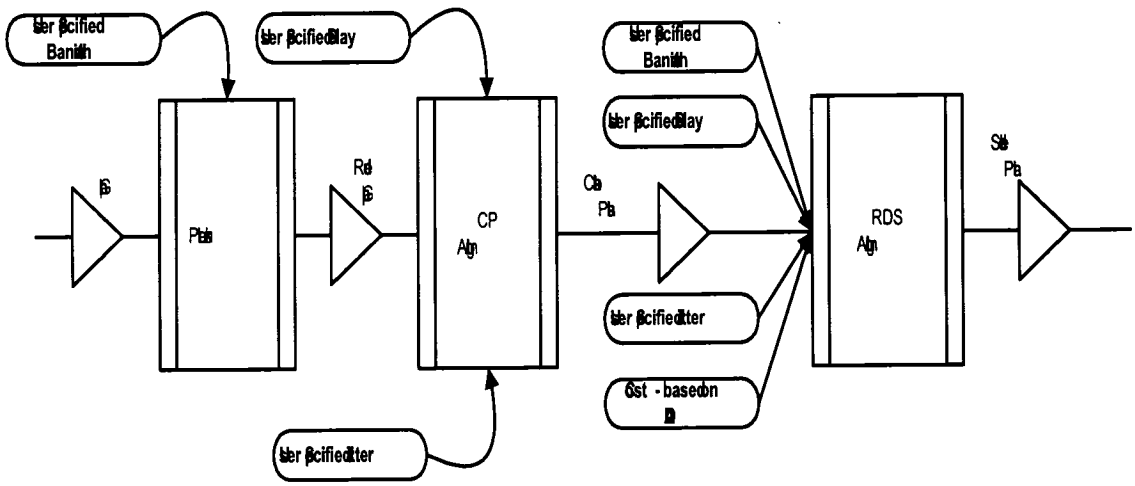


Figure 6.1: A pictorial view of how the CP-H/RDS algorithm works in a TE environment.

CP/RDS separates the path finding process from the optimization process this makes it relatively simple to modify the algorithm to accommodate online TE. In this section the CP-H/RDS algorithm 5 is adapted to make it suitable for use in an online traffic engineering environment.

The problem definition that the CP-H/RDS algorithm addresses is as follows. Consider, a network represented by a directed graph (N, L) where N is a set of nodes and L is a set of links. Let R_l denote the maximum reservable bandwidth of the link l and let $P_{i,e}$ denote the set of feasible paths connecting ingress-egress pair (i, e) and that each link is characterized by bandwidth, delay and jitter. Assume that there exists a global objective of maximizing the number of flows that the network carries.

Let each user request consist of a bandwidth demand $d^B_{i,e}$, a delay demand $d^D_{i,e}$, and a jitter demand $d^J_{i,e}$, and that future demands concerning requests are unknown. The cost function that is used is similar to the LIOA. Let $L_p = \sum_{l \in p} L_l(I_l, r_l)$ denote the cost of path p where $L_l(I_l, r_l)$ is the cost on each link l when carrying I_l flows and r_l is the total bandwidth reserved by all paths traversing link l. The routing problem

consists of finding a feasible path $p^* \in P_{i,e}$ on an ingress-egress pair (i,e) that meets the following multi-criteria objectives:

1. $d^B_{i,e} \leq \min_{l \in p^*}(R_l - r_l)$
2. $d^D_{i,e} \geq \sum_{p^* \in P_{i,e}} l_d$, where l_d is the delay for a given link on path p^*.
3. $d^J_{i,e} \geq \sum_{p^* \in P_{i,e}} l_j$, where l_j is the jitter for a given link on path p^*.
4. $L_{p^*} = \min_{p^* \in P_{i,e}} L_p$
5. Meeting the user and network carrier's optimization objectives for the traffic flow. For example, if the carrier's objective is to minimize bandwidth and the user objective is to minimize delay and jitter then the v^1 vector for such a flow would be (-1, 1, 1).

Objective 1 can be achieved simply by pruning the network for all links that do not satisfy $R_l - r_l < d^B_{i,e}$. Objectives 2 and 3 can be addressed by using the CP-H algorithm with delay and jitter as the two constraints. Objectives, 4 and 5 can be addressed by using the RDS algorithm with the optimization requirements for bandwidth, delay, jitter and cost as input. Figure 6.1 shows a representation of the entire process.

6.4 Framework for using CP-H/RDS in online TE

In order for Traffic Engineering to take advantage of multiple constraint-based routing algorithms like the CP/RDS the following must be in place: 1) A mechanism to distribute/acquire information about the network topology (including link characteristics); 2) A mechanism to capture the constraint information of user flows; 3) A mechanism to install explicit routes in the core network calculated by the CP-H/RDS;

[1]See page RDS algorithm

and 4) A mechanism to setup traffic flow priorities in routers and switches. These goals are achieved by adopting a centralized route server approach similar to the RATES server [83] where a centralized routing server is used to make routing decisions on behalf of ingress routers. An alternative would be a distributed approach that involves extensions to existing IGPs similar to the proposals in [18] which include an upgrade to the Link State Advertisement (LSA) OSPF packet to carry both cost and bandwidth. After network link attributes and topology information are flooded by the IGP and placed in the TED, the CP/RDS algorithm running at each node in the network will then use this information to calculate paths for flows. However, extensions can take a long time to become standards and this solution would not be flexible in the context of the rapid changes in Internet technologies on the management plane against stable IGPs like OSPF.

The following sections examines how the centralized routing server can be used to accomplish the five goals outlined above. The server is called the Routing Decision System Server (RDSS) and Figure 6.2 shows a summary of the tasks involved in the relationship between the ingress router and RDSS.

6.4.1 Maintenance of Topology Information

RDSS assumes that a link state protocol like OSPF is implemented in the core. There are two ways to get topology information about the core network. The first is to use a network management protocol like Simple Network Management Protocol (SNMP) to read the Management Information Base (MIB) of the ingress router. However, this approach is not fool proof since SNMP is unreliable (uses UDP) and the management information base (MIB) of routers is not standardized [83]. The second approach is to peer as a link state protocol with some other node in the network. An advantage of using link state peering is that, in the case of a link failure, the routing server will learn about the failure in the same time frame that the link state protocol takes to

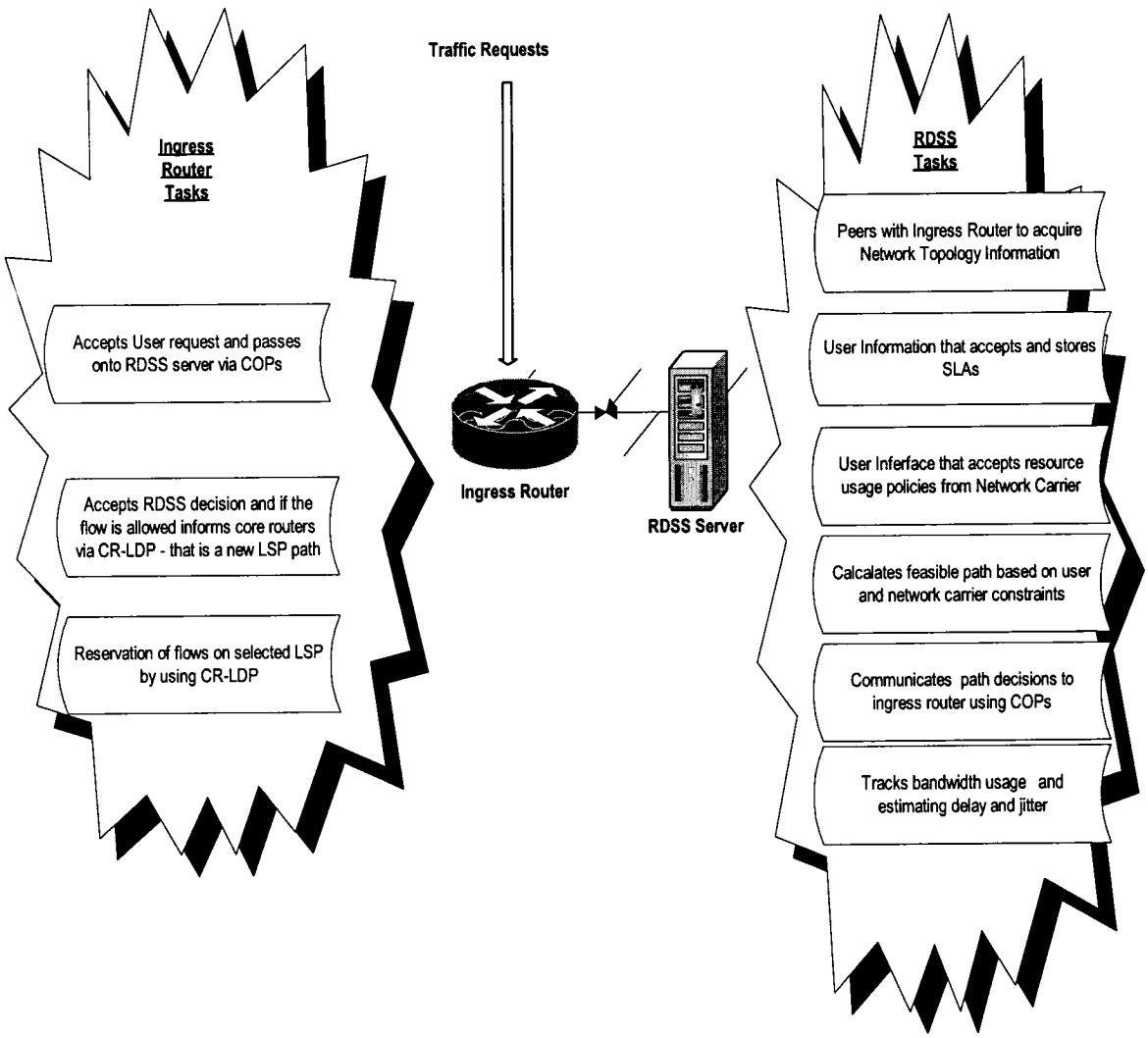


Figure 6.2: A summary of tasks performed by both RDSS and ingress router when flows request admission into the network

converge. This is a significant advantage since in the case of SNMP the only way for the route server to learn about the failure in a timely fashion is to use SNMP traps. The latter consume bandwidth, are difficult to install and manage, and are unreliable. Hence it is proposed that the RDSS use protocol peering in order to acquire and maintain topology information.

6.4.2 User Constraint Information

User constraint information may be provided to the RDSS by means of a user interface. Typically a user or customer communicates their expectations of the network via the use of a Service Level Agreement (SLA) [13]. This SLA is then transformed into a technical document called a Service Level Specification (SLS) [13] which describes the technical details of how the customer traffic will be treated. SLS configuration parameters that the RDSS store are given in Table 6.1. Extensions to the Common Open Policy Service protocol (COPS) could be used so that the RDSS can communicate policy decisions to the ingress routers.

Remark 23 *RDSS is not concerned with the configuration or provisioning of core routers.*

6.4.3 Calculating and Installing Explicit Paths

Explicit paths can be installed using the MPLS signaling protocol after route calculation by RDSS. When an egress node receives customer traffic requesting use of the network these requests are passed onto the RDSS via COPS. The RDSS then uses information in the TED and the CP-H/RDS algorithm calculate a LSP that the customer traffic should use. This information is communicated to the ingress node by COPS. The ingress node then uses the MPLS explicit option to install the LSP in the core network.

Table 6.1: Contents of the Service level Specification

Content	Description	Information Stored
Scope	The area of the network where the policies outlined for traffic in a given direction within the SLS will be enforced.	ingress - egress {interface identifier (s)}
Flow Identification id	Indicates which IP packets in a stream will be given a particular treatment	source, destination, application information, - Differentiated Services, information Per-Hop-Behavior (PHB)
Traffic Parameters	Traffic parameters indicate what should be done with in- & out-of-profile packets	Token bucket algorithm (b, r) - Token bucket rate (r) - Bucket depth = b
Performance Guarantees	Service guarantees that the network offers to the customer for the packet stream identified by Flow Id over the geographical region defined by Scope	Bandwidth, packet loss, delay (optional), jitter (optional), cost, minimum hop count

6.4.4 Estimating delay and jitter

Metrics like delay and jitter require extensions of the routing protocol. In order for the TED to ascertain delay and jitter from network topology information it is proposed that these values be based on the estimates for link residue bandwidth, link propagation delay and token bucket parameters [26] from routers and switches that use rate proportional policing. The queuing delayand jitter bounds are given by [26]:

$$D(P,r,b) = \frac{b}{r} + \frac{nL_{\max}}{r} + \sum_{j=1}^{n} \frac{L_{\max}}{C_j} + \sum_{j=1}^{n} d_j \tag{6.1}$$

$$J(P,r,b) = \frac{b}{r} + \frac{nL_{\max}}{r} \tag{6.2}$$

where,

$r =$ the bandwidth constrained by a token bucket

$\sigma =$ is the average token rate

$b =$ is the token bucket size

$P =$ given path with n hops

$L_{\max} =$ maximal packet size

$C_j =$ link capacity of link j

$d_j =$ is the propagation delay of each link on path P.

6.4.5 Setting up Reservations

Calculating an appropriate path for a given customer flow in the core network is only part of the job of achieving QoS and resource optimization for the flow. The other part is to reserve or pin the requested resources so that the flow can guarantee consistent QoS. This can be done if network managers specify the treatment of packets at each node, also known as PHB (per-hop behavior), for each class of differentiated service. This information can be stored in the SLS as indicated in Table 6.1. PHB and traffic conditioning information can be passed to the ingress router after the LSP path calculation is done by the RDSS. The ingress node could then use the Constraint-based Routing Label Distribution Protocol (CR-LDP) to enable the specification of QoS and traffic parameters. This includes the specification of service class and traffic descriptors such as bandwidth requirements for all routers/switches on the LSP. In addition, CR-LDP facilitates state management, path-tear-down and maintenance of LSPs.

6.4.6 Path Protection

The CP-H/RDS by design has a high probability of returning disjoint paths that are ranked in a preference ordering that conforms to the optimization goals of the user flow and network carrier. Sharing protection capacity that is assumed to be exempt from simultaneous failure is an efficient solution for providing failed primary paths with backup paths. Hence the CP-H/RDS naturally supports the technique of sharing protection capacity that is presented in [45][47].

6.5 Conclusion

In this chapter a framework under which the CP-H/RDS algorithm can be used in the context of a TE environment was presented. A set of protocols and procedures were proposed that allows the CP-H/RDS algorithm to function in the management plane. The most important advantage in this approach is that there is opportunity for graceful migration to the CP-H/RDS algorithm with little disruption to existing layer 3 operations. Equally important the complexity introduced by QoS awareness remains outside the network.

Chapter 7

User Performance in networks that use CP/RDS

So far simulations were done to study the success rate, execution times and optimization characteristics associated with the CP/RDS algorithm and its variants. In addition, a framework for using the CP-H/RDS in a TE environment was proposed. However, one of the main benefits of QoS routing is to improve user application performance. Consequently, this chapter presents a simulation that implements aspects of the RDSS framework using the ns-simulator [84]. The main goal of the simulation is to investigate the effect that the CP/RDS has on application performance.

7.1 Simulation Model

To evaluate the CP/RDS algorithm as it relates to finding a suitable QoS path for a given flow request, aspects of the RDSS described in Chapter 6 is implemented in C++/TCL and incorporated in network simulator (ns-2). To reduce the amount of coding it is assumed that the RDSS is part of the ingress router. This eliminates the role of the COPS protocol. In addition, no traffic conditioning or traffic shaping strategies are implemented. Furthermore, the RDSS is implemented in such a way that it can use either the SWP or the CP/RDS algorithm to calculate explicit MPLS paths.

Figure 7.1 shows the network topology used in the simulation. Each link in the graph is characterized by bandwidth, delay and a Traffic Source Link Policy (TSLP). The TSLP link metric is added because it is a well known fact that because of TCP's congestion mechanisms, an increase in UDP traffic would affect the performance of TCP traffic. The TSLP is used by the network administrator to help control the

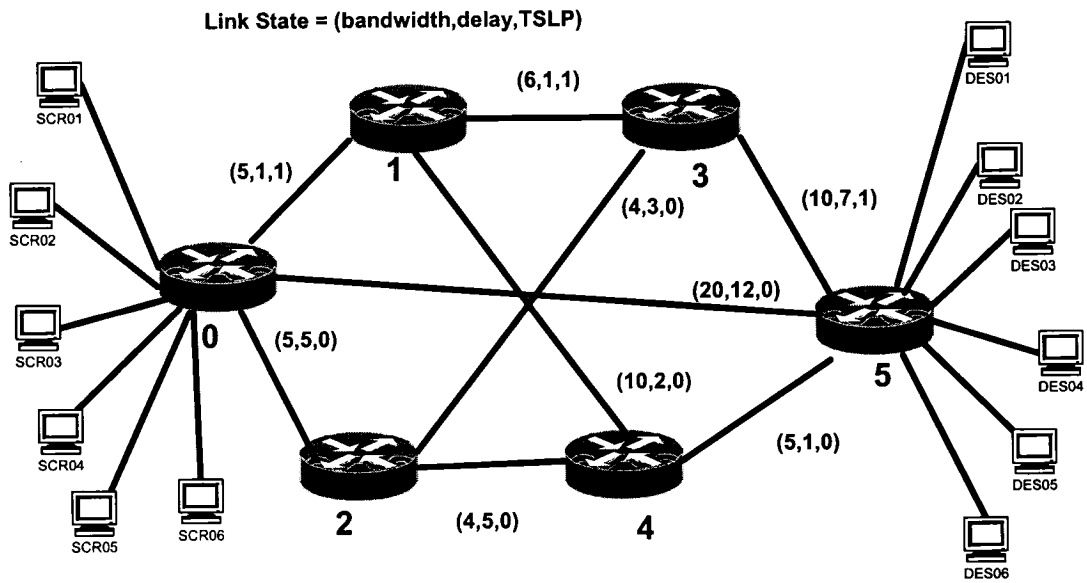


Figure 7.1: Network Topology used in simulation. Each link is symmetrical and characterized by bandwidth, delay and a TSLP value.

degree of TCP and UDP traffic present on a link. TSLP has only two values 0 and 1. A TSLP value of 1 means that only TCP traffic is encouraged on the link, while a TSLP value of 0 means that only UDP traffic is encourage on the link.

Remark 24 *The TSLP can be considered a concave metric. However, it cannot be pruned since links that are set to 1 (TCP encourage traffic) can still carry UDP traffic and links that are set to 0 (UDP encourage traffic) can still carry TCP traffic. This feature demonstrates the power of using the CP/RDS algorithm since the network designer can easily include policy metrics that make it possible to have more control over traffic.*

Simulations were performed to study the impact of the dynamic assignment of QoS paths on network and end-user performance for constant bit rate (CBR) and connection oriented variable bit rate (VBR) traffic with varying demands. The traffic

Table 7.1: Traffic flow configurations

	Flows	Capacity
CBR UDP	SCR01 to DES01 (flow 1)	2M
CBR UDP	SCR02 to DES02 (flow 2)	15M
CBR UDP	SCR03 to DES03 (flow 3)	5M
TCP	SCR04 to DES04 (flow 4)	1M
TCP	SCR05 to DES05 (flow 5)	1M
TCP	SCR06 to DES06 (flow 6)	1M

Table 7.2: Paths selected by algorithms based on the QoS and network state at the time the flow requested admission

Flow ID	Selected Path CP/RDS	Selected Path SWP
1	0_2_3_5	0_1_4_5
2	0_5	0_5
3	0_5	0_5
4	0_2_3_5	0_1_4_5
5	0_1_3_5	0_1_4_5
6	0_1_3_5	0_1_4_5

specifications for the flows used in the simulation are summarized in Table 7.1. Flows 1, 2, 3, 4, 5 and 6 are started are at 21, 31, 41, 51, 61 and 71 seconds respectively for the SWP and CP/RDS algorithms.

7.2 Simulation Results

The main goal of the simulation were to assess the impact that path selection has on receiver throughputs based on existing flows in the network. Tables 7.1 and 7.2 show the paths calculated when SWP and CP/RDS are used respectively. It is clear that the selection policy of the algorithms is different as different paths for the same user requests are produced. An analysis of receiver throughputs is given on a flow by flow basis below.

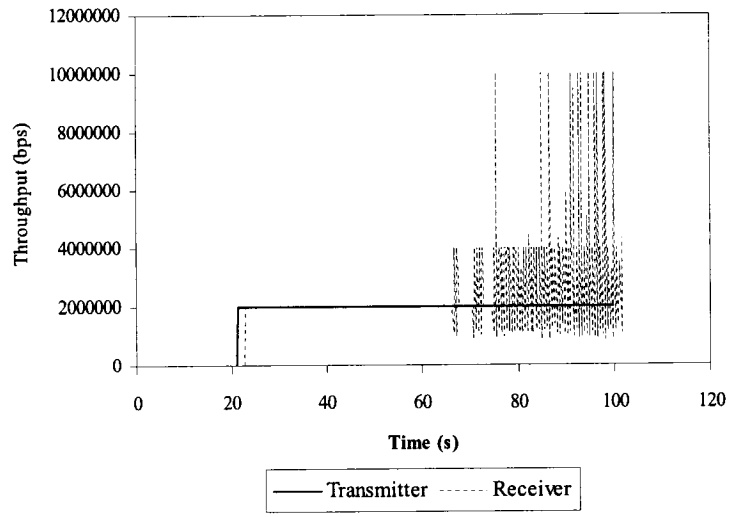


Figure 7.2: Throughputs of source and receiver for flow 1 under the CP/RDS algorithm

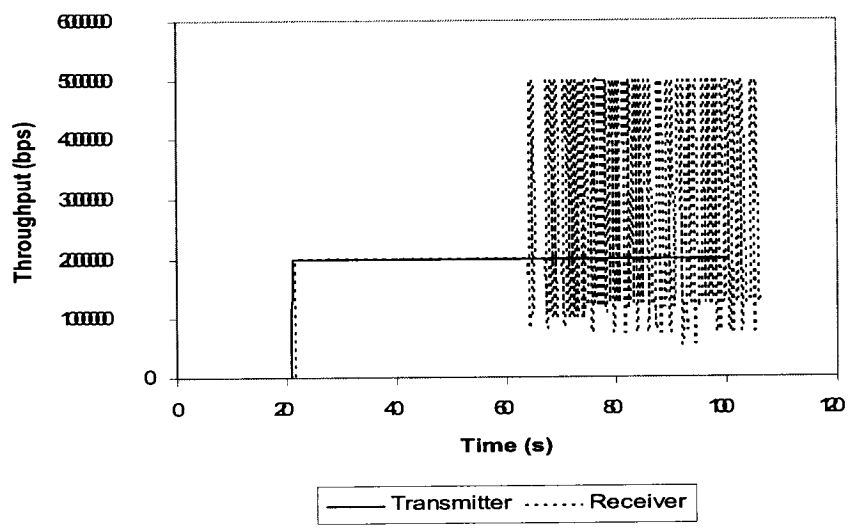


Figure 7.3: Throughputs of source and receiver for flow 1 under the SWP algorithm

Flow 1 Figures 7.2 and Figure 7.3 show the throughputs for the transmitter and receiver for flow 1 when the CP/RDS and SWP algorithms are used respectively. As flows are admitted into the network the graphs show how the throughput of flow 1 is affected. When flows 2 and 3 are started at 31 and 41 seconds respectively, no change in flow 1 throughput is observed. Neither of the algorithms put flows 2 and 3 on the same path as flow 1. When flow 4 is introduced both algorithms placed it on the same path as flow 1. However, flow 1 is not affected by flow 4 since enough link bandwidth is present on the path selected for flow 1. When flow 5 is started 40s after flow 1, fluctuations in receiver throughput is observed for both algorithms. This is because both algorithms selected a path that contained at least one bottleneck link. However, the fluctuation of receiver's throughput in the case of the SWP algorithm is significantly higher than with RDS. This is because the RDS algorithm selected a path with one link common to the path of flow 1, while SWP put flow 5 on the same path as flow 1, thus creating an environment where flows have to compete for resources. Finally, when flow 6 is started 50s after flow 1, an increase in the intensity of the fluctuation of the receiver's throughput is evident. This increase is also present in the case of SWP but difficult to observe since the receiver's throughput fluctuation is so intense as a result of flow 5. Note that the SWP put flow 6 on the same path as flow 1 creating a highly congested path, while RDS selected a path for flow 6 that only have one common link with the path of flow 1.

Flow 2 Figures 7.4 and Figure 7.5 show the throughputs for the transmitter and receiver for flow 2 when the CP/RDS and SWP algorithms are used respectively. Both algorithms selected the same path for flows 2 and 3 which lead to identical graphs for both algorithms. The receiver's throughput corresponded to the transmitter throughput until flow 3 started about 10s after flow 2. Since flow 3 used the same path as flow 2 for both algorithms, a congested link (0_5) is created which accounted for the delay jitter observed.

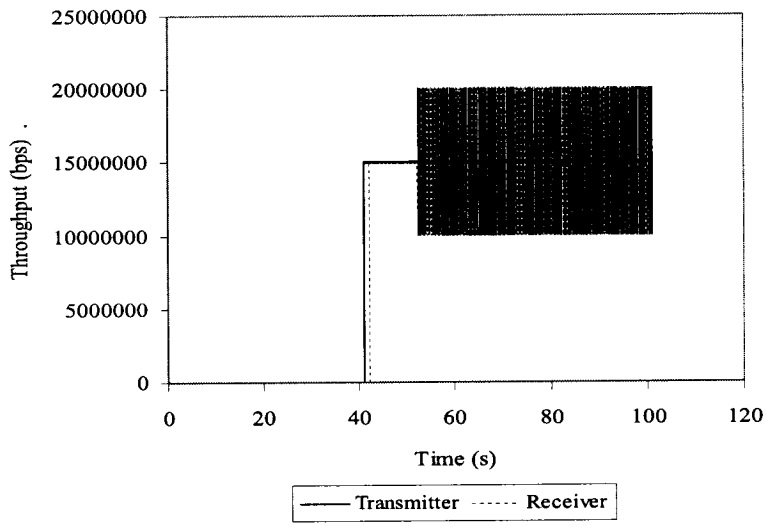


Figure 7.4: Throughputs of source and receiver for flow 2 under the CP/RDS algorithm

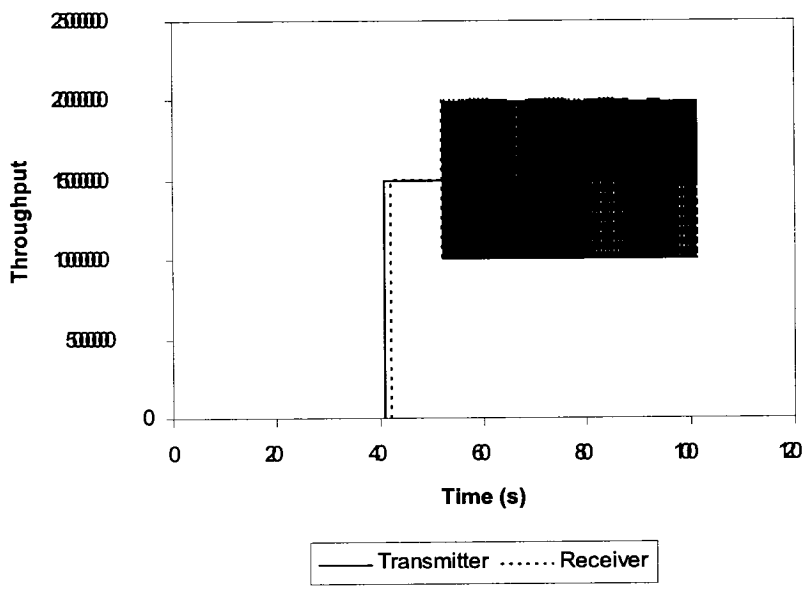


Figure 7.5: Throughputs of source and receiver for flow 2 under the SWP algorithm

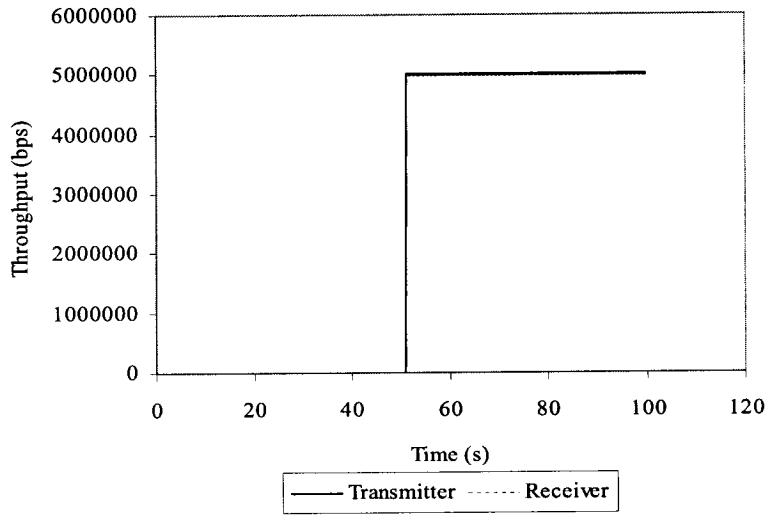


Figure 7.6: Throughputs of source and receiver for flow 3 under the CP/RDS algorithm

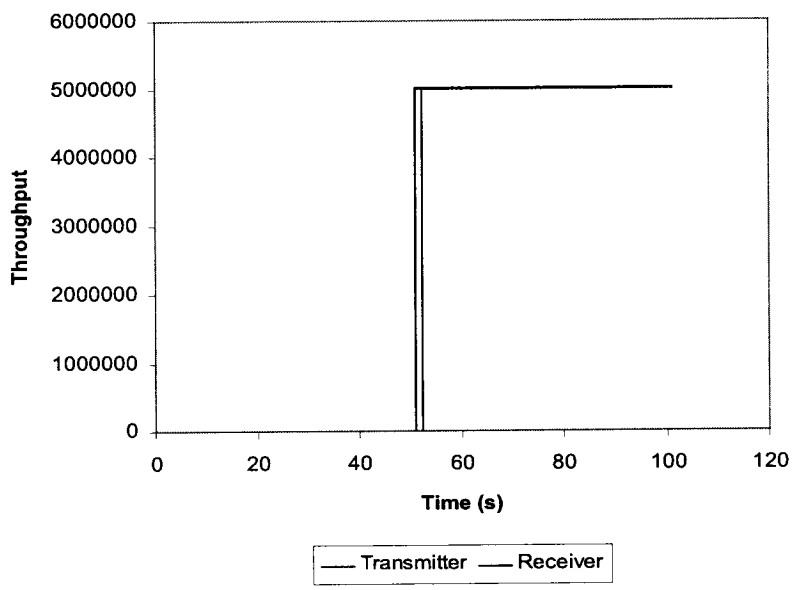


Figure 7.7: Throughputs of source and receiver for flow 3 under the SWP algorithm

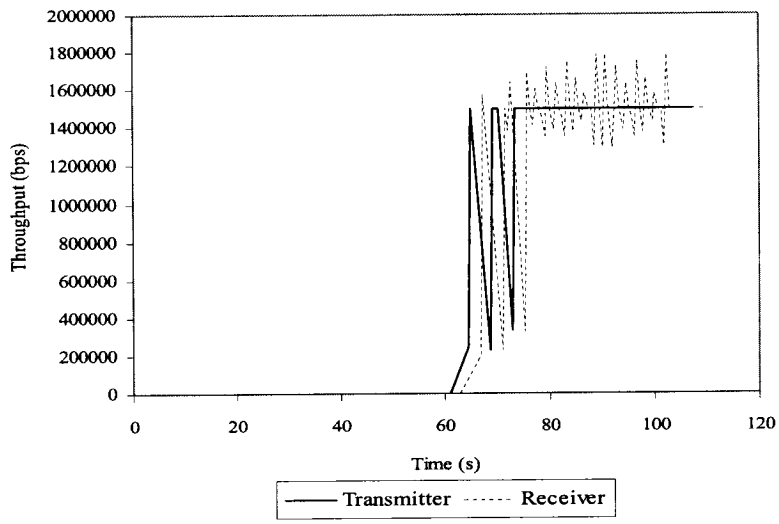


Figure 7.8: Throughputs of source and receiver for flow 4 under the CP/RDS algorithm

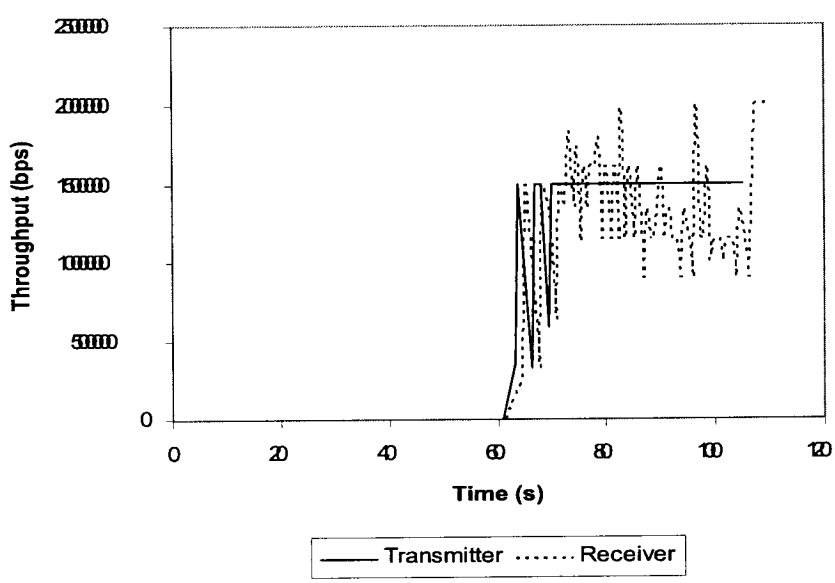


Figure 7.9: Throughputs of source and receiver for flows 4 under the SWP algorithm

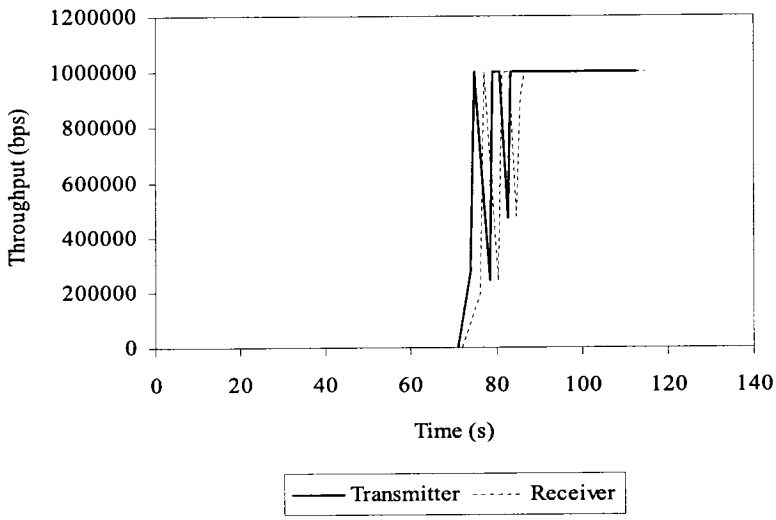


Figure 7.10: Throughputs of source and receiver for flow 5 under the CP/RDS algorithm

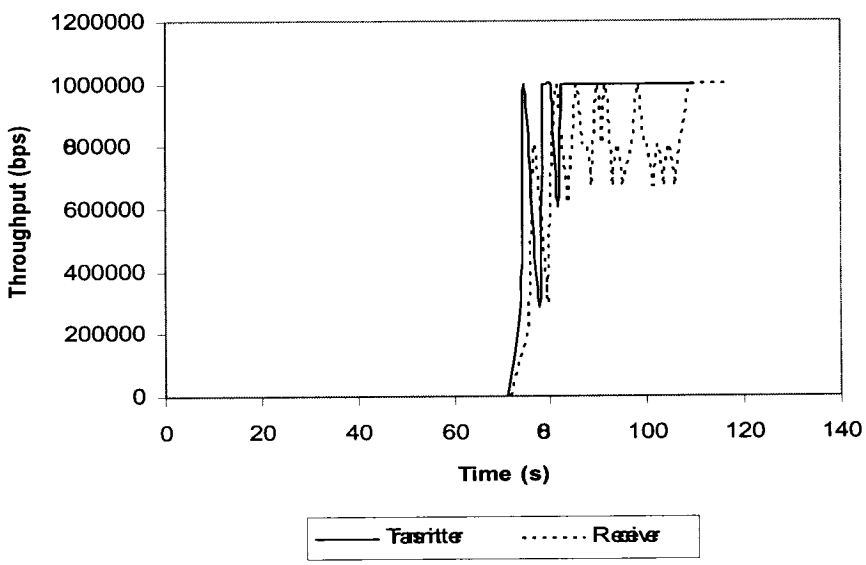


Figure 7.11: Throughputs of source and receiver for flow 5 under the SWP algorithm

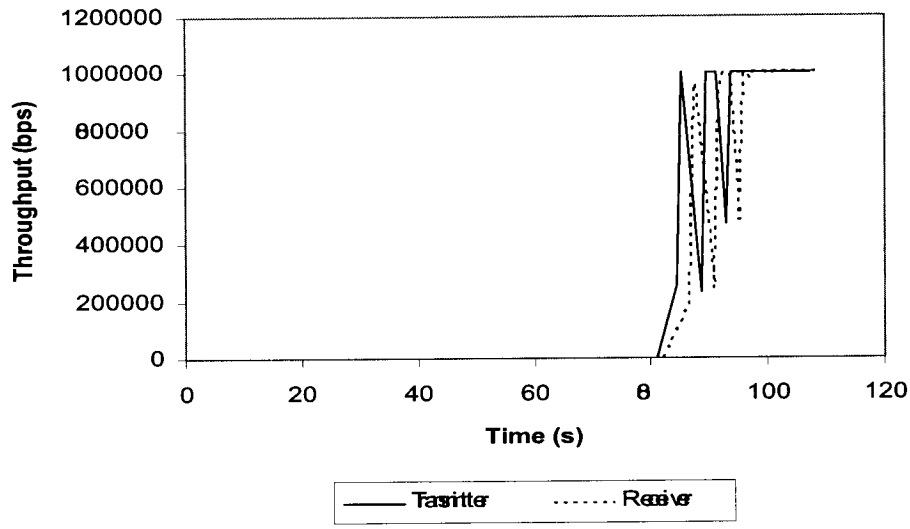


Figure 7.12: Throughputs of source and receiver for flow 6 under the CP/RDS algorithm

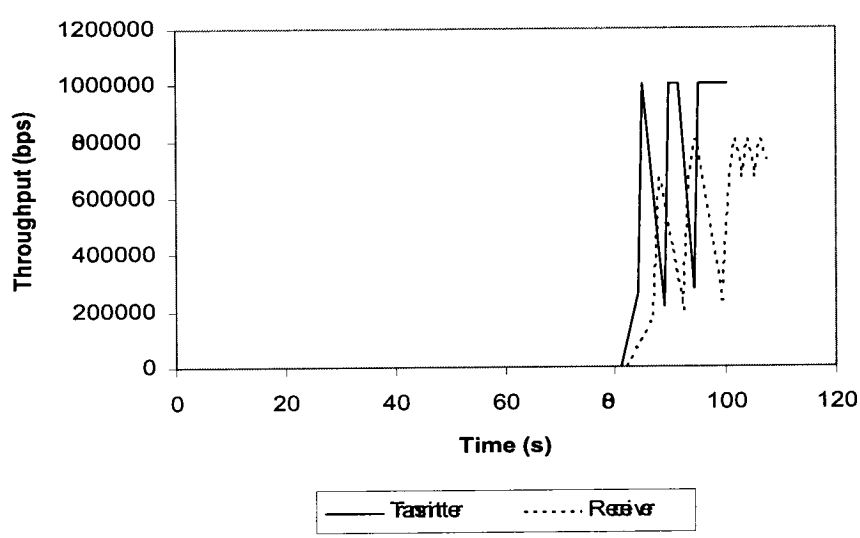


Figure 7.13: Throughputs of source and receiver for flow 6 under the SWP algorithm

Flow 3 Figures 7.6 and Figure 7.7 show the throughputs for the transmitter and receiver for flow 3 when the CP/RDS and SWP algorithms are used respectively. Both algorithms selected the same path for flows 3 which lead to identical graphs for both algorithms.

Flow 4 Figures 7.8 and Figure 7.9 show the throughputs for the transmitter and receiver for flow 4 when the CP/RDS and SWP algorithms are used respectively. Since flow 4 is a TCP flow it transmits until it detects congestion which occurred around 16000 bps. The average receiver throughput for both the CP/RDS and SWP algorithms is equal to the transmitter data rate of 16000bps.

Flow 5 Figures 7.10 and Figure 7.11 show the throughputs for the transmitter and receiver for flow 5 when the CP/RDS and SWP algorithms are used respectively. The RDS selected a path for flow 5 that resulted in the TCP flow having a receiver data rate identical to the transmitter data rate. However, in the case of SWP the average throughput of the receiver is less than the transmitter with significant jitter.

Flow 6 Figures 7.12 and Figure 7.13 show the throughputs for the transmitter and receiver for flow 6 when the CP/RDS and SWP algorithms are used respectively. Similar, to flow 5, the RDS selected a path for flow 6 that resulted in the TCP flow having a very high throughput. However, in the case of SWP the average throughput of the receiver is less than the transmitter with significant jitter.

7.3 Summary

Tables 7.3, 7.4, 7.5, 7.6, 7.7 and 7.8 summarize the simulation results for flows 1, 2, 3, 4, 5 and 6 respectively.

Table 7.3: Flow 1 receiver's throughput status as new flows are admitted into the network.

Algorithm	Flow 2	Flow 3	Flow 4	Flow 5	Flow 6
CP/RDS	no effect	no effect	no effect	fluctuation of receiver's throughput	increase fluctuation of receiver's throughput
SWP	no effect	no effect	no effect	higher fluctuation of receiver's throughput than in the case of CP/RDS	increase fluctuation of receiver's throughput

Table 7.4: Flow 2 receiver's throughput status as new flows are admitted into the network.

Algorithm	Flow 3	Flow 4	Flow 5	Flow 6
CP/RDS	no effect	fluctuation of receiver's throughput	no effect	no effect
SWP	no effect	fluctuation of receiver's throughput similar to CP/RDS	no effect	no effect

Table 7.5: Flow 3 receiver's throughput status as new flows are admitted into the network.

Algorithm	Flow 4	Flow 5	Flow 6
CP/RDS	no effect	no effect	no effect
SWP	no effect	no effect	no effect

Table 7.6: Flow 4 receiver's throughput status as new flows are admitted into the network.

Algorithm	Flow 5	Flow 6
CP/RDS	receiver's throughput closely matches rate	moderate fluctuation in receiver's throughput
SWP	receiver's throughput matches transmission rate	significant fluctuation in receiver's throughput

Table 7.7: Flow 5 receiver's throughput status as new flows are admitted into the network.

Algorithm	Flow 5 (before flow 6)	Flow 6
CP/RDS	receiver's throughput closely matches transmission rate	no effect
SWP	receiver's throughput matches transmission rate	receiver's throughput decreased significantly

Table 7.8: Flow 6 receiver's throughput status.

Algorithm	Flow 6
CP/RDS	receiver's throughput closely matches transmission rate
SWP	receiver's throughput is significantly lower than transmission rate

7.4 Conclusion

The simulation results of this chapter illustrated the importance of having an intelligent path finding algorithm. By using a simple network path selection policy it was shown that receiver throughput and delay jitter could be significantly improved over a path selection policy that only considered bandwidth and hop count. In addition, the ease with which the TSLP policy was incorporated into the CP/RDS illustrates the power of the RDS portion of the CP/RDS algorithm.

Problems associated with using QoS routing in core networks such as routing with imprecise state information and stability of paths can be addressed by introducing network policies into the RDS algorithm. For example, a safety-based metric can be easily added to the RDS algorithm to implement the concept of safety-based routing introduced in the Section 2.7.1.1.

Chapter 8

Conclusion

This thesis presented a multi-constraint optimization algorithm called CP/RDS that searches for paths based on traffic-oriented and resource-oriented objectives. The algorithm operates in two phases. The first phase finds a subset of the constraint paths that exist between a source and destination node, and the second phase uses the power of a multiple optimization algorithm to find a path from the set of paths produced by phase one that best meets the optimization goals of the user and network carrier.

An exact version of the CP/RDS algorithm was proposed first. Results of extensive simulations showed that the exact CP/RDS has a 100% success rate with respect to finding paths that satisfy multiple constraints. However, this ability to find feasible paths can lead to high execution times of the algorithm due to the fact that the problem is NP-complete. When compared to the SAMCRA exact algorithm the CP/RDS algorithm was shown to have lower running times under strict user constraints and significantly lower running times under loose user constraints. However, the average execution times of the CP/RDS algorithm were too high for the application of the algorithm in an online TE environment. This lead to the development of a heuristic version of the CP/RDS algorithm called CP-H/RDS. The CP-H portion of the algorithm finds optimal paths for each metric and then passes all the paths to the RDS portion of the algorithm. Simulation results showed that the CP-H/RDS algorithm has a success rate of between 93 and 96 percent when used in Waxman graph topologies. However, the average running time of the algorithm was shown to be significantly improved over the exact version.

The time and space complexities for the algorithms implemented and analyzed in

this thesis are given in Table 8.1.

Table 8.1: Worst-case time and space complexities for algorithms analyzed in this thesis.

Algorithm	Time	Space
CP/RDS	$O(N\lambda \log(N\lambda) + mE + m(\lambda + 1))$	$O(\lambda mN)$
CP-H/RDS	$O(\lambda(mN \log(N) + mE) + m(\lambda + 1))$	$O(N)$
SAMCRA	$O(Nk_{min}log(Nk_{min}) + k^2mE)$	$O(kmN)$
Iwata	$O(mNlogN + mE)$	$O(N)$
Jaffe	$O(NlogN + mE)$	$O(N)$

The main original contribution of this thesis lies in the novel approach that was used to solve the multiple optimization problem associated with the traffic-oriented and resource-oriented objectives. The use of strong scales provided the basis for constructing a multiple criteria preference function in an affine space. The use of preference functions in turn made it possible for paths that match traffic-oriented and resource-oriented goals to be selected by the algorithm.

This thesis also proposed a framework for implementing the CP/RDS in the management plane rather than the network layer (control plane). Simulations showed that this framework can create a management environment that makes it easy for network administrators to create and deploy network policies that can benefit both the network carrier and traffic flows.

8.1 Future Work

Extensions to research done in this thesis are as follows:

1. The proposed optimization technique used in the RDS algorithm only works with independent decision criteria. However, if the RDS algorithm is used in situations where this independence assumption is not true, or in applications like

multiple objective decision making that involves sub-criteria, then the current model will not work. The model needs to be extended to include sub-criteria.

2. Network security is a very important aspect of network applications. Providing a secure network infrastructure plays an important role in an ISP security solution. Since the RDS algorithm can easily accommodate network policies, future work needs to be done to investigate how an ISP can use the power of the RDS to assist with improving network security.

3. Routing in an ad hoc wireless environment is complicated by the failing and mobility of nodes. Future work needs to be done to investigate how the CP-H/RDS can used to help maintain connectivity in an ad hoc wireless environments.

4. Further work needs to be done to improve both the success rate and execution times of the CP-H/RDS algorithm.

5. This thesis only provided a framework for using the CP/RDS algorithm in a single ISP network. However, extensions of the framework for using the CP/RDS algorithm across multiple ISP domains to achieve end-to-end QoS needs to be investigated.

6. This thesis did not provide a solution to the resource allocation problem that results from the order in which traffic flows are admitted to the network; further research needs to be done to investigate how the power of the RDS algorithm can be used to help solve this problem.

7. The proposed RDSS framework uses dynamic admission control to admit user flows into the network. Future work needs to be done to investigate how pricing and billing can be incorporated into the RDSS framework. One possible approach is to use a two-tariff system that offers users a best effort service at

a fixed rate and the option to purchase additional QoS services based on the application type and the preference scale value of the user flow.

8. The major future work involves implementing the RDSS framework with the CP/RDS algorithm in a TE environment.

Bibliography

[1] A. S. Tanenbaum, *Computer Networks.* New Jersey: Prentice Hall PTR, 4 ed., 2003.

[2] F. Kuipers, T. Korkmaz, M. Krunz, and P. V. Mieghem, "Performance evaluation of constraint-based path selection algorithms," *IEEE Network*, vol. 18, pp. 16–23, 2004.

[3] J. Barzilai, "Theory of measurement: a summary of recent results," in *Proceedings of the NSF Design, Service and Manufacturing Conference*, 2003.

[4] P. V. Mieghem and F. A. Kuipers, "On the complexity of qos routing," *Computer Communications, special issue on QofIS'01*, vol. 26, pp. 376–387, March 2003.

[5] G. Figueiredo, N. da Fonseca, and J. Monteiro, "A minimum interference routing algorithm," *2004 IEEE International Conference on Communications*, vol. 4, pp. 1942– 1947, 2004.

[6] D. Awduche, "Mpls and traffic engineering in ip networks," *Communications Magazine*, vol. 37, pp. 42–47, 1999.

[7] D. Mitra and Q. Wang, "Stochastic traffic engineering, with applications to network revenue management," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 1, pp. 396– 405.

[8] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[9] P. V. Mieghem, H. D. Neve, and F. Kuipers, "Hop-by-hop quality of service routing," *Computer Networks*, vol. 37, pp. 407–423, 2001.

[10] G. Liu and K. Ramakrishnan, "A*prune: An algorithm for finding k shortest paths subject to multiple constraints," *IEEE INFOCOM 2001*, vol. 2, p. 743Ű749, 2001.

[11] J. Barzilai, "Measurement foundations for preference function modelling," in *Proceedings of the IEEE International Conference on Systems*, 1998.

[12] P. V. Mieghem and F. A. Kuipers, "The impact of correlated link weights on qos routing," in *IEEE INFOCOM 2003*, (San Francisco, USA), 2003.

[13] S. Sanjay and M. Hassan, *Engineering Internet QoS*. 685 Canton Street Norwood, MA 02062: Artech House, 1 ed., 2002.

[14] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the internet," internet draft, December 1997.

[15] E. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol label switching architecture," RFC RFC 3031, January 2001.

[16] S. Chen and K. Nahrstedt, "On finding multi-constrained paths," in *Conference Record.1998 IEEE International Conference on Coomunications*, 1998.

[17] G. Apostolopoulos, "Intra-domain qos routing in ip networks: A feasibility and cost/benefit analysis," *IEEE Network*, vol. 13(5), pp. 42–54, 1999.

[18] G. Apostolopoulo and T. Przygienda, "Qos routing mechanisms and ospf extensions," *RFC 2676, IETF*, August 1999.

[19] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Inc.: Prentice Hall, 1993.

[20] A. M. S. Alkahtani, M. E. Woodward, K. Al-Begain, and A. Alghannam, "Qos routing with multiple and prioritised constraints using the concept of the analytic hierarchy process (ahp)," *International Network Conference,INC2002, UK*, pp. 243–251, 2002.

[21] M. M. M. Al-Fawaz and M. E. Woodward, "Fast quality of service routing algorithms with multiple constraints," *8th IFIP Workshop on ATM/IP, Ilkely, UK*, pp. 01/1–01/10, 2000.

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

[23] R. Widyono, "The design and evaluation of routing algorithms for real-time channels," technical report icsi tr-94-024, International Computer Science Institute, U.C. Berkley, June 1994.

[24] R. Hassin, "Approximation schemes for the restricted shortest path problem," *Mathematics of Operations research*, pp. 36–42, February 1992.

[25] H.Salama, D. Reeves, and Y.Viniotis, "A distributed algorithm for delay-constrained unicast routing," in *IEEE INFOCOM Ś97*, 1997.

[26] Q. Ma and P. Steenkiste, "Quality of service routing with performance guarantees," in *4th International IFIP Workshop on QoS*, May 1997.

[27] J. Bennet and H. Zhang, "Hierarchical packet fair queueing algorithms," in *ACM SIGCOMM Š96*, 1996.

[28] D. Clark and J. Pasquale, "Strategic directions in networks and telecommunications," *ACM Computing Surveys*, vol. 28(4), pp. 579–690, 1996.

[29] L. Hui-Lan and I. Faynberg, "An architectural framework for support of quality of service in packet networks," *Communications Magazine IEEE*, vol. 41, pp. 98–105, 2003.

[30] A. Juttner, B. Szviatovszki, I. Mecs, and Z. Rajko, "Lagrange relaxation based method for the qos routing problem," in *INFOCOM2001*, pp. 859–868, 2001.

[31] G. Cheng, Y. Tian, and N. Ansari, "A new qos routing framework for solving mcp," *IEICE Transaction in Communications*, vol. E86-B, no. 3, pp. 534–541, 2003.

[32] L. Guo and I. Matta, "Search space reduction in qos routing," in *IEEE INFOCOMM Conference*, pp. 142–149, May 1999.

[33] E. Chong, S. Maddila, and S. Morley, "On finding single-source single-destination k shortest paths," *J. Computing and Information, special issue ICCI'95*, pp. 40–47, 1995.

[34] S. Chen and K. Nahrstedt, "An overview of quality-of-service routing for the next generation high- speed networks: Problems and solutions," *in IEEE Network, Special Issue on Transmission and Distribution of Digital Video*, pp. 64–79, November-December 1998.

[35] A. Iwata, R. Izmailov, D.-S. Lee, B. Sengupta, G. Ramamurthy, and H. Suzuki, "Atm routing algorithms with multiple qos requirements for multimedia internetworking," *IEICE Transactions Communications*, vol. E79-B, pp. 281–291, August 1996.

[36] R. Motwani and P. Raghavan, *Randomized Algorithms.* Cambridge University Press, 1995.

[37] T. Korkmaz and M. Krunz, "A randomized algorithm for finding a path subject to multiple qos requirements," *Computer Networks*, vol. 36, p. 251Ű268, 2001.

[38] T. Korkmaz and M. Krunz, "Multi-constrained optimal path selection," in *Proceedings of the IEEE INFOCOM 2001 Conference*, vol. 2, (Alaska), p. 834Ű843, April 2001.

[39] H. D. Neve and P. V. Mieghem, "Tamcra: A tunable accuracy multiple constraints routing algorithm," *Computer Communications*, vol. 23, p. 667Ű679, 2000.

[40] P. V. Mieghem, "Paths in the simple random graph and the waxman graph," *Probability in the Engineering and Information Sciences (PEIS)*, vol. 15, pp. 535–555, 2001.

[41] B. Waxman, "Routing of multipoint connections," *IEEE Journal of Selected Areas in Communications*, pp. 1617–1622, December 1988.

[42] V. Chankong and Y. Y. Haimes, *Multiobjective decision making theory and methodology.* New York: North-Holland, 1983.

[43] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, and J. Rexford, "Netscope: Traffic engineering for ip networks," *IEEE Network*, vol. 14, pp. 11–19, 2000.

[44] K. K. D. Katz, D. Yeung, "Traffic engineering extensions to ospf version 2," *INTERNET-DRAFT, draft-katz-yeung-ospf-traffic-09.txt*, 2002.

[45] M. Kodialam and T. V. Lakshman, "Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information," *IEEE/ACM*, vol. 7, no. 3, 2001.

[46] X. S. B. Wang and C. Chen, "A new bandwidth guaranteed routing algorithm for mpls traffic engineering," in *International Conference on Communications*, 2002.

[47] J. Marzo, E. Calle, C. Scoglio, and T. Anjah, "Qos online routing and mpls multilevel protection: a survey," *Communications Magazine IEEE*, vol. 41, pp. 126–132, 2003.

[48] K. Kar, M. Kodialam, and T. V. Lakshman, "Minimum interference routing with applications to mpls traffic engineering," in *IEEE INFOCOM*, vol. 2, (Tel-Aviv, Israel), pp. 884–893, 2000.

[49] A. Bagula, M. Botha, and A. Krzesinki, "The least interference optimization algorithm," *2004 IEEE International Conference on Communications*, vol. 2, pp. 1232– 1236, 2004.

[50] E. Crawley, R. Nair, B. Jajagopalan, and H. Sandick., "A framework for qos-based routing in the internet," *RFC, http://www.ietf.org/rfc/rfc2386.txt*, 1998.

[51] R. Guerin and A. Orda, "Qos-based routing in networks with inaccurate information: Theory and algorithms," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 350–364, June 1999.

[52] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi, "Improving qos routing performance under inaccurate link state information," in *16th International Teletraffic Congress (ITC Š16)*, 1999.

[53] S. Nelakuditi, Z. Zhang, R. Tsang, and H. David, "Adaptive proportional routing: A localized qos routing approach," 2002.

[54] S. Kim and M. Lee, "Server based qos routing with implicit network state updates," *IEEE GLOBECOM*, vol. 4, pp. 2182–2187, November 2001.

[55] R. Guerin, S. Kamat, and S. Herzog, "Qos path management with rsvp," Internet Draft draft-qos-path-mgmt-rsvp00.txt, 1997.

[56] Z. Zhang, C. Sanchez, B. Salkewicz, and E. Crawley, "Quality of service extensions to ospf or quality of service path first routing (qospf)," Internet Draft draft-zhang-qos-ospf-01, 1997.

[57] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, vol. 14, p. 95Ű116, 1984.

[58] M. Henig, "The shortest path problem with two objective functions," *European Journal of Operational Research*, vol. 25, pp. 281–291, 1985.

[59] F. Kuipers and P. V. Mieghem, "Non-dominance in qos routing: an implementational perspective," *IEEE Communications Letters*, 2004.

[60] D. Estrin, R. Govindan, S. Shenker, and H. Tangmunarunkit, "The impact of routing policy on internet paths," in *INFOCOM*, 2001.

[61] I. Juva, *Analysis of Quality of Service Routing Approaches and Algorithms.* Unpublished doctoral dissertation, Helsinki University of Technology, Finland, 2003.

[62] J. Barzilai, "On the decomposition of value functions," *Operations Research Letters*, vol. 22, no. 4, 1998.

[63] W. Goodridge, W. Robertson, W. Phillips, and S. Sivakumar, "Traffic driven multiple constraint-optimization for qos routing," *Inaugural issue of International Journal of Internet Protocol Technology*, vol. To Appear, 2005.

[64] W. Goodridge, W. Robertson, W. Phillips, and S. Sivakumar, "Comparing a novel qos routing algorithm to standard pruning techniques used in qos routing algorithms," in *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 805–808, 2004.

[65] W. Goodridge, W. Robertson, W. Phillips, and S. Sivakumar, "Multiple metric qos in differentiated services networks using preference functions measurement concepts," in *ICOIN 2004*, vol. 3090 of *Lecture Notes in Computer Science*, pp. 390–399, Springer, 2004.

[66] W. Goodridge, W. Robertson, W. Phillips, and S. Sivakumar, "Over-constraint qos routing in large networks," in *London Communications Symposium 2004*, vol. 2, pp. –, 2004.

[67] H. Tyan, C. Hou, , and Y. Chen, "Qos extension to the core-based tree protocol," in *9th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 1999)*, June 1999.

[68] K. Deb, "Evolutionary algorithms for multi-criterion optimization in engineering design," in *Proceedings of Evolutionary Algorithms in Engineering and Computer Science (EUROGEN-99)*, 1999.

[69] C. T.H., L. C.E., and R. R.L., *Introduction to algorithms.* Cambridge, MA, ISBN 0-262-03141-8: The MIT Press, 1997.

[70] R. L. Keeny and H. Raiffa, *Decisions with Multiple Objectives.* Wildey, 1976.

[71] J. Barzilai, "A new methodology for dealing with contradicting engineering design criteria," in *Proceedings of the 18th Annual Meeting of the American Society for Engineering Management*, pp. 73–79, 1997.

[72] J. Barzilai, "Notes on measurement and decision theory," in *Proceedings of the NSF Design and Manufacturing Research Conference*, 2002.

[73] J. Barzilai, "On the foundations of measurement," in *Proceedings of the IEEE International Conference on Systems*, 2001.

[74] J. Barzilai, "Theory of measurement for voltage and affine variables," in *Proceedings of the IEEE International Conference on Systems*, 2000.

[75] J. Barzilai, "Deriving weights from pairwise comparison matrices," *Journal of the Operational Society*, vol. 48, no. 12, pp. 1226–1232, 1997.

[76] M. A. Weiss, *Data Structures and Algorithm Analysis in Java.* Indiana, 46240: Addison-Wesley Professional, 2 ed., 1998.

[77] A. M. I. Matta and J. Byers, "Brite: A flexible generator of internet topologies," internet draft, Boston MA, 2000.

[78] M. Hongyun, X. Linying, L. Zijian, and Z. Lianfang, "End-to-end qos implement by diffserv and mpls," in *Canadian Conference on Electrical and Computer Engineering*, vol. 2, pp. 641– 644, May 2004.

[79] N. Rouhana and E. Horlait, "Differentiated services and integrated services use of mpls," in *Fifth IEEE Symposium on Computers and Communications*, pp. 194–199, 2000.

[80] G. Yuan, W. Wang, Y. Lin, and S. Cheng, "A qos management architecture for diffserv-aware mpls-based ip network," in *International Conference on Communication Technology*, vol. 2, pp. 1603– 1607, 2003.

[81] F. L. Faucheur, T. D. Nadeau, A. Chiu, and W. Townsend, "Requirements for support of diff-serv-aware mpls traffic engineering," *IETF Internet drafts*, 2000.

[82] F. L. Faucheur, T. D. Nadeau, A. Chiu, and W. Townsend, "Extensions to rsvp-te and cr-ldp for support of diff-servaware mpls traffic engineering," *IETF Internet drafts*, 2000.

[83] P. Aukia, M. Kodialam, P. Koppol, T. Lakshman, H. Sarin, and B. Suter, "Rates: a server for mpls traffic engineering," *Network, IEEE*, vol. 14, pp. 34–41, 2000.

[84] K. Fall, "Network emulation in the vint/ns simulator," in *IEEE International Symposium on Computers and Communications*, pp. 244–250, 1999.

Appendix A

Existing QoS Architectures

Many Quality-of-service (QoS) models have been proposed by the Internet Engineering Task Force (IETF) to address the demand of applications requesting specific services from the network. Among them are the integrated services/Resource Reservation Protocol(RSVP) model, the differentiated services (DiffServ) model, and multiprotocol label switching(MPLS). These models are briefly discussed below.

A.1 Integrated Services (IntServ)

The IntServ framework was developed within IETF to provide individualized QoS guarantees to individual sessions. IntServ provides services on a per flow basis where a flow is a packet stream with common source address, destination address and port number. An IntServ router must maintain per flow state information.

The two key IntServ features are:

1. Reservation of resources

   The router must maintain current resource status such as reserved bandwidth and buffer usage. This allows the router to be able to decide whether or not it can facilitate a flow request.

2. Call Setup

   A flow requiring QoS guarantees must be able to reserve sufficient resources at each router on the path to ensure QoS requirements are met. The QoS requirements are communicated to each router by using a Rspec and a TSpec.

Signaling for call setup is done by a signaling protocol. RSVP is the signaling protocol of choice.

A.2 Reservation protocol (RSVP)

RSVP is used to specify the QoS by both hosts and routers. Hosts use RSVP to request a QoS level from the network on behalf of an application data stream. Routers use RSVP to deliver QoS requests to other routers along the path(s) of the data stream. In doing so, RSVP maintains the router and host state to provide the requested service. To initiate an RSVP multicast session, a receiver first joins the multicast group specified by an IP destination address by using the Internet Group Membership Protocol (IGMP). After the receiver joins a group, a potential sender starts sending RSVP path messages to the IP destination address. The receiver application receives a path message and starts sending appropriate reservation-request messages specifying the desired flow descriptors using RSVP. After the sender application receives a reservation-request message, the sender starts sending data packets.

A.3 Differentiated Services (DiffServ)

The DiffServ architecture is a proposed model to address adaptive QoS issues in IP networks. The model uses a marking technique to classify packets at the edge of a network domain and devices in the core of the network use these markings on the packets (called per-hop behavior or PHB code points) to help enforce network traffic policies. This approach is attractive because it is simple and scalable. End-to-End QoS is obtained by the concatenation of per-domain services and SLAs between adjoining domains along the path that the traffic crosses in going from source to destination.

Traffic enters a DiffServ domain at an ingress node and leaves the domain at an egress node. A DiffServ ingress node is responsible for ensuring that the traffic

entering the DiffServ domain conforms to any SLAs between it and the other DiffServ domain or stub network to which the ingress node is connected. A DiffServ egress node may perform traffic conditioning functions on traffic forwarded to a directly connected peering domain, depending on the details of the SLA between the two domains.

A DiffServ router tasks can be divided as follows:

1. Defining packet treatment classes
2. Specifying the amount of resources for each class
3. Sorting all incoming packets into their corresponding classes

The Diffserv effort addresses both task 1 and 3 since it specifies traffic classes as well as provides a simple packet classification mechanism. Diffserv Routers also easily sort packets into their corresponding treatment classes by the TOS value, without having to know which flows or what types of applications the packets belong to.

A.4 Multiple Protocol Label Switching (MPLS)

MPLS is essentially a labeling system designed to accommodate multiple protocols. The use of MPLS labels enables routers to make a routing decision without the processing overhead of checking each packet. MPLS uses a fixed-length label to decide how a packet should be handle.

Each MPLS packet has a header. The header contains a 20-bit label, a 3-bit Class of Service (COS) field, a 1-bit label stack indicator, and an 8-bit Time-to-Live (TTL) field. The MPLS header is encapsulated between the link layer header and the network layer header. An MPLS-capable router, call the labels witched router (LSR), examines only the label in forwarding the packet.

In MPLS, incoming packets are classified and routed at the ingress LSRs of an MPLS-capable domain. MPLS headers are inserted which direct traffic onto virtual

circuits called label-switched paths (LSPs). The LSP between two routers can be the same as the Layer 3 hop-by-hop route, or the sender LSR can specify an explicit route (ER) for the LSP. The ability to setup ERs is a very useful feature of MPLS since this can facilitate traffic engineering.

A.5 Traffic Engineering

Traffic Engineering (TE) provides a means of controlling traffic through a network, offering services tailored for traffic flow requirements while ensuring economical use of network resources. The evolution of TE is largely due to the fact that shortest paths algorithms used in protocols like OSPF has significant limitations. One such limitation is the recurrent points of congestion and bottlenecks in the links of the calculated shortest path while other links are under utilized.

The key to a TE solution is how the traffic mappings are created. There are two main ways that TE algorithms can be executed; offline (pre-computation) and online (on demand). TE is usually associated with offline routing and assumes that all the traffic requests are known at the time of running the TE algorithm. The main objective is to find a feasible path for all requests while minimizing the total network resource usage. With an offline algorithm this objective is possible since algorithms have a longer time frame to execute and all the requests are known upfront.

In online routing algorithms path requests are attended to in a one by one fashion. Constraint-based routing (CBR) is used to compute routes subject to a set of network and policy constraints. Examples of network metrics include bandwidth, delay and packet loss while an example of a policy constraint is the routing of TCP traffic over selected links only. However, to accomplish CBR each router that takes part in the path selection process must have a complete picture of the topology and link state of the network. One way of accomplishing this is to extend current protocols IGPs like OSPF to carry additional network resource attributes. Another more efficient

way, is to use a centralized approach similar to the one proposed in [83], where a routing server stores a traffic engineering database (TED) consisting of customer flow needs and network topology information, and make routing decisions using a CBR algorithm. One possible way of obtaining the network topology information is by reading the MIBs of core routers using SMNP.

Online TE is more desirable than offline TE because it is dynamic and reacts in real time to changes in the network and user requests. However, online TE depends on CBR algorithms which can pose serious difficulties when too many network metrics are considered. Another, disadvantage of online TE is related to the optimization that takes into account all traffic requests in the sequence they are proposed to the network. This approach is not deterministic and there is no way of determining that the order in which paths are setup will lead to an optimal solution. Hence using an online TE can lead to a non-optimal allocation of resources like bandwidth.

Traffic engineering solutions are also concerned with improving the reliability of the network and protecting the network against failures. Fault-tolerant service is an essential requirement of traffic engineering, because a single failure of network components may result in a huge amount of user traffic loss and disruption, which may lead to a large loss of revenue.

Appendix B

Graph Topologies

B.1 Waxman Graphs

In a real network, two nodes close to each other are, of course, more likely to be connected by a link. The Waxman graph [41] is a variation of the random graph where the probability of there being a link between two nodes decreases as the distance between the nodes increases. The Waxman's method starts by placing the nodes of the network randomly on a two-dimensional grid. Then, a link between any pair of two distinct nodes u and v is added according to the following probability function:

$$p(u,v) = \alpha e^{\frac{-d(u,v)}{\beta L}}$$

where $0 \leq \alpha,\beta \leq 1$, $d(u,v)$ is the Euclidian distance between u an v, and L, the maximum Euclidean distance between any two nodes.

B.2 Barabasi Graphs

A common property of many large networks is that the vertex connectivity follows a scale-free power-law distribution. This feature is evident because of the following: (i) networks expand continuously by the addition of new vertices, and (ii) new vertices attach preferentially to sites that are already well connected. To generate power law graphs a model is proposed in [41] which involves starting with a small number of nodes (n_0), and repeatedly adding new nodes with m edges. These edges are preferentially attached to nodes with high degree. By using mean-field theory is can be shown that in a graph with n nodes the probability $p(k)$ that a node is connected to k other nodes is:

$$p(k) = \frac{2m^2(n-n_0)}{n}\frac{1}{k^3}$$

B.3 Brite Parameters

Table B.1: Description of Brite parameters

Parameter	Meaning	Values
HS	Size of one side of the plane	integer > 1
LS	Size of one side of a high-level square	integer > 1
N	Number of nodes	$1 \leq N \leq HS \times HS$
alpha	Waxman-specific exponent	$0 < \alpha \leq 1$; $\alpha \in \mathbb{R}$
beta	Waxman-specific exponent	$0 < \beta \leq 1$; $\beta \in \mathbb{R}$
Node Placement	how nodes are placed in the plane	1:Random, 2:HeavyTailed
m	Number of links per new node	integer > 1
Growth Type	how nodes join the topology	1: Incremental, 2: Random

Appendix C

Group Theory Definitions

Definition 14 (Group) *A group G is a set which is equipped with one operation, say $\circ$ such that:*

1. *$\circ$ is a close operation*

   $c = a \circ b \qquad \forall\ a, b, c \in G$

2. *$\circ$ is associative*

   $(a \circ b) \circ c = a \circ (b \circ c)$

3. *Identity is present in*

   $a \circ e = a$

   $e \circ a = a$

4. *Each element in G has an inverse that is:*

   $a \circ (a^{-1}) = e$

Definition 15 (Cummutative Group) *A commutative group is a group that have the following property:*

$a \circ b = b \circ a$ *(commutative property)*

Definition 16 (Field) *A field F is a set with addition and multiplication operations such that:*

1. F is a commutative group under addition

2. $F - \{0\}$ is a commutative group under the operation of multiplication.

3. The distributive rule is obeyed:

   $a \times (b + c) = a \times b + a \times c$

Definition 17 (Vector Space) *Vector Space is a system consisting of a set of generalized vectors and a field of scalars, having the same rules for vector addition and scalar multiplication as physical vectors and scalars.*

Appendix D

Details of Sample Data

Table D.1: 95% Confidence Intervals for mean execution times of Figure 4.12.

m	Algorithm	Mean time	95% Confidence intervals
2	CP/RDS	10.11	(9.61, 10.61)
	SAMCRA	12.72	(12.02, 13.42)
4	CP/RDS	20.48	(19.00, 21.96)
	SAMCRA	35.24	(29.76, 40.72)
6	CP/RDS	23.77	(21.15, 26.38)
	SAMCRA	44.50	(34.77, 54.23)
8	CP/RDS	30.73	(27.31, 34.14)
	SAMCRA	67.25	(50.11, 84.39)

Table D.2: 95% Confidence Intervals for mean execution times of Figure 4.13.

m	Algorithm	Mean time	95% Confidence intervals
2	CP/RDS	13.91	(13.42, 14.40)
	SAMCRA	19.96	(19.04, 20.90)
4	CP/RDS	29.13	(28.00, 30.27)
	SAMCRA	66.75	(56.42, 77.08)
6	CP/RDS	45.72	(43.64, 47.80)
	SAMCRA	147.95	(119.40, 176.50)
8	CP/RDS	54.74	(51.49, 57.98)
	SAMCRA	200.25	(162.83, 237.66)

Table D.3: 95% Confidence Intervals for mean execution times of Figure 4.14.

m	Algorithm	Mean time	95% Confidence intervals
2	CP/RDS	8.33	(7.85, 8.81)
	SAMCRA	13.87	(12.70, 15.04)
4	CP/RDS	15.02	(13.62, 16.41)
	SAMCRA	21.08	(19.06, 23.10)
6	CP/RDS	13.03	(11.50, 14.55)
	SAMCRA	17.95	(15.38, 20.53)
8	CP/RDS	10.38	(8.79, 11.98)
	SAMCRA	13.00	(10.97, 15.04)

Table D.4: 95% Confidence Intervals for mean execution times of Figure 4.15.

m	Algorithm	Mean time	95% Confidence intervals
2	CP/RDS	11.48	(11.00, 11.96)
	SAMCRA	21.09	(19.76, 22.42)
4	CP/RDS	39.70	(37.63, 41.77)
	SAMCRA	60.83	(57.18, 64.46)
6	CP/RDS	65.17	(61.59, 68.75)
	SAMCRA	95.38	(89.30, 101.47)
8	CP/RDS	90.27	(84.78, 95.77)
	SAMCRA	118.74	(110.83, 126.65)

Table D.5: 95% Confidence Intervals for mean execution times of Figure 5.4.

m	Algorithm	Mean time	95% Confidence intervals
2	CP/RDS	10.11	(9.61, 10.61)
	SAMCRA	12.72	(12.02, 13.42)
	CP-H/RDS	5.27	(5.05, 5.49)
	Iwata	1.96	(1.79, 2.13)
4	CP/RDS	20.48	(19.00, 21.96)
	SAMCRA	35.24	(29.76, 40.72)
	CP-H/RDS	13.23	(12.91, 13.55)
	Iwata	2.24	(2.05, 2.43)
6	CP/RDS	23.77	(21.15, 26.38)
	SAMCRA	44.50	(34.77, 54.23)
	CP-H/RDS	18.95	(18.54, 19.35)
	Iwata	1.99	(1.83, 2.17)
8	CP/RDS	30.73	(27.31, 34.14)
	SAMCRA	50.12	(50.12, 84.39)
	CP-H/RDS	31.59	(30.96, 32.22)
	Iwata	2.10	(1.92, 2.28)