A STOCHASTIC APPROACH TO SOLVING THE WEIGHT SETTING PROBLEM IN OSPF NETWORKS

by

SHAIK MUZIBUR REHMAN

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

In Partial Fulfillment of the Requirements
for the Degree of

MASTER OF SCIENCE

IN

COMPUTER ENGINEERING

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
Dhahran, Saudi Arabia
DEC 2007

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by Shaik Muzibur Rehman under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER ENGINEERING.**

Thesis Committee

Dr. Mohammed H. Sqalli (Chairman)

Dr. Sadiq M. Sait (Member)

Dr. Marwan Abu – Amara (Member)

Dr. Adnan Gutub
Department Chairman

Dr. Mohammed S. Al – Homoud
Dean of Graduate Studies

Date

Acknowledgements

In the name of Allah, Most Gracious, Most Merciful. Praise be to Allah, the Cherisher and Sustainer of the worlds and Master of the Day of Judgement. Peace and blessing of Allah be upon the Prophet Muhammad (PBUH). As I begin this, I wish to acknowledge and honor the people who have been instrumental in allowing this project to be completed.

Firstly, I take this opportunity to thank my committee members. I would like to convey my deep sense of gratitude to my thesis advisor, Dr. Mohammed H. Sqalli for his relentless encouragement and patience throughout the duration of this project. The amazing attitude of Dr. Sadiq M. Sait, committee member, helped in straining every nerve from the discussions to the implementations. I gratefully acknowledge the support of Dr. Marwan Abu-Amara, committee member, for his sincere efforts throughout the thesis work. I would like to convey my appreciativeness to Computer Engineering department of King Fahd University of Petroleum & Minerals for supporting me with the necessary infrastructure.

I would like to deeply thank my parents, sisters and brothers-in-law for the constant source of inspiration through out my career, and my teacher and guide, V.V.Ajith Kumar, who had been with me in all the pros and cons of my life right from the day I came to conscience. I cannot leave KFUPM without mentioning the support of my friends Shafeeq, Faisal, Muqtadir, Waseem, Shujath, Ilyas and other buddies of graduation and undergraduation.

Contents

List of Tables

List of Figures

THESIS ABSTRACT

Name:	SHAIK MUZIBUR REHMAN
Title:	A STOCHASTIC APPROACH TO SOLVING THE WEIGHT SETTING PROBLEM IN OSPF NETWORKS
Major Field:	COMPUTER ENGINEERING
Date of Degree:	DECEMBER 2007

In the world of Internetworks, to maintain a good connectivity of household, business and commercial computing, an extraordinary talent is important. Unpredictable dysfunction in its proper administration adds to the problems of this sophisticated network. One of the contributions in attempting to maintain the proper functioning of internetworking is made by the Open Shortest Path First (OSPF) protocol. It is a link state protocol designed to overcome the gap created by the Routing Information Protocol (RIP) in the internetworking domain. OSPF calculates the shortest paths from each source to all destinations using the Dijkstra's algorithm based on the weights assigned to the links. In the past, various attempts have been made to resolve the congestion issues of Traffic Engineering. With such complex issues in the frameset, assigning weights to these large networks, resulting in the best cost is an NP-hard problem. In this thesis, an approach of mitigating the mentioned problem by using a Stochastic Evolution (StocE) heuristic is used which provides a close to optimal solution to these kinds of problems. Through this work, an attempt has been made to optimize the weights on the network so as to minimize congestion. This approach is well supported by the results embedded towards the end of this work. Another core issue addressed in this work is the improvement of the network by considering single link failure scenarios. Two innovative strategies have been developed, where the same set of optimized weights for both topologies, i.e., with-link and without-link-failure, have been considered. These approaches resulted in robust solutions, fulfilling the constraints levied by the networking world, especially for OSPF networks.

MASTER OF SCIENCE DEGREE

King Fahd University of Petroleum and Minerals, Dhahran.
DECEMBER 2007

Chapter 1

Introduction

1.1 Introduction

The Internet is a large network formed out of more than 30000 autonomous systems (AS), in which each AS is a collection of IP networks sharing a common routing strategy [4]. These networks are operated by thousands of Internet service providers (ISPs). Over the last ten years, many ISP companies have sprung into existence and now compete massively for customers. On the one hand, the ISPs compete with each other for customers and traffic, on the other, they have to cooperate and exchange traffic, otherwise the worldwide connectivity would be lost. Due to the dramatic increase in competition, it has become increasingly important for ISPs to run an efficient and cost-effective business.

Designing an internetwork can be a challenging task. An internetwork that consists of only 50 meshed routing nodes can pose complex problems that lead to unpredictable results. Attempting to optimize internetworks that feature thousands of nodes can pose even more complex problems. Despite improvements in equipment performance and media capabilities, internetwork design is becoming more difficult. The trend is toward increasingly complex environments involving multiple media, multiple protocols, and interconnection to networks outside any single organization's dominion of control. Carefully designing internetworks can reduce the hardships associated with growth as a networking environment evolves.

With the recent increase in demand for network services, a serious problem of how to route packets in these networks has become vital so as to guarantee a certain level of Quality of Service (QoS). To alleviate the burden on existing networks, ISPs must consider either of the following approaches [5].

1. Expand existing networks, or
2. Optimize existing networks through reconfigurations to maximize the usage of currently installed equipment.

To decrease the complexity, the network is divided into smaller domains, called autonomous systems. An autonomous system (AS) consists of a collection of routers and links managed by a single institution, such as a company, university, or Internet Service Provider as shown in Figure 1.1.

The efficient operation of the network depends on the configuration of the indi-

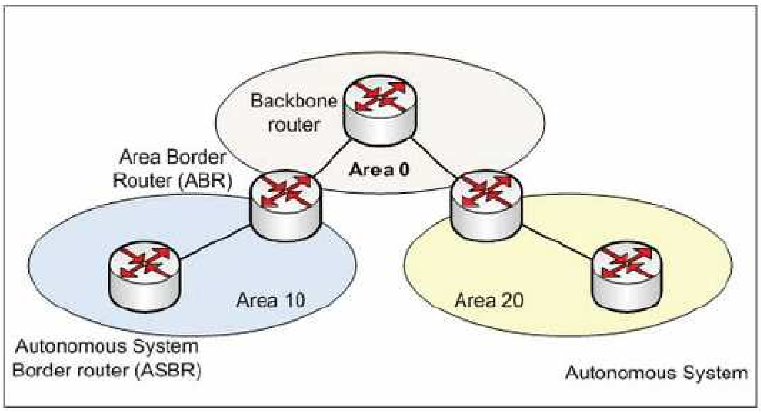


Figure 1.1: Autonomous Systems

vidual routers. Configuration controls the selection of a wide range of parameters that affect the allocation of resources (e.g., link bandwidth and buffers), the operation of the routing protocols (e.g., BGP policies and OSPF weights), and access to the network (e.g., packet filters). Configuring a large backbone network is extremely difficult for a variety of reasons:

1. Consistency of neighboring routers,
2. Complex configuration options,
3. Rapid changes to the network, and
4. Limited Configuration tools.

1.2 Traffic Engineering

Traffic Engineering(TE) is concerned with performance optimization of operational networks. TE plays a critical role in determining the performance and reliability of a network. A major challenge in TE is how to cope with dynamic and unpredictable changes in traffic demand. TE has become indispensable tool used by many ASes to select routes which effectively utilize their network resources. This is particularly important given the high cost of network's assets and the highly competitive nature of the ISP market [6, 7, 8].

Traffic characteristics are a major factor affecting the design of traffic engineering algorithms. Unfortunately, for many ASes, although their traffic demand can be relatively stable most of the time, there exist time periods during which traffic can be highly dynamic, containing unpredictable traffic spikes that ramp up extremely quickly, leaving no time for a traffic engineering algorithm to re-compute or adjust. Many factors contribute to the highly unpredictable nature of Internet traffic: outbreaks of worms/viruses, outages or routing changes of major ISPs, the occurrence of natural disasters, denial-of-service attacks, and flash-crowd effects due to major news events. For many cases, traffic spikes occur exactly when the networking service is most valuable. In addition, with more bursty UDP-based multimedia traffic, more dynamic traffic such as that from overlay networks [9], and more networks adopting traffic engineering, variability in traffic could increase further.

The key performance objectives associated with TE can be classified as being:

1. Traffic Oriented, or
2. Resource Oriented.

Traffic Oriented performance objectives include aspects that enhance the QoS of traffic streams whereas Resource Oriented performance objectives include aspects pertaining to the optimization of resource utilization. It is generally desirable to ensure that the subsets of network resources do not become overutilized and congested while other subsets along alternate feasible paths remain underutilized. In contemporary networks as bandwidth is a crucial resource, an important function of TE is to efficiently manage bandwidth resources.

1.2.1 Related Work in the Field of TE

Despite the importance of handling traffic spikes, most of the proposed traffic engineering algorithms belong to a type of algorithms which are called prediction-based TE. These algorithms optimize their routing without preparing for unpredictable traffic spikes. An advantage of this type of algorithms is their potential performance gain. When the network traffic is relatively stable and the real traffic is similar to the samples based on which the routing is computed, these algorithms can achieve near-optimal performance. However, since these algorithms optimize routing specifically for these samples, when the real traffic deviates substantially from the samples (e.g., during the presence of traffic spikes), the computed routing may perform poorly. An example of prediction-based TE is “Online Adaptation”.

Besides rapid traffic fluctuations, interdomain routing poses another set of challenges to traffic engineering. First, interdomain routing introduces point-to-multipoint demand, that is, there can be multiple equally-good egress points for some external destinations in the BGP decision process [10]. Thus, it is up to the intradomain routing determined by traffic engineering to break the tie. Since egress links may become the bottlenecks of the network [11], this tie-breaking can affect the congestion of the network. Second, although interdomain routes for most traffic volumes can be stable [12, 13], there are BGP routing changes that can cause significant shifts of traffic [14]. In particular, with the dynamic nature of the global Internet, the available interdomain routes of an AS can fluctuate as its peers announce and withdraw interdomain routes, or even reset their enhanced BGP (eBGP) sessions.

There are also recent studies on the interaction of intradomain traffic engineering with interdomain routes and traffic. Examples include evaluation (e.g., [14, 15, 16, 17, 18]) and design (e.g., [11, 19, 20, 21]). Recently, researchers observed that intradomain traffic engineering within an AS can cause substantial traffic changes outside the AS (e.g., [14, 22, 23]). For example, Agarwal et al. report in [22] that for an operational tier-1 ISP, intradomain traffic engineering can cause up to 25% of its traffic to a neighboring AS to shift the exit point. Such traffic changes could trigger routing changes at the neighboring AS, and result in network instability. Thus, the intradomain routing determined by traffic engineering should be robust against such interdomain route changes.

Intradomain traffic engineering has received significant attention in the research community. A set of algorithms were used in traffic engineering

1. for predicted traffic demands, and
2. oblivious routing.

The algorithms in the first category share the following features: they maintain a history of observed traffic demand matrices, and they optimize for the representative traffic demand matrices extracted from the observed traffic during a certain history window. For example, [22] use a traffic matrix in a one-hour window during daily peaks as the representative demand. [24, 25] consider multiple representative traffic matrices and find an optimal set of routes to minimize expected or worst-case cost for these representative matrices. It is to be noted that in their approach, the worst case is only among the samples, not all possible traffic matrices. In [26], Zhang and Ge try to identify critical matrices from past history, and then conduct traffic engineering based on these matrices. It might be possible to extend prediction-based optimization using robust optimization (e.g., [27]), but it will be challenging to estimate the variation set of parameters. MATE [28] and TeXCP [29] conduct online traffic engineering and react to instantaneous traffic demands. An advantage of these dynamic algorithms is that if they can converge quickly, they do not need to collect many samples or make prediction. However, when there are significant and fast traffic changes, these algorithms can experience a large transient penalty. The second category of algorithms is oblivious routing, a way to deal with unpredictable traffic spikes (e.g., [30], [31]).

In oblivious routing, routes are computed to optimize the worst-case performance over all traffic demands. Therefore the computed routes are prepared for dynamic changes in traffic demands, i.e., a routing that is independent of the traffic matrix is computed, and thus has the potential to handle traffic spikes well. In their pioneering work [32], Applegate and Cohen proposed an efficient algorithm to compute the worstcase oblivious routing for real networks. They also extend oblivious routing to compute failure scenarios [30]. They found that the oblivious ratio is typically around a factor of 2. A penalty as high as 100% may be acceptable when traffic demands are completely unpredictable, but it is a high cost to pay under predictable demands. This happens because, as the networks scale up, the optimal oblivious ratio of arbitrary symmetric networks can grow logarithmically. A potential drawback of oblivious routing, however, is its sub-optimal performance for normal traffic, which may account for the vast majority of time periods. In other words, oblivious routing takes a pessimistic point of view and may not be appropriate in relatively stable periods or stable networks.

1.2.2 Congestion Related Issues in Networks

One of the main objectives of Traffic Engineering (TE) is to avoid congestion by controlling and optimizing the routing function. While this approach is simple, highly distributed and scalable, these protocols do not consider network utilization and do not always make good use of network resources. The traffic is routed on the shortest path through the network even if the shortest path is overloaded and there exists alternate paths.

Minimizing congestion is a primary Traffic and Resource Oriented performance objective. Congestion typically manifest under two scenarios:

1. When network resources are insufficient or inadequate to accommodate the offered load, or
2. When traffic streams are inefficiently mapped onto available resources, causing subsets of network resources to be over-utilized while others remain under-utilized.

The first type of congestion problem can be addressed by either:

- Expansion of capacity,
- Application of classical congestion control techniques, or
- Both of the above.

Classical congestion control techniques aim to regulate the demand so that the traffic fits into available resources. Classical techniques for congestion control include: *Rate limiting, Window flow control, Router queue management, Scheduled based control* and others. The second type of congestion problems, namely those resulting from inefficient resource allocation, can usually be addressed through TE. In general congestion resulting from inefficient resource allocation can be reduced by adopting *load balancing policies.* The objective of such strategies is to minimize the maximum congestion or alternatively to minimize the maximum resource utilization, through efficient resource allocation. When congestion is minimized through efficient resource allocation, packet loss and transit delay decrease, while aggregate throughput increases. Thereby, the perception of network service quality experienced by end users become significantly enhanced. In this thesis, we worked on the OSPFWS problem which falls in the second type of congestion problems. In [33], it was shown that this is an NP-hard problem. In [5, 34], it was shown that optimizing these link weights leads to the efficient network utilization which is the main goal of TE.

1.2.3 Challenges for TE

Ambient networks is a network integration solution to the modern day problems of switching from one network to the other in order to keep in contact with the outside world. The main challenge for TE in Ambient Networks is to cope with the dynamics of both topology and traffic demands. The TE problem can best be modeled as a multi-commodity flow optimization (MCF) problem. This type of optimization techniques take as input the global information about the network state (i.e., traffic demands and link capacities) and calculate the global optimized solution.

MCF optimization as well as heuristic methods for setting optimal weights in OSPF are both typical examples of centralized schemes that use global information in the form of topology and traffic matrix and produce global optimal routing or at least results that are good, i.e., with lesser maximum utilization and fewer number of congested links for the network as a whole.

1.3 Thesis Organization

This thesis is dedicated to OSPFWS problem and the stochastic approach adopted in assigning weights to get to a robust solution. The importance of the contribution of TE and the challenges the TE needs to cope with for a successful internetworking is described in this introduction chapter.

The rest of the thesis is organized as follows: The second chapter *Routing Protocols* concisely describes the routing basics and its classification, and the interfaces and links used in the ISP backbone network architecture. The third chapter *Combinatorial Optimization using Iterative Heuristics* takes a close look at the combinatorial optimization methods. It introduces and discusses in details, the core of this thesis, i.e., the Stochastic Evolution algorithm. The fourth chapter *Reliability and Robust Optimization for OSPFWS* addresses the reliability issues in case of single link failures. The fifth chapter *Problem Description* provides a complete discussion of the problem formulation, terminology used, assumptions made, and in detail description of Dijkstra's algorithm and related work. The sixth chapter *Results and Discussion* describes the results followed by the comparison with other heuristics. The thesis concluding chapter *Conclusion and Future Work* collates the essence of knowledge and learning made through this work and the anticipations for the future work. The appendix incorporates some detailed readings that have been touched upon in the main thesis but were not covered in the interest of information required by a more general audience.

Chapter 2

Routing Protocols

2.1 Introduction

The Internet uses the terms *gateway, bridge, or router* to describe the machine that performs relaying functions between networks. IP is a simple networking protocol. It is an example of connectionless service. It permits the exchange of traffic between two host computers without any prior setup. IP relies on a module called the Internet Control Message Protocol (ICMP) to:

- report errors in the processing of a datagram, and
- provide for some administrative and status messages.

IP is not a route discovery protocol but its a forwarding protocol; it makes use of the routing table that a route discovery protocol creates.

Forwarding packets to their ultimate destinations depends on the complex interaction of various routing protocols. Static routes provide a simple way for a router to associate destination prefixes with particular outgoing interfaces. Static routes enable an ISP to direct traffic to its customers without requiring the customers to participate in a routing protocol with the ISP. The customer's network can direct all outgoing traffic to the access link connecting to the ISP, and the ISP can configure static routes to direct inbound traffic to the access link. If a customer connects to the ISP in multiple locations, the ISP may have multiple static routes to associate the same prefix with multiple access interfaces at one or more routers. The ISP must ensure that other routers in the backbone and the rest of the Internet can reach the customer's destination prefix. The AS also learns about destination prefixes via dynamic routing protocols, such as the Border Gateway Protocol (BGP).

2.1.1 Routing Basics

Routing is the method by which a host or router decides where to send the datagram. The goal of a routing protocol is to supply the information needed to do routing. Routers use the following four primary routing mechanisms to create and modify the routing tables:

Direct Connection : If the link is up, then any network connection to which the router is directly connected is automatically added to the routing table.

Static Routing : Manual entries can be configured on routers to instruct the router to use a given route to get to a particular destination.

Dynamic Routing : Router messages are announced and received. These update messages are used to create routes in the routing table. The routing algorithm associated with a particular routing protocol determines the optimal path to a particular destination and updates the route table. It can automatically adapt to changes in the network.

Default Routing : A manually entered route is used as a last resort to reach a destination when the route is not known by any other routing mechanism.

It is to be noted that when multiple routes are available from multiple sources in the routing table, then static routes typically always take precedence over any other routes.

2.1.2 Routing Protocol Classification

Routing protocols can be classified into the following:

1. An Interior Gateway Protocol (IGP), and
2. An Exterior Gateway Protocol (EGP) [35].

The four most common routing protocols used as IGPs are:

1. Routing Information Protocol (RIP),

2. Enhanced Interior Gateway Routing Protocol (EIGRP),
3. Open Shortest Path First (OSPF)[1], and
4. Intermediate System - Intermediate System (IS-IS) [36, 37].

A major concern in routing protocol security is to avoid false routing update packets that falsely modify routing tables. Often, this is due more to misconfiguration rather than malicious intent. Two basic approaches for protecting the routing table integrity are:

1. Use only static routes - used for very small networks.
2. Authentic routing updates.

As discussed in the previous paragraphs, routing is a fundamental engineering task on the Internet. It consists of finding a path from a source to a destination host. Routing is complex in large networks because of the many potential intermediate destinations a packet might traverse before reaching its destination [36].

2.2 ISP backbone network

The Internet consists of thousands of autonomous systems (ASes) that interact to coordinate the delivery of IP traffic. The routers within the ISP backbone are interconnected using core links.

[1]Refer to Appendix B

2.2.1 Network Architecture

An ISP has complete control over the configuration and operation of a core link, by virtue of controlling the adjacent routers. The edge links include access links that connect to the ISP's customers, such as business or university campuses, small regional providers, and local services like modem banks for dial-up users or a Web-hosting complex. The ISP may also have peering links that connect to other large providers via dedicated connections or public Internet exchange points. Configuration of access and peering links depends on interaction with the customers and the peers, respectively. In practice, an ISP may have multiple connections to the same customer or peer for load balancing and fault tolerance.

2.2.2 Interfaces and Links

An interface receives incoming packets, and queues and transmits outgoing packets. Each interface has a name that indicates its position in the router. Physically, an interface resides on a card that connects to a slot in the router's switching fabric. A link consists of two or more interfaces with the same network address [38]. The IP addresses of the interfaces are assigned by the network operator as part of configuring the router. Interface IP addresses do not necessarily have any relationship to the loopback IP addresses of the incident routers.

2.2.3 Packet and Route Filters

In addition to forwarding IP packets, an interface may perform various filtering functions. An AS may employ packet filters to control the traffic entering or leaving the backbone via a particular interface. Access control lists (ACLs) identify which packets should be accepted or denied, based on fields in the IP headers such as source and destination addresses. Packet filtering helps in detecting spoofed source IP addresses and protects the customer from receiving traffic from unwanted sources [39]. An AS may also use route filters to balance the load across a collection of links to the same neighboring domain.

2.3 Summary

This chapter gives an introduction to the concept of IP, the various mechanisms used to create and modify the routing tables. The routing protocol classification is made, of which, OSPF, an example of IGP, forms the core of this thesis work. The ISP backbone network, its architecture, interfaces and links are briefly discussed here and a detailed description is presented in Appendix B for further interest of the more focussed audience.

Chapter 3

Combinatorial Optimization using Iterative Heuristics

Evolutionary algorithms are derived from observations in nature where living beings are improved by evolutionary mechanisms. Before the actual problem is discussed, this chapter introduces the concept of Combinatorial Optimization and the heuristic used to solve the OSPFWS Optimization problem.

3.1 Combinatorial Optimization

Combinatorial Optimization constitutes one specific class of problems. The word Combinatorial is derived from the word combinatorics, which is a branch of Mathematics concerned with the study of arrangement and selection of discrete objects [1].

Combinatorial Optimization is not concerned with whether a particular arrangement or ordering exists but rather, it is concerned with the determination of an optimal arrangement or order [40].

A solution (optimal or not) to a combinatorial optimization problem usually requires that one comes up with a suitable algorithm, which when applied to an instance of the problem produces the desired solution. Combinatorial Optimization problems are encountered everywhere, in science, engineering, operation research, economics, etc.

3.1.1 Optimization Methods

Combinatorial Optimization problems can be categorized into two general categories:

1. Exact Algorithms, and
2. Approximation Algorithms.

Linear programming, Dynamic Programming, Branch and Bound, Backtracking, etc., are few prominent examples of Exact Algorithms. Christofields‘ algorithm, Algorithm 2, an H(n)-approximation algorithm, Algorithm 3, a 2-approximation algorithm, Algorithm 4, a 1/2-approximation algorithm, LP rounding, etc [41] are examples of Approximation algorithms. However, there are instances, where optimal enumerative techniques cannot be used. Such problems are considered to be NP-hard. In such cases, it is easier to resort to heuristic approach. Examples of heuristic algorithms are constructive greedy method, local search, and the modern general iterative algorithms such as Simulated Annealing, Genetic Algorithms, Tabu Search, Simulated Evolution and Stochastic Evolution. A common feature of all the aforementioned search algorithms (whether Exact or Approximate) is that they constitute general solution methods for combinatorial optimization.

3.2 Iterative Heuristics

Local Search heuristic is considered to be one of the oldest known iterative heuristic upon which the other modern heuristics such as Simulated Annealing, Genetic Algorithm, Simulated Evolution, Stochastic Evolution and Tabu Search are built upon [1]. All these combinatorial optimization algorithms try to optimize a given objective function *Cost.* The cost function is also referred to as *Objective or Utility function.*

In a nutshell, combinatorial optimization algorithms can be classified as *Deterministic* or *Stochastic.* In the *Deterministic* approach, the extremum of the objective function is obtained by making deterministic decisions; whereas in the *Stochastic* approach, random decisions are made in the search of the extremum of the utility function. Hence, irrespective of the number of times the simulations are repeated, deterministic algorithms provide the same solution always whereas this is not the case with stochastic algorithms.

Another classification of heuristic algorithms leads to *Constructive* and *Iterative* algorithms. In a constructive heuristic, a seed component or several seeds are considered as the initial point. Gradually, other components are selected and added to the previous solutions. An important consideration made here is that when a component is selected, it is not moved during future steps of the procedure. The seed component is randomly selected in case of iterative heuristics , whereas in the case of constructive heuristics, the selection of seed component is choiceless. Hence, constructive algorithms are also known as *successive augmentation algorithms.* However, in the case of iterative heuristics, an initial solution is randomly generated and passed as an input. Random solutions are generated quickly, but the iterative algorithm may take a large number of iterations to converge to either a local or global optimum solution. On the other hand, a constructive heuristic takes up time, nevertheless the iterative improvement phase converges rapidly if started off with a constructive solution. Typically constructive algorithms are deterministic while iterative algorithms may be deterministic or stochastic.

In terms of convergence, constructive algorithms have an upper hand when compared to iterative algorithms taking time into consideration. Constructive algorithms are greedy in nature. Once they come to a local minima, the convergence is said to be achieved, unlike iterative algorithms which take longer time. Hence, iterative algorithms require several iterations to attempt various modifications to the solutions to bring it to a feasible state. For practical problems such as assigning weights to a large topology, this becomes unmanageable. Care must be taken so that the iterative procedure is tuned to quickly converge to a solution satisfying all design constraints if the search has to be speeded up.

3.3 Stochastic Evolution (StocE) Algorithm

StocE adopts the following generic model [1]:

Given a finite set M of movable elements and a finite set L of locations, a state is defined as a function $S : M \rightarrow L$ satisfying certain constraints.

Optimized assignment of weights to the network considered in this thesis can be formulated according to this model.

The following summarizes the StocE algorithm described in more detail in [1]. The inputs to StocE algorithm are:

1. An initial state S_0,
2. An initial value of the control parameter p_0, and
3. A parameter R used in the stopping criterion.

Consider a state space Ω. A suitable solution $S(m)$ for each movable element $m \in M$ is found by StocE algorithm. This process continues until a lower cost of the whole state S is obtained. A general outline of the StocE algorithm is given in Figure 3.1 [1].

```
**Algorithm StocE**($S_0$, $p_0$, $R$);
Begin
    $BestS = S = S_0$;
    $BestCost = CurCost = Cost(S)$;
    $p = p_0$;
    $\rho = 0$;
    Repeat
        $PrevCost = CurCost$;
        $S = PERTURB(S, p)$; /* perform a search in the neighborhood of S */
        $CurCost = Cost(S)$;
        $UPDATE(p, PrevCost, CurCost)$; /* update $p$ if needed */
        If ($CurCost \leq BestCost$) Then
            $BestS = S$;
            $BestCost = CurCost$;
            $\rho = \rho - R$; /* Reward the search with $R$ more generations */
        Else
            $\rho = \rho + 1$;
        EndIf
    Until $\rho > R$
    Return ($BestS$);
End
```

Figure 3.1: The Stochastic Evolution algorithm [1].

Initially, the solution is considered to be the best solution as well as the current solution and the corresponding cost as best cost as well as the current cost, respectively. The main loop runs till the value of ρ exceeds the stopping parameter 'R'. The StocE algorithm retains the state of the lowest cost among those produced by the function 'Perturb'. Then, the **PERTURB** function (see Figure 3.2) is invoked to make a compound move from the current state S. Each time the algorithm visits a state which has a lower cost than the best cost so far, the StocE decrements the counter by 'R', thereby rewarding itself by increasing the number of iterations.

```
FUNCTION PERTURB($S, p$);
Begin
    ForEach ($m \in M$) Do   /* according to some apriori ordering */
        $S' = MOVE(S, m)$;
        $Gain(m) = Cost(S) - Cost(S')$;
        If ($Gain(m) > RANDINT(-p, 0)$) Then
                $S = S'$
        EndIf
    EndFor;
    $S$ =MAKE_STATE($S$); /* make sure $S$ satisfies constraints */
    Return ($S$)
End
```

Figure 3.2: The PERTURB function [1].

The parameter R represents the expected number of iterations the StocE algorithm needs until an improvement in the cost with respect to the best solution seen so far takes place, that is, until *CurCost*$\leq$ *BestCost*. If R is too small, the algorithm will not have enough time to improve the initial solution, and if R is too large, the algorithm may waste too much time during the later generations. Experimental studies indicate that a value of R between 10 and 20 gives good results [42].

PERTURB scans the set of movable elements M according to some apriori ordering and attempts to move every $m \in M$ to a new location $l \in L$. For each trial move, a new state S' is generated, which is *a unique* function $S' : M \rightarrow L$ such that $S'(m) \neq S(m)$ for some movable object $m \in M$. The move associated with 'm' could itself be a simple move or a compound move. In our case, it is a compound move for the OSPFWS problem. This movement changes the whole state of the solution resulting in a new cost calculation again. To evaluate the move, the gain function $Gain(m) = Cost(S) - Cost(S')$ is calculated. The function stochastically decides whether or not to accept the move associated with the element being scanned with the help of a non-negative control parameter 'p'. If the calculated gain, Gain(m) is greater than some randomly generated integer number in the range $[-p, 0]$, the move is accepted and S' replaces S as the current state. Since the random number is ≤ 0, moves with positive gains are always accepted. The algorithm then goes to scan the next element in 'M'. After scanning all the movable elements $m \in M$, the **MAKE_STATE** routine makes sure that the final state satisfies the state constraints. If the state constraints are not satisfied then **MAKE_STATE** *reverses* the fewest number of latest moves until the state constraints are satisfied. This procedure is required when perturbation moves that violate the state constraints are accepted.

The new state generated by **PERTURB** is returned to the main procedure as the current state, and its cost is assigned to the variable *CurCost*. Then the routine **UPDATE** (Figure 3.3) is invoked to compare the previous cost (*PrevCost*) to the current cost (*CurCost*).

The **UPDATE** procedure is mainly responsible for updating the value of the control parameter 'p'. If $PrevCost = CurCost$, there is a good chance that the algorithm has reached a local minimum and therefore, p is increased by p_{incr} to tolerate larger uphill moves, thus giving the search the possibility of escaping from

```
PROCEDURE UPDATE($p$, $PrevCost$, $CurCost$);
Begin
    If ($PrevCost=CurCost$) Then /* possibility of a local minimum */
        $p = p + p_{\text{incr}}$; /* increment $p$ to allow larger uphill moves */
    Else
        $p = p_0$; /* re-initialize $p$ */
    EndIf;
End
```

Figure 3.3: The UPDATE procedure [1].

local minima. Otherwise, p is reset to its initial value p_0.

At the end of the loop, the cost of the *current state* S is compared with the cost of the *best state BestS*. If S has a lower cost, then the algorithm keeps S as the best solution (*BestS*) and decrements ρ by R, thereby rewarding itself by increasing the number of iterations (allowing the search to live R generations more). This allows a more detailed investigation of the neighborhood of the newly found best solution. If S, however, has a higher cost, ρ is incremented, which is an indication of no improvements.

3.3.1 Convergence Aspects

The run-up in obtaining the optimized state in StocE can be considered to be a combination of two phases [1]. The first phase consists of a *perturbation* step also called *generalization* and a *makeup* step referred to as *specialization*. Relocating some of the movable elements is done in the perturbation step which further creates an intermediate step, a step that may violate some of the problem constraints. To get to an optimized state from the intermediate state, the make-up step is invoked which satisfies all the problem constraints.

As discussed in the section 3.1, the convergence features of StocE are dependent on both p and R. If p and R are small, the algorithm may never converge. The reason is that for small p some of the transitions become impossible (have a zero probability), thus pruning portions of the search space. On the other hand, the algorithm lacks enough time to get to an optimal state if the value of R is too small. Hence judicious decision has to be taken while considering the parameters discussed.

3.3.2 Adaptations of StocE to various applications

Apart from solving the OSPFWS problem, StocE algorithm can be used to solve a wide variety of applications. The following constraints are to be followed by different applications to get adapted to StocE.

1. The solution space has to be defined,
2. A suitable state representation be adopted,
3. The notion of cost and perturbation have to be appropriately identified,
4. An initial value of the control parameter p and a method to update it must be chosen, and

5. A value for the stopping criterion be selected.

As widely known, the benefits of implementing an algorithm can be reaped to its maximum on comparing it with other widely known algorithms applicable to the problem targeted. However, this has been clearly discussed in *Results and Discussion* chapter.

3.4 Summary

This chapter discusses one of the core issues on which the entire implementation process is dependent. An introduction to combinatorial optimization is illustrated and an overview of the iterative heuristics is discussed. A generic outline of StocE algorithm is portrayed with a clear explanation of its core modules such as *Perturb*, *Update* and *Make-State*. An implementation of StocE and its comparison to other widely known algorithms such as Simulated Annealing, Simulated Evolution, Genetic Algorithm, etc., is illustrated explicitly in *Results and Discussion* chapter. Another important factor, the convergence aspect, is discussed in this chapter.

Chapter 4

Reliability and Robust Optimization for OSPFWS

4.1 Introduction to Reliability

Reliability, i.e., availability and survivability of services, is one of the important issues in transport networks of all kinds. The network should be designed, configured and reconfigured to satisfy this demand effectively. The impact of component (link or node) failure on the performance of the network as a whole depends in part on the network topology and in part on the routing strategy. Traditionally, two approaches have been taken to evaluate the reliability of a stochastic network [43]:

- Connectedness: From the probabilities of node or link failure, a measure of the probability of network connectedness.

- Capacity: From the probabilities of node or link failure, a measure of the probability that network capacity is sufficient to meet demands.

These two methods are computationally intensive for large networks. However, links with a large potential impact on the network performance have low probability of failing, in which case the network is in practice robust.

Each router in the network using shortest path forwarding protocols has a knowledge of the topology and the associated weights to determine the shortest paths to different destinations. IP networks frequently experience IP link failures. These could be in the order of several a day and most of them are transient, i.e., in the order of tens of minutes. However when there is a failure in the network (link or node, failure), these protocols take some time to detect the failure and reestablish a consistent view of the new topology. During this transient period, the data traffic forwarded towards this failed device will be dropped. Additionally, artificial congestion in the network might result due to emergence of routing loops. Since, lot of packets are queued during transient periods, a possible congestion of the network is clearly visible. When new routes are established, there could be further delay. The network must be robust to such perturbations. Manual intervention or reconfiguration is not a viable solution.

All communication networks are designed for certain demands and requirements. In the course of time, traffic demands typically increase and the networks are often not as satisfying as at the beginning. Therefore, the augmentation of networks by additional links plays an important role in network design. In addition to the increase of bandwidth, an increase of robustness and survivability is often needed. A network can be made robust against failures in connections between two sites or against site failures. The costs of such augmentations should usually be as small as possible. A carrier network often uses a lower layer transport (or data link layer failure) detection and restoration techniques, so that the service level assurances (SLA) provided by the carrier network to the customers is not excessively impacted. Depending on just the routing layer for recovery from failures has been typically considered unacceptable, because it takes too long to recover from failures. Incorporating protection against failures at the transport layer is expensive as it requires significant redundant capacity. This is a Cisco approach. Generally, all the devices incorporate multihoming, a technique to increase the reliability of the Internet connection for an IP network. Minimizing the failure recovery time has the benefit of a reduced need to depend on transport/data link layer recovery and the possibility that a more complete, network layer failure recovery mechanism could be put in place.

4.2 Failure Detection and Recovery Mechanism in OSPF

In OSPF, two adjacent routers in the same area periodically exchange *Hello* messages to maintain the link adjacency. If a router doesn't receive a *Hello* message from its neighbor within a *RouterDeadInterval* (typically 40 seconds or 4 *HelloIntervals*), it assumes the link between itself and the neighbor to be down and generates a new *Router* LSA (Link State Advertisement) to reflect the changed topology. All such LSAs generated by the routers affected by the failure, are flooded throughout the network and cause the routers in the network to redo the *shortest path first (SPF)* calculation and update the next hop information in the forwarding table. Thus, the time required to recover from the failure consists of [44]:

- The failure detection time.
- The LSA flooding time, and
- The time to complete the new SPF calculations and update the forwarding tables.

The failure detection can take place between 10 to 40 seconds.

Table 4.1 [45] lists different standard and vendor introduced delays that affect the OSPF operation in networks of popular commercial routers.

Table 4.1: Standard and Vendor introduced delays

Standard Configurable Delays	
RxmtInterval	The time delay before an un-acked LSA is retransmitted. Usually 5 seconds.
Hello Interval	The time delay between successive Hello packets. Usually 10 seconds.
Router Dead Interval	The time delay since the last Hello before a neighbor is declared to be down. Usually 4 times the HelloInterval.
Vendor Introduced Configuration Delays	
Pacing Delay	The minimum delay enforced between two successive Link State Update packets sent down an interface. Observed to be 33 ms. Not always configurable.
spfDelay	The delay between the shortest path calculation and the first topology change that triggered the calculation. Used to avoid frequent shortest path calculations. usually 5 seconds.
spfHoldTime	The minimum delay between successive shortest path calculations. Usually 10 seconds.
Standard Fixed Delays	

LSrefreshTime	The maximum time interval before an LSA needs to be re-flooded. Set to 30 minutes.
MinLSInterval	The minimum time interval before an LSA can be reflooded. Set to 5 seconds.
MinLSArrival	The minimum time interval that should elapse before a new instance of an LSA can be accepted. Set to 1 second.
Router Specific Delays	
Route install delay	The delay between shortest path calculation and update of forwarding table. Observed to be 0.2 seconds.
LSA generation delay	The delay before the generation of LSA after all the conditions for the generation of LSA are met. Observed to be around 0.5 seconds.
LSA processing delay	The time required to process an LSA including the time required to process the Link State Update packet before forwarding the LSA to the OSPF process. Observed to be less than 1 ms.
SPF calculation delay	The time required to do the shortest path calculation. Observed to be $0.00000247x^2 + 0.000978$ seconds on Cisco 3600 series routers; x being the number of nodes in the topology.

However, there are few additional steps involved in the service restoration process that depend on the router architecture and the design of the router control plane. These are:

1. Detection of an interface being up or down.
2. Initial delay before the protocol stack is notified of the link status change. An example of this is *carrier-delay* timer value used in Cisco routers.
3. Initial wait to generate a new LSP that informs other routers about the event, determined by another timer, *lsp-gen interval.*
4. LSP flooding across the network.
5. Delay between the arrival of LSP and the start of SPF computation.
6. SPF computation and update of Routing Information Base (RIB)
7. Pushing new routing entries to the Forwarding Information Base (FIB) at the linecards.

4.3 Optimization of OSPF weights in a Robust Fashion

Weight settings for a fixed network (i.e., with a fixed topology and no failure) proposed by Fortz and Thorup [46] have shown that 50% to 110% more demands can be sustained when compared to Cisco's default inverse-capacity weights, and get within a few percent of the best possible solution with general routing including MPLS. In one of the results shown in [47] regarding robustness of OSPF routing with optimized weights with respect to link failures, weights are optimized for the no-failure case and a method to recover the performance with fewer weight changes is provided. However, as described in [5], operators do not like to change weights, so it would be preferable to take into account the failure scenarios when optimizing the weights, in order to obtain a robust weight setting. Evaluating all link and node failures for each solution is intractable for the problem considered. Similar to the link failure scenario used in our case, a problem is considered in [48] where only the maximum utilization in the normal state and for the worst link failure are considered. In this case, metrics have been optimized through IGP engineering to reduce the maximum link utilization across the network in both states. Therefore, it makes the network more resilient to link failures and sudden traffic surges.

4.3.1 Network Upgrade

One of the critical challenges in the area of network management is the problem of *graceful network upgrade*, where ISPs need to add new nodes and links into their operational network in a graceful manner so that the perceived network performance from the perspective of existing customers does not deteriorate. From ISP point of view, planning a successful network upgrade involves three main steps:

1. Identifying a set of potential locations where new nodes can be added,

2. Determining a subset of all the identified locations where nodes should be added (along with the links) such that it results in maximum revenue for the ISP given certain budget and performance constraints, and

3. Identifying an ideal sequence for adding nodes and links into the network such that the upgrade process has the minimum impact on the existing customers.

An obvious approach for an ISP is to add nodes at all possible locations and connect them into full mesh to give the best performance, but such a strategy is usually infeasible due to budget constraints. One of the major reasons of performance deterioration in ISP networks is due to link failures that occur frequently due to several reasons like fiber cuts, software bugs, and hardware errors.

4.3.2 Single Link Failures

Failures can be attributed either due to a link or a node. Considering the possibility of a link failure, a *state* of the network is defined as a subset S $\subseteq$ A of arcs containing the arcs that are operational. Here, we consider only *normal state* A and *single link failure states* A_a for each link a $\in$ A. It is assumed that once the operator has fixed the set of OSPF weights, he/she does not want to change it whatever the state of the network is. The possibility of getting a better routing in case of failures by allowing a few weight changes is discussed in [47].

The cost function has been designed in such a way that it tries to keep the flow on each link below the capacity of that link. It is assumed that at most there is one link failure at a time. This is because individual link failures account for nearly 70% of all unplanned failures [49] and rerouting around multiple failures requires more complex algorithms. Fast rerouting schemes are desirable to shorten the time to route around failures since, it is shown in [49] that about half of the unplanned failures last less than a minute in backbone networks.

4.3.3 Link Failure Evaluation

It has been reported in the literature [50] that the complexity of link failure evaluation increases ten fold. Scaling for larger networks is also skeptical. Assuming that the performance deterioration due to either link or node, failure is dominated by cost of failures in links, selecting a link becomes predominantly important. While selecting a link, as there are no critical links and the number of links interconnected with the nodes is huge, having a track of link failure was difficult. So, we added a link and later removed it. In this way, it was was made sure of the link that fails.

4.4 Related Work in the Field of Link Failures

Reducing the failure detection time is one of the main components of the overall failure recovery time in OSPF based networks. The failure detection via the *Hello* protocol can be substantially speeded up by reducing the *HelloInterval.* However, a realistic approach has to be made regarding how small the *HelloInteval* can be to achieve faster detection and recovery from network failures while limiting the occurrence of false alarms. In [51], Alaettinoglu et al. proposed reducing the *HelloInterval* to millisecond range to achieve subsecond recovery from network failures but did not consider any side effects of *HelloInterval* reduction. In [52], Shaikh et.al. used Markov Chain based analysis of a simple network topology to obtain the expected times before high packet drop rates cause a healthy adjacency to be declared down and then back up again. However, this work did not study the network wide generation of false alarms caused by congestion as the *HelloInterval* is reduced. In [53], Basu and Riecke examined using sub-second *HelloIntervals* to achieve faster recovery from network failures. It reports 275 ms to be an optimal value for *HelloInterval* providing faster failure detection while not resulting in too many false alarms. However, this work did not consider the impact of different levels of network congestion and topology characteristics on the optimal *HelloInterval* value.

The possibility of false alarms being generated increases with the reduction in *RouterDeadInterval.* Delays introduced by vendors regarding SPF calculation (spfDelay and spfHoldTime) result in slowing down the failure recovery process. In [51], Alaettinoglu et al. proposed eliminating any restriction on SPF calculations arguing that the frequency of SPF calculations can be reduced by using modern algorithms such as [54, 55, 56] instead of Dijkstra's algorithm.

Mukul [45] proposed that with around 10% overload on the system, any *HelloInterval* value less than 10 seconds leads to unacceptable number of false alarms. The probability of a false alarm occuring in the network increases with the number of links in the network. It is difficult to prescribe a single *HelloInterval* that will perform optimally in all cases. The network operator should set the *HelloInterval* conservatively taking into account both the expected congestion levels as well as the number of links in the network topology.

Thorup [57] presented data structures to be kept in the routers for usage in case of single link failures. If a link fails, according to OSPF/IS-IS protocol, routing should be done along shortest paths, but this time in the network without the failed link. However, it may take seconds for the forwarding tables to get updated. First of all, the information about the failure has to be flooded to all the routers using link state advertisements (LSAs). Secondly, each router has to update its forwarding table so as to use its outgoing links on the new shortest paths. In the transition time, a period of *inconsistent forwarding* is obtained where a packet may be sent back and forth between routers, some of which still forward it towards the failed link whereas those updated, try to send it around the failed link. This work suggests a scheme to get a constant time recovery from a link failure in OSPF/IS-IS routers. However, when more than one link fails simultaneously, the same old system of recomputing the shortest paths with failed links has to be followed.

Narvaez et.al. proposed a local restoration algorithm for link state protocols in [58]. The algorithm requires routers on a restoration path to change the weights of links on the path to zero and recalculate their routing tables. The calculation of the routing tables and the update of forwarding tables increase the response time of the algorithm to link failures and increases the work load of the router.

Liu et.al. [59] proposed a fast rerouting extension for link state protocols. In this approach, when a link fails, the affected traffic is rerouted along a precomputed rerouting path. In case, rerouting cannot be done locally, the local router will signal a minimal number of upstream routers to set up the rerouting path for rerouting.

[48] used a Tabu search heuristic to determine link weights for evaluating link failure impact, though its computationally very intensive.

4.5 Importance of No-Weight Changes in OSPF-IS/IS Networks

Demand matrices and networks change as described in [60]. However, there are several reasons why one should avoid weight changes as much as possible. Firstly, even a single weight change is disruptive for a network. The weight change has to be flooded in the network. When the routers get to know about these changes, they start computing their shortest paths to update their routing tables, and it may take seconds before all routers agree on the new shortest paths. Meanwhile, packets may arrive out of order, degrading the performance of TCP. Obviously, the more weight changes that are flooded simultaneously, the more chaos is introduced in the network with different LSAs being sent back and forth between routers in the network. Secondly, with many weight changes, a human operator should oversee the configuration of weights. It is made sure that the resulting routing is well behaved, and a diverse set of requirements is satisfied. These requirements may be very specific to the network, and may never be explicitly formulated until the network operator sees a concrete problem. If less changes are made, the operator is able to endorse them easily. Limiting the range of weights is technically trivial.

It was found by experimenting on AT&T IP backbone in [47] that increasing a single weight from 1024 to 1025 reduced the maximum utilization by 8%. The maximum utilization is the utilization of the link with the highest utilization in the network. This suggests that the weight change was worth approximately 8.7% increase in link capacities. Checking the impact of a single weight change is relatively easy and executing it is much cheaper than buying and installing new links with higher capacity. Here, the problem of reestablishing performance after a local change such as link failure or hot spot is considered. The starting point is a set of weights optimized for a network with a given demand matrix.

Typically network operators use the maximum link utilization in the network as a measure of network performance. However, link utilization does not consider the network performance from user point of view. Hence, the optimization decision based solely on link utilization could result in performance degradation for some or all customers. Minimizing the maximum utilization as in [61] is a natural and intuitive objective for routing. There could be some links for which a particular concern about high utilization is made, but that can be viewed as links having a reduced capacity. In general, a single overloaded link does not take the complete network down.

By applying the technique of assigning as few as possible or literally no-weight changes, the classical combinatorial optimization problems can be applied to problems of true practical relevance. This technique is of much importance to problems where parameter changes are undesirable. From the programming perspective, the adaptation of using the same code to link failure case has the advantage that the original StocE code could be reused. A naive approach to the choice of topologies to use for link failure case is to choose the nodes with the highest demand but not directly connected, thus avoiding the need for sophisticated tools.

4.6 Reliability of the Network

Network reliability concerns the capability of the underlying network to provide connections to support a required network functionality. In small networks with relatively unreliable components, disconnection is a major concern - more important than a loss in performance. Frank and Frisch gave a clear exposition of the importance of connectedness [62]. Frisch, Chou and Kahn [62, 63, 64] provided a useful study of the ARPA network in its early days, reflecting the focus on connectivity. Recent research with reliability point of view revolves around the delivery of acceptable performance in the presence of failures.

Generally, when the load exceeds the capacity, the links get congested and the packets get dropped. The lost demand is the amount of flow lost when a given demand has to be sent on the link. Part of this flow is sent upto the capacity of the link and the remaining is lost. This lost demand gives the measure of the reliability of the network. As described earlier, the routing in OSPF networks is highly dependent on the weights assigned on the operational arcs given by a probability p_a. The lost demand was structured by Fortz [65] with the help of the following equation:

$$R(w) = \sum_{S \subseteq A} Pr(S) LD(S, w)$$

where R(w) gives the measure of reliability of the network with OSPF weights 'w' and LD(S,w) gives the lost demand in state 'S'. The probability Pr(S) of state 'S' is given by the following equation:

$$Pr(S) = (\prod_{a \in S} p_a)(\prod_{a \notin S} (1 - p_a))$$

An example is considered from [65] to show the actual lost demand that will heavily depend on the order in which demands are routed as explained later.

Consider the network depicted in the figure 4.1 with the corresponding arc capacities.

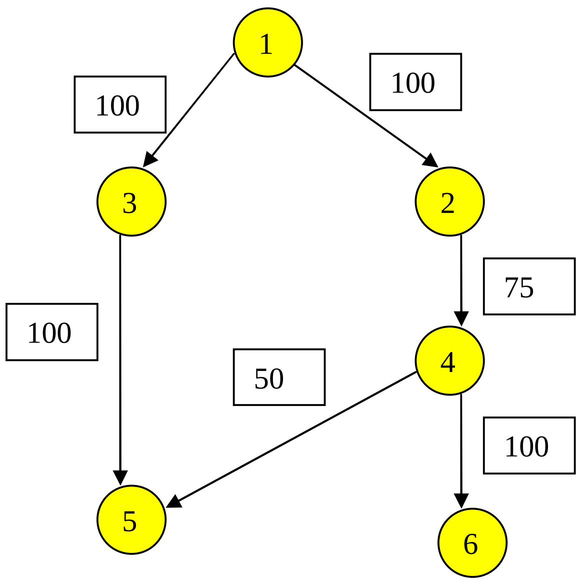


Figure 4.1: Network Example [2].

Suppose there are two demands to route: a demand of 100 units between nodes 1 and 5, and a demand of 50 units between nodes 2 and 6. The optimal routing is obviously to send 100 units on the path $1 \rightarrow 3 \rightarrow 5$ and 50 units on the path $2 \rightarrow 4 \rightarrow 6$, assuming weight equal to 1 on each arc.

In a circuit-switched network, a connection is established on these two paths and the appropriate capacity is reserved from the origin to the destination of these demands. Now, suppose link $1 \rightarrow 3$ fails. The operator can decide to use path $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$ as a backup path for the demand from $1 \rightarrow 5$. In this case, since all demands must cross arc $2 \rightarrow 4$, only 75 units of demand can be sent and the lost demand amounts to 75, while the operator has the choice to send 50 units for demand 1 → 5 and 25 units for demand 2 → 6, or 25 units for demand 1 → 5 and 50 units for demand 2 → 6, or any linear combination of these two possibilities, depending on the priorities he assigns to these demands.

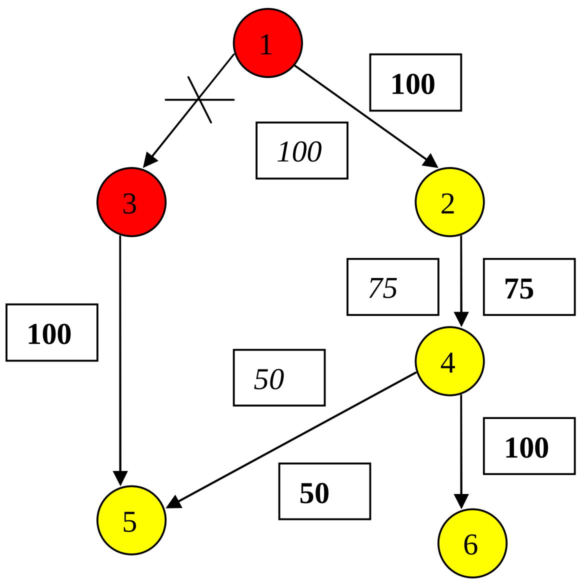
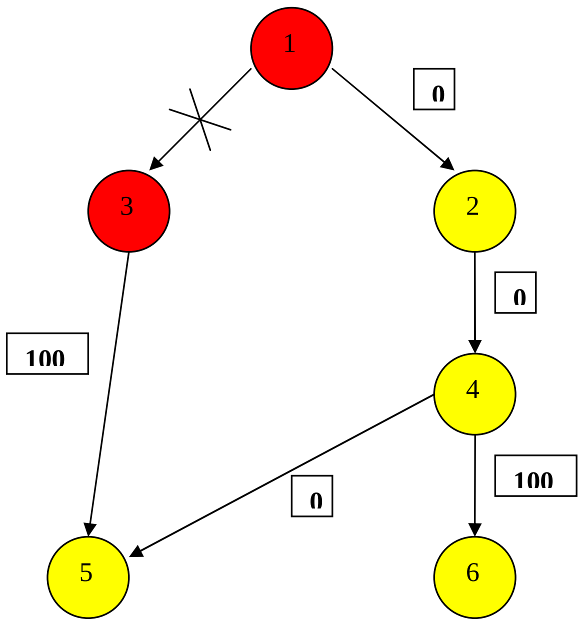


Figure 4.2: Demand 1 → 5 is routed first. 25 units are lost at node 4 [2].

The same network and demands are considered, but let the network be packet-switched and routing is performed using OSPF. If all links operate, the same routing as for a circuit-switched network can be obtained by assigning a weight equal to one on each arc. But the situation becomes quite different if link 1 → 3 fails. As OSPF operates in a best effort way, as much demand as possible is forwarded to the next node in the routing table, independently of capacities on the remainder of the path, while in a circuit-switched network, the capacity on the entire path is considered before sending the demand. As a consequence, the lost demand depends on the order in which demands are routed, as illustrated in the figure 4.2, where the residual capacities are denoted in bold and the flow sent on a link is denoted in italic.



Suppose the demand $1 \rightarrow 5$ is routed first (figure 4.2). As link $1 \rightarrow 3$ is unavailable, the shortest path becomes $1 \rightarrow 2 \rightarrow 4 \rightarrow 5$. From node $1, 100$ units are forwarded to node 2. As only 75 units of capacity are available on arc $2 \rightarrow 4$, only 75 units are forwarded to node 4 and 25 units are lost. Similarly, only 50 units are forwarded from 4 to 5 and an additional 25 units are lost. Then we try to route demand $2 \rightarrow 6$.

The routing table says flow has to be sent from $2 \rightarrow 4$, but no capacity is left on this link and all 50 units are lost. Therefore, the queued traffic in this case is

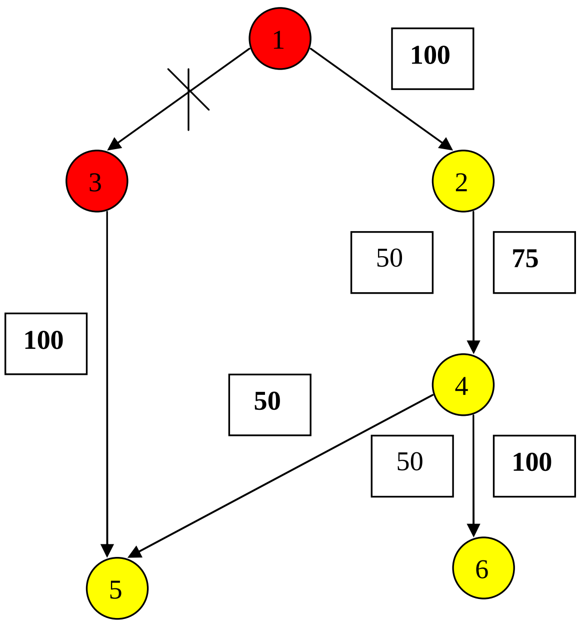
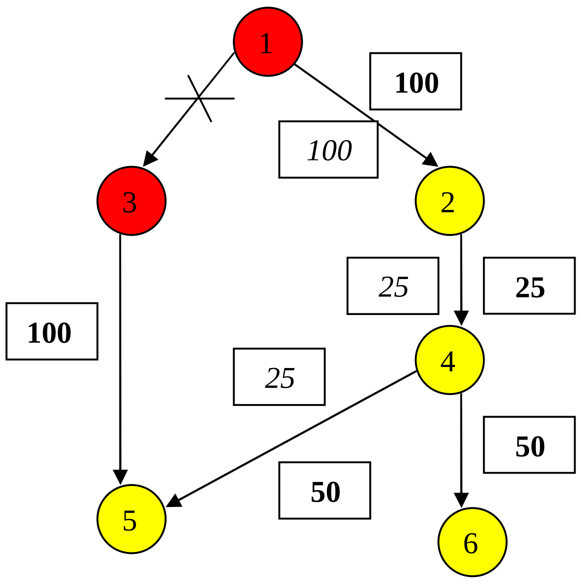


Figure 4.4: Demand 2 → 6 is routed first [2].

100. If demand 2 → 6 is routed first (Figure 4.4), it can be sent completely on path 2 → 4 → 6. Then we try to route demand 1 → 5 (Figure 4.5). 100 units are sent on link 1 → 2, then only 25 units can be forwarded to node 4 and then to node 5. In this case, the queued traffic is 75, the same as for a circuit-switched network.

It is clear from this example that the main drawback of OSPF routing is that some capacity can be used for flows that will not arrive to their destination, decreasing the available capacity for other demands. The actual lost demand will heavily depend on the order in which demands are routed. In the real environment, it will depend on the order of arrival of packets, and the buffer size in each router. However, a good compromise is to route small demands first. Indeed, the wasted capacity due to demands that are not routed completely will be smaller if small



demands are routed first, hence assuming demands are routed in increasing order of magnitude starting from the smallest demand. This is the procedure incorporated in many routing networks. By doing so, we are reducing the lost demand. Though, we are not doing it on a node by node basis, however, with this example, a sound approach is built in supporting this behavior of routing the smallest demands first.

4.7 Strategies Implemented

Since the topologies considered are connected networks, deleting a link can result in a disconnected graph. Since we are not aware which link could be crucial, hence the link failure case is approached in the following way:

1. Initially, a network topology is considered from those presented in the chapter 6

and a link is added. This approach makes sure that the new topology obtained represents a connected network.

2. The link that is added is generally the one with the highest demand in the given network. This can be further explained in the following manner:

    - Firstly, all the demands are checked from the input file to identify the link with the highest demand. This link is then added if it does not exist.
    - If this link already exists in the given network, then the link with the next highest demand is chosen and added. This process continues till a link with the next highest demand is not found. This link is then added.

Two strategies have been proposed to deal with link failure cases leading to a better understanding of these types of problems.

4.7.1 Strategy Original Heuristic (StratOH)

This strategy is approached in a comparative fashion. This is the original heuristic used to compare strategy 2. Initially the topology with an added link is considered for optimization, namely (N+1)OH. Weights are assigned to the (N+1) topology, shortest paths are obtained and the cost for this topology is computed, i.e., cost 2. Here, the cost obtained through the topology, (N+1)OH, is considered to be the best cost and this is given back as input for further iterations. This step shows that irrespective of the link failure, the weights which have been assigned are the best.

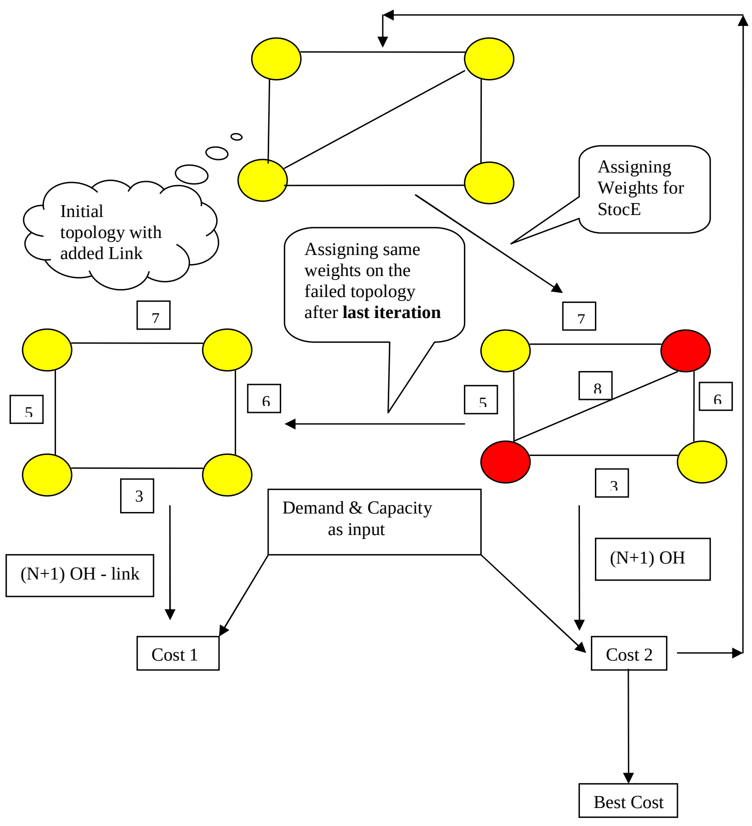

Initial topology with added Link
Assigning Weights for StocE
Assigning same weights on the failed topology after **last iteration**
7
5
6
3
7
8
5
6
3
(N+1) OH - link
Demand & Capacity as input
(N+1) OH
Cost 1
Cost 2
Best Cost

Figure 4.6: Initial topologies used for cost comparisons.

This process continues till the best assignment of weights leading to the best cost is obtained. After the final iteration, the added link is taken out for the new topology, namely [(N+1)OH - link] illustrating the fact that the topology has a link failure as shown in Figure 4.6. The weights assigned to (N+1) topology are transferred to the [(N+1)OH - link] topology. Now, based on the input demand and capacity, the cost function is calculated to get the cost of this failed topology, i.e., cost1.

The important thing here is to have the weights obtained from the previous topology be assigned back to the link-failed topology, i.e., the weights are still the same as shown in Figure 4.6. It is to be noted here that the weights obtained for the (N+1)OH topology will have one more weight value than the failed topology. So, the weight corresponding to the failed link has to be removed before assigning weights to the failed topology.

4.7.2 Strategy 1 (Strat1)

In this case, initially, topologies without an added link as well as with added link are considered simultaneously as shown in Figure 4.7.

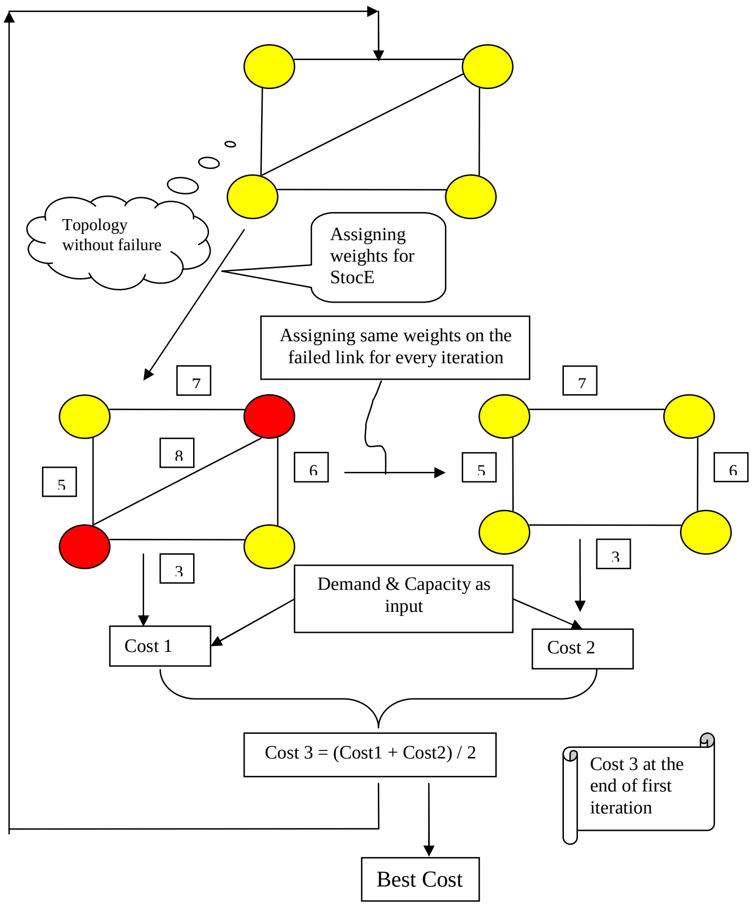

Topology without failure
Assigning weights for StocE
Assigning same weights on the failed link for every iteration
7
8
5
6
3
7
5
6
3
Demand & Capacity as input
Cost 1
Cost 2
Cost 3 = (Cost1 + Cost2) / 2
Cost 3 at the end of first iteration
Best Cost

Figure 4.7: First strategy implemented for robust link failure topology.

Here we consider one iteration at a time. The same StocE inputs, $R = 20$, $p_{incr} = 3$ and $p_0 = 0$ are used for both the topologies, i.e., without the added link as well as with the added link. Weights are assigned to the topology with no-failure and transferred to the topology with-failure. Then, the shortest paths for this particular assignment are obtained for both topologies. It is made sure that both the topologies, have the same weights on the corresponding arcs. Based on the demands and capacities on the links, the respective costs are calculated. At the end of the first iteration, the cost from the first topology, cost 1, and the cost for the second topology, cost 2, as shown in Figure 4.7, are obtained. The average of these two costs, cost 3, is again given as input to the same topologies for the next iteration. The purpose of this step is to get a final assignment of weights which is practically robust, i,e., the same set of weights can be used for both with-link and without-link-failure. This process continues till we finally end up with the best solution. Through this approach, an attempt is made to have the same set of weights as described in the previous sections and this is well-supported with the results discussed in detail in later chapters.

4.7.3 Strategy 2 (Strat2)

A compound move in every iteration required a great amount of computing power in the first strategy. This discouraging factor turned out to be a motivation for the implementation of the second strategy. Consider the topology without the added

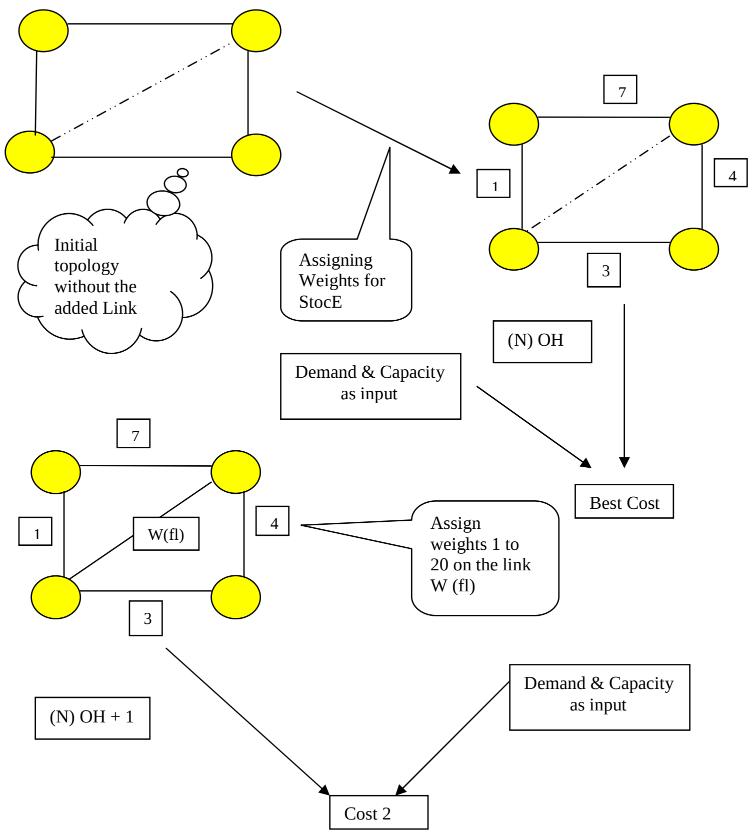

Initial topology without the added Link
Assigning Weights for StocE
7
1
4
3
(N) OH
Demand & Capacity as input
Best Cost
7
1
W(fl)
4
3
Assign weights 1 to 20 on the link W (fl)
(N) OH + 1
Demand & Capacity as input
Cost 2

Figure 4.8: Considering 20 weights on the failed link.

link as discussed in the previous sections, namely N(OH), as shown in Figure 4.8. OSPFWS is run, weights are assigned and the best cost is obtained. After calculating the cost for N(OH), keeping the weights the same on all the links, the failed link is added with weight W_{FL}, where W_{FL} takes weights 1 to 20 as shown in Figure 4.8. For each value from 1 to 20, the cost is calculated. Here, there is no need to run the entire algorithm again, rather, only the cost is calculated.

In all the twenty cases of cost calculation, the remaining weights are the same. So, finally 20 costs due to these 20 weights are obtained and the best cost amongst these is considered for comparison with strategy 1 for the (N+1) topology, i.e., cost 2. Also, cost 1 is compared to the cost for the (N) topology of strategy 1. The results depict that strategy 2 performs better than strategy 1 for the test cases considered; both in terms of cost as well as time.

4.8 Summary

This chapter discusses in details, the second phase of this thesis work, i.e., reliability and the approaches that have been considered to evaluate the reliability of a stochastic network. When a failure occurs in the network, especially a single link failure, the mechanism involved in detecting and recovering through OSPF is portrayed in this chapter. Some of the issues related to link failure from the literature are mentioned in this chapter. The approach used in this work, trying to assign robust weights without changing them in the case of a link failure, is justified in the section "Importance of No-Weight Changes in OSPF-IS/IS Networks" of this chapter. An example of a network with a single link failure is appended towards the end of the chapter. This is done to show that the reliability issue in the OSPF process is dependent on the way routing is done, i.e., there could be cases where the capacity is used for flows which will not arrive to the destination, thus decreasing the available capacity for other demands. Hence, it is assumed that the demands are routed in an increasing order, i.e., smallest demands are routed first based on shortest paths.

In chapter 6, link failure results section, it will be shown that many weight changes are bad. Changing the weight on a single congested link ("hot spot") creates a localized disturbance limited to those flows that use the congested link. Generally, the flows using the congested link will suffer inferior service only for a short period of time.

Towards the end of the chapter, two strategies have been discussed which eventually lead us to conclude that robust weight assignment gives us a wider perspective of the link failure problem

Chapter 5

Problem Description

5.1 Problem Formulation

The Open Shortest Path First (OSPF) protocol, defined in RFC 2328, is an Interior Gateway Protocol used to distribute routing information within a single Autonomous System. OSPF is a link-state protocol. The link can be considered as an interface on the router. The state of the link is a description of that interface and of its relationship to its neighboring routers. A description of the interface would include, for example, the IP address of the interface, the mask, the type of network it is connected to, the routers connected to that network and so on. The collection of all these link-states would form a link-state database.

The following section provides details of notations, assumptions and terminology [66] used to formulate the OSPF weight setting problem considering cost and maximum utilization.

5.2 Notations

G	Graph.
N	Set of nodes.
n	A single element in set N.
A	Set of arcs.
A^t	Set of arcs representing shortest paths from all sources to destination node t.
a	A single element in set A. It can also be represented as (i, j).
s	Source node.
v	Intermediate node.
t	Destination node.
D	Demand matrix.
$D[s, t]$	An element in the demand matrix that specifies the amount of demand from source node s to destination node t. It can also be specified as d_{st}.
w_{ij}	Weight on arc (i, j). If $a = (i, j)$, then it can also be represented as w_a.
c_{ij}	Capacity on arc (i, j). If $a = (i, j)$, then it can also be represented as c_a.
u_{ij}	Utilization on link (i, j).
Φ	Cost function.
$\Phi_{i,j}$	Cost associated with arc (i, j). If $a = (i, j)$, then it can also be represented as Φ_a.
δ_u^t	Outdegree of node u when destination node is t.
$\delta^+(u)$	Outdegree of node u.
$\delta^-(u)$	Indegree of node u.
l_a^t	Load on arc a when destination node is t.
l_a	Total load on arc a.
$f_a^{(s,t)}$	Traffic flow from node s to t over arc a.
MU	Maximum utilization of the network. It is defined as the utilization of the link whose value is more than that of all other links.
MC	Maximum cost of links present in the network.
$\mid A \mid$	Number of arcs.
$SetCA$	Set of congested arcs.

5.3 Assumptions and Terminology

1. A single element in the set N is called a “Node”.

2. A single element in the set A is called an “Arc”.

3. A set $G = (N, A)$ is a graph defined as a finite nonempty set N of nodes and a collection A of pairs of distinct nodes from N.

4. A “directed graph” or “digraph” $G = (N, A)$ is a finite nonempty set N of nodes and a collection A of ordered pairs of distinct nodes from N; each ordered pair of nodes in A is called a “directed arc”.

5. A digraph is “strongly connected” if for each pair of nodes i and j there is a directed path $(i = n_1, n_2, ..., n_l = j)$ from i to j. A given graph G must be strongly connected for the OSPFWS problem.

6. A “demand matrix” is a matrix that specifies the traffic flow between s and t, for each pair $(s, t) \in N \times N$.

7. $(n_1, n_2, ..., n_l)$ is a “directed walk” in a digraph G if (n_i, n_{i+1}) is a directed arc in G for $1 \leq i \leq l - 1$.

8. A “directed path” is a directed walk with no repeated nodes.

9. Given any directed path $p = (i, j, k, ..., l, m)$, the “length” of p is defined as $w_{ij} + w_{jk} + ... + w_{lm}$.

10. The "outdegree" of a node u is the size of set of arcs leaving node u, i.e., $\{(u, v) : (u, v) \in A\}$.

11. The "indegree" of a node u is the size of set of arcs entering node u, i.e.,$\{(v, u) : (v, u) \in A\}$.

12. The input to the OSPFWS problem will be a graph G, a demand matrix D, and capacities of each arc.

In OSPF, the network operator assigns a weight w_a to each link a $\in$ A, and shortest paths from each router to each destination are computed using these weights as lengths of the links. The shortest path from a point to a destination is calculated using Dijkstra's algorithm, as illustrated in the next section. In practice, link weights are integer coded on 16 bits, therefore they can take any value between 1 and $65, 535$. In each router, represented by a node of the graph, the next link on all shortest paths to all possible destinations is stored in a table. A flow arriving at the router is sent to its destination by splitting the flow between the links that are on the shortest paths to the destination. The splitting is done using pseudo-random methods leading to an approximately even splitting. The quality of OSPF routing depends highly on the choice of weights.

5.4 Problem Statement

The OSPF weight setting (OSPFWS) problem can be stated as follows: Given a network topology and predicted traffic demands, a set of OSPF weights is found that optimizes network utilization. More precisely, given a directed network $G = (N, A)$, a demand matrix D, and capacity C_a for each arc $a \in A$, the objective is to determine a positive integer weight $w_a \in [1, w_{max}]$ for each arc $a \in A$ such that the objective function or cost function Φ is minimized; w_{max} is a user-defined upper limit. The chosen arc weights determine the shortest paths, which in turn completely determine the routing of traffic flow, the loads on the arcs, and the value of the cost function Φ. The quality of OSPF routing depends highly on the choice of weights. Figure 5.1 depicts a topology with assigned weights within the range $[1, 20]$. A solution for this topology can be $(12, 3, 1, 15, 1, 19, 13)$. These elements (i.e., weights) are arranged in a specific order for simplicity. The cost is calculated based on the weights assigned in the first phase of this work.

In the second phase, two strategies have been adopted to assign robust weights. Once the assignment of weights is done on a particular topology, a strategy is adopted to lock these weights. By targeting the same topology with one of the links dropped, i.e., link failed, the cost is calculated and made sure that the weights assigned in case of no link failure are the same as the ones assigned for the link failure topology, resulting in close to optimum solutions.

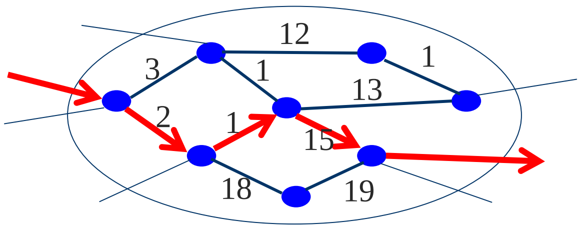


Figure 5.1: Representation of a topology with assigned weights.

5.4.1 Dijkstra's Algorithm

Djikstra's algorithm [67] solves the problem of finding the shortest path from a point in a graph (the source) to a destination. This problem is sometimes called the single-source shortest paths problem since one can find the shortest paths from a given source to all points in a graph in the same time. The graph representing all the paths from one vertex to all the others must be a spanning tree - it must include all vertices. There will also be no cycles as a cycle would define more than one path from the selected vertex to at least one other vertex. For a graph, $G = (V, E)$ where 'V' is a set of vertices and 'E' is a set of edges, Dijkstra's algorithm keeps two sets of vertices: 'S', the set of vertices whose shortest paths from the source have already been determined and 'V-S' the remaining vertices.

The other data structures needed are: 'd', representing an array of best estimates of shortest path to each vertex, and, 'pi', that represents an array of predecessors for each vertex. The basic mode of operation is:

1. Initialize d and P_i, for $i = 1, ..., n$,

2. Set S to empty,
3. While there are still vertices in V-S,
    (a) Sort the vertices in V-S according to the current best estimate of their distance from the source,
    (b) Add u, the closest vertex in V-S, to S,
    (c) Relax all the vertices still in V-S connected to u.

The relaxation process updates the costs of all the vertices, v, connected to a vertex, u, if we could improve the best estimate of the shortest path to v by including *(u,v)* in the path to v. This sets up the graph so that each node has no predecessor *(pi[v] = nil)* and the estimates of the cost (distance) of each node from the source *(d[v])* are infinite, except for the source node itself *(d[s] = 0)*. The relaxation procedure checks whether the current best estimate of the shortest distance to v *(d[v])* can be improved by going through u (i.e. by making u the predecessor of v).

Figure 5.2 gives a formal algorithmic description of Dijkstra's algorithm. In this algorithm, $A(i)$ represents the adjacency list which contains the arcs emanating from node i.

Determining weights for a particular topology can be done in many ways. One of them is by making weights proportional to their physical distances. The other could be by making the weight of the link inversely proportional to its capacity without taking any knowledge of the demand into account, a Cisco approach. The capacity

```
Algorithm Dijkstra
Begin
          (*$N$ is the set of nodes *)
          (*$S, \bar{S}$ are two temporary sets of nodes *)
          (*$n$ is the total number of nodes *)
          $S := \Phi$; $\bar{S} := N$;
          $d(i) := \infty$ for each node $i \in N$;
          $d(s) := 0$ and $pred(s) := 0$;
          While $\mid S \mid < n$
          Begin
                    let $i \in \bar{S}$ be a node for which $d(i) = min\{d(j) : j \in \bar{S}\}$;
                    $S := S \cup i$;
                    $\bar{S} := \bar{S} - i$;
                    For $(i, j) \in A(i)$
                              If $d(j) > d(i) + c_{ij}$ Then $d(j) := d(i) + c_{ij}$ and $pred(j) := i$;
                    End For
          End While
          End
End.
```

Figure 5.2: Dijkstra's algorithm.

of the link is a measure of how much traffic flow the link can sustain. The demand of the link tells us how much traffic flow has to be sent from source to destination. The demand matrix can be based on concrete measures of the flow between source-destination pairs or it can also be based on a concrete set of customer subscriptions to virtual leased line services.

Knowing the optimal solution for the general routing problem is an important benchmark for judging the quality of solutions based on OSPF routing. It is to be ensured that no packet gets sent across overloaded arcs. Though the splitting of the paths for load balancing depending on shortest path is complicated, yet it can be taken as an even split in a nutshell.

In OSPF routing, the length of the arcs is the sum of its arc weights, W_a. The extra condition is that all flow leaving a node aimed at a given destination is evenly spread over arcs on shortest paths to that destination, making the problem NP-hard. For requiring guarantees on delay or jitter, congestion cannot be tolerated. The solution to this problem has to be "Load Balancing" so that the maximum utilization on the links is reduced as much as possible.

With each arc $a \in A$, a cost function $\Phi_a(l_a)$ is associated with load l_a, depending on how close the load is to the capacity c_a. However the formal objective is to distribute the demand flow so as to minimize the sum

$$\Phi = \sum_{a \in A} \Phi_a(l_a) \tag{5.1}$$

of the resulting costs over all arcs. Usually, Φ_a increases rapidly as loads exceed capacities, hence the objective is to keep the maximum utilization $max_{a \in A}(l_a/c_a)$ below 1.

5.4.2 Methodology

As discussed in [47], a problem in the current formulation of Φ is that it does not provide a universal measure of congestion. Independently of the network topology and demand matrix, it is natural to require that the maximum utilization remains below 1. For the sake of comparison, a normalized cost function is used:

$$\Phi^* = \frac{\Phi}{\Psi} \tag{5.2}$$

where Ψ is the cost that would have been obtained if all the flow was sent along hop-count shortest paths and the capacities matched the loads. When capacity matches the load on a link a, $\Phi_a(c_a)/(c_a)$ per unit of flow on a [68], is given by:

$$\Phi_a(c_a)/(c_a) = 1/3 + 3.1/3 + 10.7/30 + 70.1/10 = 10\frac{2}{3} \tag{5.3}$$

This cost per unit flow when load matched capacity is thus constant over all arcs and is equal to $10\frac{2}{3}$. If $\Delta(s,t)$ is the hop count distance between 's'and 't', then:

$$\Psi = \sum_{(s,t)\in N\times N} (10\frac{2}{3}.D[s,t].\Delta(s,t)) \tag{5.4}$$

With this scaling, it can be interpreted that if $\Phi^* \geq 1$, then the routing is as bad as if all flows were along hop-count shortest paths with loads matching the capacities. The same cost can also stem from some loads going above the capacity and others going below, or by flows following longer detours via less utilized arcs. However, it can be summed up saying that a routing congests a network if $\Phi^* > 1$.

Three flavors of synthetic graphs are considered in this work for the sake of experimentation:

2-level hierarchical graphs : Graphs produced using the generator GT-ITM [69] based on the model of Calvert et.al [70, 71].

Purely Random Graphs : The probability of having an arc between two nodes is given by a constant parameter used to control the density of the graph. All arc capacities are set to 1000.

Waxman Graphs : Nodes are uniformly distributed points in a unit square and the probability of having an arc between two nodes 'u' and 'v' is given by:

$$p(i,j) = \alpha e \frac{-L_2(u,v)}{\beta\Delta} \tag{5.5}$$

where α and β are parameters used to control the density of the graph, $L_2(u,v)$ is the Euclidean distance between 'u' and 'v' and Δ is the maximum distance between two nodes. All arc capacities are set to 1000.

The demands are generally modeled inspired by classical entropy models for urban traffic [72]. For each node 'x' two random numbers are selected, O_x and $d \in [0,1]$. For each pair (x,y) of nodes, a random number $c_{(x,y)} \in [0,1]$ is selected. If the Euclidean distance between 'x' and 'y' is $\delta(x,y)$, the demand between 'x' and 'y' is:

$$\alpha O_x d_y c_{(x,y)} e^{-\delta(x,y)/2\Delta} \tag{5.6}$$

where α is a parameter and Δ is the largest Euclidean distance between any pair of nodes. The distance factor $e^{-\delta(x,y)/2\Delta}$ implies relatively more demand between close pair of nodes. In our experimental set up, one demand matrix is used for each network, and it is scaled up by multiplying each entry by a constant at different levels to obtain different total demands, which allows to measure, for a given routing, at which level of total demand congestion occurs for the given topology and demand pattern. We ran the simulations based on a given set of demands. In our case, these graphs and related data sets have not been generated, but rather obtained from existing work. This way, we were able to compare our results to existing ones.

5.4.3 Normalized Cost Function

A normalizing scaling factor for the cost function is used to compare costs across different sizes and topologies of networks.

This is given by:

$$\Psi = \sum_{(s,t)\in N\times N} (D(s,t).dist_1(s,t)) \tag{5.7}$$

where $dist_1(s,t)$ is the distance measured between nodes 's' and 't' in unit weights and $dist_1(s,t) = 10\frac{2}{3} \times \Delta(s,t)$

5.4.4 Modeling the Cost Evaluation

Evaluating cost for the static case

A directed multigraph $G = (N, A)$ with arc capacities $\{ C_a \}_{a \in A}$, demand matrix D and weight setting $\{ W_a \}_{a \in A}$ are given. The graph is sparse with $|A| = O(|N|)$ and the maximal distance between any two nodes is $O(|N|)$. One destination t at a time is considered and the total flow from all sources $s \in N$ to t is computed. This gives rise to a certain partial load $l_a^t = \sum_{s \in N} f(a)^{(s,t)}$ for each arc, where the variable $f(a)^{(s,t)}$ gives the amount of traffic flow from s to t over a. The load l_a on arc a is computed as $\sum_{t \in N} l(a)^t$ based on the above computation for each destination t. This problem can be formulated into the following multi-commodity flow problem:

$$Minimize \qquad \Phi = \sum_{a \in A} \Phi_a(l_a) \tag{5.8}$$

subject to the following conditions:

$$\sum_{a \in \delta^+(u)} f(a)^{(s,t)} - \sum_{a \in \delta^-(u)} f(a)^{(s,t)} = \begin{cases} D[s,t], & \text{if } u = s, \\ -D[s,t], & \text{if } u = t,\ u, s, t \in N \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

$$l_a = \sum_{(s,t) \in N \times N} f(a)^{(s,t)}, a \in A, \tag{5.10}$$

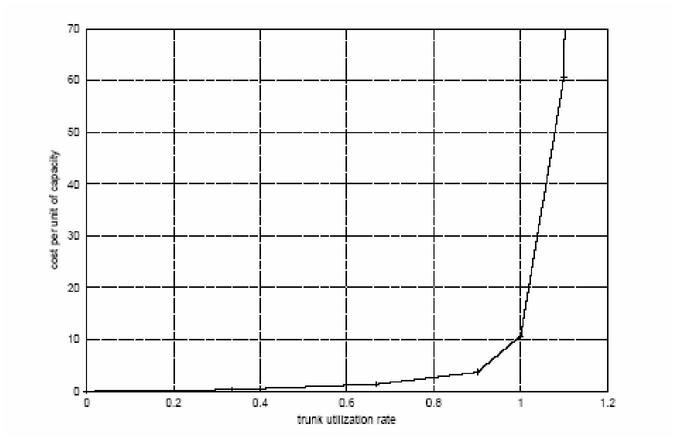


Figure 5.3: Linear Cost function Curve [3]

.

$$f_a^{(s,t)} \geq 0 \qquad a \in A; s,t \in N$$

where $\delta^+(u)$ and $\delta^-(u)$ denotes the set of arcs leaving a node u and the set of arcs entering a node u respectively. In the setup, Φ_a are piece wise linear functions, which increasingly penalizes flows approaching or violating the capacity limits [3].

$$\Phi_a^(l_a) \quad = \begin{cases} u_a, & \text{u}_a \in \ 0,1/3, \\ 3.u_a - 2/3, & \text{u}_a \in 1/3,2/3, \\ 10.u_a - 16/3, & \text{u}_a \in 2/3,9/10, \\ 70.u_a - 178/3, & \text{u}_a \in 9/10,1, \\ 500.u_a - 1468/3, & \text{u}_a \in 1,11/10, \\ 5000.u_a - 16318/3, & \text{u}_a \in 11/100,\infty. \end{cases}$$

$\Phi_a(0) = 0$ and derivative

$$\Phi_a^{'}(l) = \begin{cases} 1 & \text{for } 0 \leq l/c_a < 1/3, \\ 3 & \text{for } 1/3 \leq l/c_a < 2/3, \\ 10 & \text{for } 2/3 \leq l/c_a < 9/10, \\ 70 & \text{for } 9/10 \leq l/c_a < 1, \\ 500 & \text{for } 1 \leq l/c_a < 11/10, \\ 5000 & \text{for } 11/100 \leq l/c_a < \infty. \end{cases} \tag{5.12}$$

Dijkstra's algorithm is used to calculate the distances away from some source. However, this algorithm can be used to compute the flow of t by computing all distances to t by reversing the orientation of all arcs in G. The set A^t of arcs on shortest paths to t is computed after calculating the distance d_x^t to t for each node.

$A^t \quad = \{(x, y) \in A : d_x^t - d_y^t = W_{(x,y)}\}$

For each node x, let δ_x^t denotes its outdegree in A^t, i.e.,

$\delta_x^t = |\{y \in N : (x, y) \in A_t\}|$.

For all $(y, z) \in A^t$,

$$l_{(y,z)}^t = \frac{1}{\delta_y^t}\{D(y, t) + \sum_{(x,y)\in A^t} l_{(x,y)}^t\} \tag{5.13}$$

Evaluating cost for the dynamic case

To overcome the bottleneck of evaluating the cost for different weight settings, it is assumed that only few arcs change weight while evaluating consecutive weight settings. For recomputation, only time equal to the number of arcs incident to node'x' whose d_x^t to t changes is considered. The set of changed distances immediately gives us a set of "update" arcs to be added to or deleted from A^t.

5.4.5 New Cost Function

The cost function proposed by Fortz and Thorup [47] is focussed on the ratio of load to capacity on a particular link as represented in Equation 5.13. The optimization of the number of congested links for larger topologies is not an objective of this cost function. This problem is addressed through a new cost function shown in [73]. This cost function is given by:

$$\Phi \qquad = MU + \frac{\Sigma_{a \in SetCA}(l_a - c_a)}{E} \tag{5.14}$$

where 'MU' gives the maximum utilization in the network and the other term represents the extra load on the network. To normalize the cost function, the second term is divided by the number of edges present in the network. Through simulations, it has been shown in the *Results and Discussion* chapter that the number of congested links have drastically reduced in the network. However, the trade-off is with the MU and the cost function provides better results in terms of number of congested links and percentage of extra load than the previous one especially in the case of topologies with higher demands.

5.5 Summary

This chapter gives in detail description of the core of this thesis work. It introduces the notations, assumptions and terminology used in defining the problem. It discusses the Dijkstra's algorithm on which routing in OSPF is done. The demands used in this work and the three flavors of synthetic graphs used are discussed in this chapter. To compare the results of this work with other published results, normalization of the cost function is made. Towards the end of the chapter, the cost evaluation for the static and the dynamic cases are also described. The problem has been clearly represented dividing the work into two phases. The first of which illustrates the implementation of StocE on the given problem whereas in the second phase, an attempt has been made to come up with two strategies for robust weight assignments in case of single link failures in the network.

Chapter 6

Results and Discussion

This chapter describes in detail the experimental setup and the computational results of this work. A detailed description of tuning the parameters is illustrated. For comparing with related work described in previous chapters, the same test cases as proposed by Fortz and Thorup [33, 47, 57] are considered. Comparably, long time is consumed for the cost function calculations during the simulations especially in the case of StocE. This can be explicitly made out from the figures and tables that follow.

6.1 Experimental Setup

The coding of the algorithm is done in C language using Microsoft Visual $C++6.0$ compiler. To maintain consistency with the results, a set of 12 systems with the following configurations are used:

- Pentium(R) 4,
- CPU 2.66GHz, and
- 1.0 GB of RAM

and another set of 4 systems with the following configurations are used:

- Pentium(R) 4,
- CPU 2.66GHz, and
- 512 MB of RAM

Table 6.1 shows the characteristics of the test cases.

Table 6.1: Test Cases

Test Code	**Network type**	**N**	**A**
h50N148a	2-level hierarchical graph	50	148
h50N212a	2-level hierarchical graph	50	212
h100N280a	2-level hierarchical graph	100	280
h100N360a	2-level hierarchical graph	100	360
r50N228a	Random graph	50	228
r50N245a	Random graph	50	245
r100N403a	Random graph	100	403
r100N503a	Random graph	100	503
w50N169a	Waxman graph	50	169
w50N230a	Waxman graph	50	230
w100N391a	Waxman graph	100	391
w100N476a	Waxman graph	100	476

For each test case, the table lists its network type, the number of nodes (N), the number of arcs or edges (A). The *2-level hierarchical networks* are generated using the GT-ITM generator[1], based on a model of Calvert [70] and Zegura [71]. This model places nodes in a unit square, thus getting a distance $\delta(x, y)$ between each pair of nodes. These distances lead to a random distribution of 2-level graphs with arcs divided into two classes:

- Local access arcs, and
- Long distance arcs.

In *hierarchical networks*, local access arcs have capacities equal to 200, while long distance arcs have capacities equal to 1000. In *Random networks* and *Waxman networks* capacities are set to 1000 for all arcs. Fortz and Thorup [46] generated the demands to force some nodes to be more active senders or receivers than others, thus modeling *hot spots* on the network. This is further demonstrated with the help of figures in the sections ahead.

In our work, we considered 12 test cases as shown in table 6.1. Each test case has 12 demands.

[1]E.W.Zegura, GT-ITM:Georgia tech internetwork topology models (software)

6.2 Tuning of parameters for StocE

The three parameters mainly responsible for efficient solutions in StocE are:

1. An initial value of the control parameter p_0,

2. The increment value of the control parameter p_{incr}, and

3. The reward parameter R.

Experimental studies [1] indicate that a value of R between 10 and 20 gives good results.

For the OSPFWS problem, extensive experimentations were conducted to tune these parameters. The combination of parameters selected are:

- $R \rightarrow$ 10, 15 and 20

- $p_{incr} \rightarrow$ 1, 2 and 3

- $p_0 \rightarrow$ 0, 1, 2, 3 and 4.

as shown in table 6.2.

Out of the 12 test cases available, the Hierarchial type of networks has links with capacities 200 and 1000, whereas the other two types, i.e., Random and Waxman networks, have links with capacity 1000. We consider the topologies with the least number of nodes using the smallest demand and the largest demand, and the

Table 6.2: Initial Parameter Tuning

S.No	**R**	**Pincr**	**Pzero**
1	10	1	0
2	10	1	1
3	10	1	2
4	10	1	3
5	10	1	4
6	10	2	0
7	10	2	1
8	10	2	2
9	10	2	3
10	10	2	4
11	10	3	0
12	10	3	1
13	10	3	2
14	10	3	3
15	10	3	4
16	15	1	0
17	15	1	1
18	15	1	2
19	15	1	3
20	15	1	4
21	15	2	0
22	15	2	1
23	15	2	2
24	15	2	3
25	15	2	4
26	15	3	0
27	15	3	1
28	15	3	2
29	15	3	3
30	15	3	4
31	20	1	0
32	20	1	1
33	20	1	2
34	20	1	3
35	20	1	4
36	20	2	0
37	20	2	1
38	20	2	2
39	20	2	3
40	20	2	4
41	**20**	**3**	**0**
42	20	3	1
43	20	3	2
44	20	3	3
45	20	3	4

topologies with the maximum number of nodes having the smallest demand and the largest demand corresponding to the three kinds of networks available.

Table 6.3 illustrates the methodology adopted for the final tuning of parameters.

Table 6.3: Final Tuning of Parameters

Topology	**Capacity**	**Demand**	**Network Type**
Smallest Topology	Fixed Capacity	Smallest Demand	2-level hierarchical graph
Smallest Topology	Fixed Capacity	Largest Demand	2-level hierarchical graph
Largest Topology	Variable Capacity	Smallest Demand	2-level hierarchical graph
Largest Topology	Variable Capacity	Largest Demand	2-level hierarchical graph
Smallest Topology	Fixed Capacity	Smallest Demand	Random graph
Smallest Topology	Fixed Capacity	Largest Demand	Random graph
Largest Topology	Fixed Capacity	Smallest Demand	Random graph
Largest Topology	Fixed Capacity	Largest Demand	Random graph
Smallest Topology	Fixed Capacity	Smallest Demand	Waxman graph
Smallest Topology	Fixed Capacity	Largest Demand	Waxman graph
Largest Topology	Fixed Capacity	Smallest Demand	Waxman graph
Largest Topology	Fixed Capacity	Largest Demand	Waxman graph

The final tuning was done on three topologies, considering demand 1, demand 4, demand 8 and demand 12. It is concluded that $R = 20$, $p_{incr} = 3$ and $p_0 = 0$ are suitable values. This conclusion comes from the extensive experimentation of the parameters and can be seen from the figures 6.1, and 6.2.

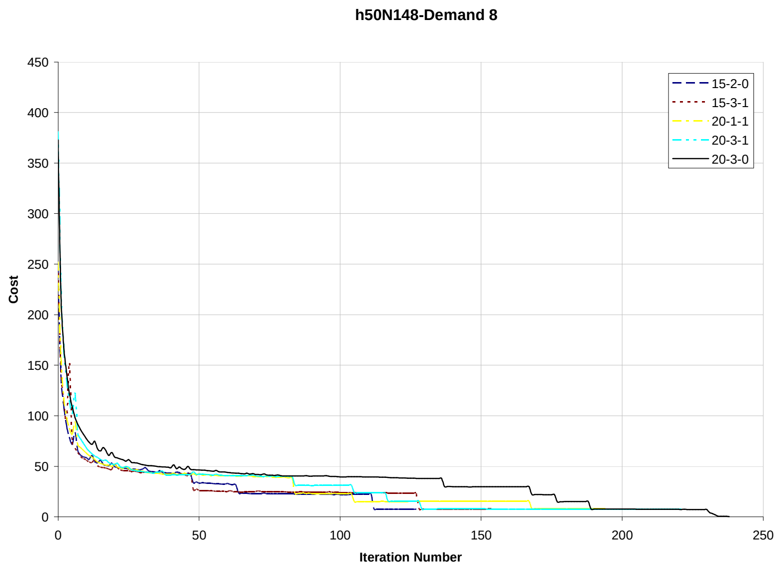


Figure 6.1: Iteration comparison for h50N148a demand8 showing $R = 20, P_{incr} = 3, and P_{zero} = 0$ as the best combination

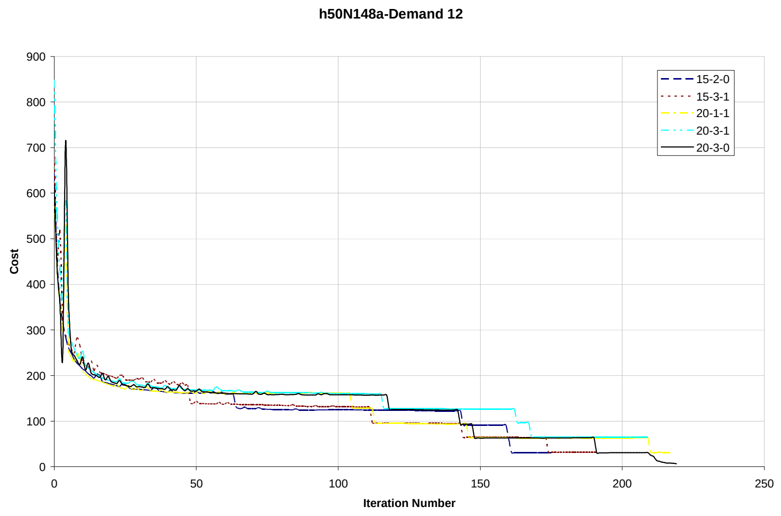


Figure 6.2: Iteration comparison for h50N148a demand12 showing $R = 20, P_{incr} = 3, and P_{zero} = 0$ as the best combination

6.3 Sum of Demands

For each instance, 12 demand matrices are considered. The highest demand matrix, i.e., demand 12, is obtained by gradually multiplying the lowest demand, i.e., demand 1, with a scaling factor. This process of getting demand 3, demand 4, etc., continues till the maximum demand, i.e., demand 12, is obtained through corresponding scaling factors. The sum of all the individual demands for Hierarchical, Random and Waxman graphs are shown in tables 6.4, 6.5, and 6.6 respectively.

Table 6.4: Sum of Demands for Hierarchical Graphs

	h50N148a	**h50N212a**	**h100N280a**	**h100N360a**
Demand 1	410.6047	280.2244	383.7675	1033.879
Demand 2	821.2814	560.4488	767.7351	2067.757
Demand 3	1231.922	840.6732	1151.303	3101.636
Demand 4	1642.563	1120.898	1535.07	4135.515
Demand 5	2053.204	1401.122	1918.838	5169.394
Demand 6	2463.844	1681.346	2302.605	6203.272
Demand 7	2874.485	1961.571	2686.373	7237.151
Demand 8	3285.126	2241.795	3070.14	8271.03
Demand 9	3695.767	2522.02	3453.908	9304.909
Demand 10	4106.407	2802.244	3837.675	10338.79
Demand 11	4517.048	3082.468	4221.443	11372.67
Demand 12	4927.689	3362.693	4605.21	2406.54

Table 6.5: Sum of Demands for Random Graphs

	r50N228a	**r50N245a**	**r100N403a**	**r100N503a**
Demand 1	3523.431	4463.462	5774.737	8382.862
Demand 2	7046.862	8926.923	11549.47	16765.72
Demand 3	10570.29	13390.39	17324.21	25148.59
Demand 4	14093.72	17853.85	23098.95	33531.45
Demand 5	17617.15	22317.31	28873.69	41914.31
Demand 6	21140.58	26780.77	34648.42	50297.17
Demand 7	24664.02	31244.23	40423.16	58680.04
Demand 8	28187.45	35707.69	46197.9	67062.9
Demand 9	31710.88	40171.16	51972.63	75445.76
Demand 10	35234.31	44634.62	57747.37	83828.68
Demand 11	38757.74	49098.08	63522.11	92211.49
Demand 12	42281.17	53561.54	69296.84	100594.3

Table 6.6: Sum of Demands for Waxman Graphs

	w50N169a	**w50N230a**	**w100N391a**	**w100N476a**
Demand 1	2117.622	3287.217	4039.477	5291.092
Demand 2	4235.244	6574.434	8078.954	10582.19
Demand 3	6352.865	9861.651	12118.43	15873.28
Demand 4	8470.487	13148.87	16157.91	21164.37
Demand 5	10588.11	16436.08	20197.39	26455.46
Demand 6	12705.73	19723.3	24236.86	31746.55
Demand 7	14823.35	23010.52	28276.34	37037.65
Demand 8	16940.97	26297.74	32315.82	42328.74
Demand 9	19058.6	29584.95	36355.29	47619.83
Demand 10	21176.22	32872.17	40394.77	52910.92
Demand 11	23293.84	36159.39	44434.25	58202.02
Demand 12	25411.46	39446.6	48473.73	63493.11

6.4 Phase I - Comparison of 12 test cases

6.4.1 Simulation Scenario

In this section, we compare the StocE algorithm with InvCap, GAEric, HGAEric, LS(FT), and LPLB, for which results have been reported in [68], SA [73], and SimE [74]:

- **InvCap**: Weights are set proportional to the inverse of the link capacity, i.e., $w_a = \lceil C_{max}/C_a \rceil$, where C_{max} is the maximum link capacity (a Cisco approach),
- **GAEric**: The basic genetic algorithm (GA) designed by Ericsson without the local search used by the hybrid GA,
- **HGAEric**: The hybrid GA designed by Ericsson with the local search,
- **LS(FT)**: The local search algorithm of Fortz and Thorup,
- **LPLB**: The Linear Programming Lower Bound considered to be the benchmark for comparing costs obtained with each algorithm,
- **SA**: Simulated Annealing,
- **SimE**: Simulated Evolution.

The following is the simulation scenario:

Simulations for StocE- $12 demands \times 12 testcases = 144$

Simulations for SA- $12 demands \times 12 testcases = 144$

Simulations for SimE- $12 demands \times 12 testcases = 144$

The number of simulations considered to come to a consensus for the combination and tuning of parameters is equal to $45 \times 12 = 540$, where 45 combinations of the StocE parameters are considered initially for each of the 12 demands. Considering the runs, an average of 8 runs are taken for each simulation. For the three algorithms, SA, SimE and StocE, the total number of simulations required are $(144+144+144 = 432)$, i.e., $432 \times 8 = 3456$ runs. Hence, the presimulation work for getting to the right combination of parameters and the simulation for the first phase of this thesis work took $(540 + 432) = 972$ simulations, i.e., 7776 runs.

The comparisons are done considering the following four metrics:

- Cost,
- Maximum Utilization,
- Number of Congested Links, and
- Percentage of Extra Load.

The cost comparison is made with respect to 8 algorithms. Simulated results from StocE are compared to the existing published results of InvCap, GA Eric, HGA

Eric, LS and LPLB, SA, and SimE. Since the other three parameters, Maximum Utilization, Number of Congested Links, and Percentage of Extra Load, considered for comparison are not available with the published results, only comparisons of SA, SimE and StocE are made for these parameters.

6.4.2 Time comparison

To come to a consensus, few topologies were tested so that time taken for a particular algorithm can be obtained. During the experimentation, care was taken to check that the same topology is considered for the three algorithms: SA, SimE, and StocE using the same machine. It has been noticed that StocE took a considerable amount of time when compared to the other two. Even though SimE also consumed a good amount of time, the best cost was obtained through StocE. So, in cases where the network has to be established and the infrastructure is in novice state, StocE poses to be the best in comparison to the other two as it provides the best cost. Another point worth considering is, though SA and SimE took less time to converge to a good cost initially, there is hardly any improvement in the cost value at later stages. However, StocE performed better though it took a long time to converge as shown in figure 6.5.

Split Time Comparison

It has been well acknowledged initially that each heuristic has its own stopping criteria to obtain the best cost. During the course of pre-simulation work, exhaustive efforts were made in getting to a consensus regarding the parameters involved in deciding the stopping criteria. For example, the parameters that affect the stopping criteria for StocE have been set to R=20, Pincr= 3, and Pzero=0. However, this had been comprehended through a large amount of time requiring 675 simulations. A similar approach was needed to come to a consensus regarding the parameters of SA [73] and SimE [74].

Now, a time comparison for SA, SimE, and StocE for the largest topology, h100N360a - demand 12, is considered. With the normal stopping criteria, SA, SimE, and StocE achieved costs of 27.26716, 37.1553, and 15.10246 respectively at their own timings. SA took lesser time compared to SimE, which also took lesser time than StocE. To make sure whether a better cost can be achieved with SA and SimE, given the same time as StocE, a brute force of simulating for longer time was made.

Figure 6.3 shows the comparison of SA, SimE and StocE for the first 100 seconds. It clearly portrays that during this time interval, StocE outperforms SA and SimE in terms of cost. However in case of figure 6.4, with a time interval of 450 seconds, SA attains gradually a better cost than SimE and StocE. Figure 6.5 gives a clear representation of three algorithms for the 10 hours interval. At around 2500 seconds, SA and SimE have attained the best costs mentioned earlier. However, due to the extra time given, they could get to a minimal change in the cost unlike StocE which gradually went through this optimization in the stipulated time interval.

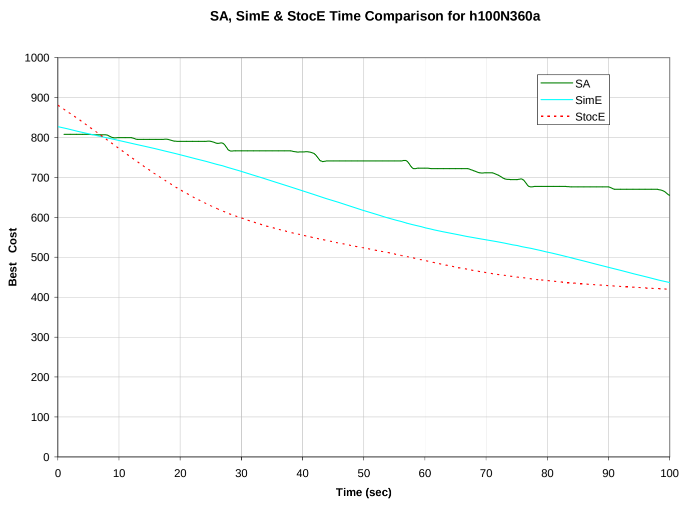


Figure 6.3: Time comparison of SA, SimE and StocE for h100N360a (demand 12) in 100 seconds interval

The bottom line of this comparison is that every heuristic is adjudged based on the parameters which play a vital role in deciding the stopping criteria.

Table 6.7 represents a cost-time comparison for SA, SimE, and StocE for h100N360a at fixed intervals. The criteria for choosing these intervals was based on the simulation results which showed changes in the cost during the course of time.

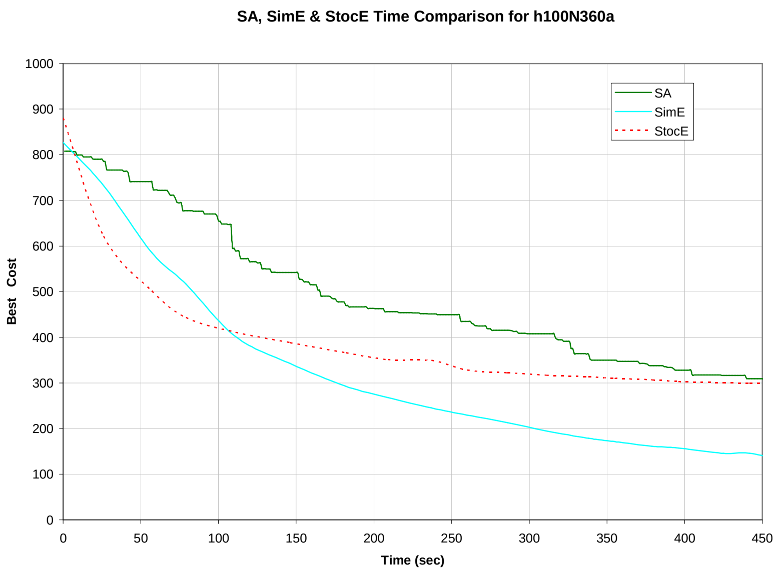


Figure 6.4: Time comparison of SA, SimE and StocE for h100N360a in 450 seconds interval

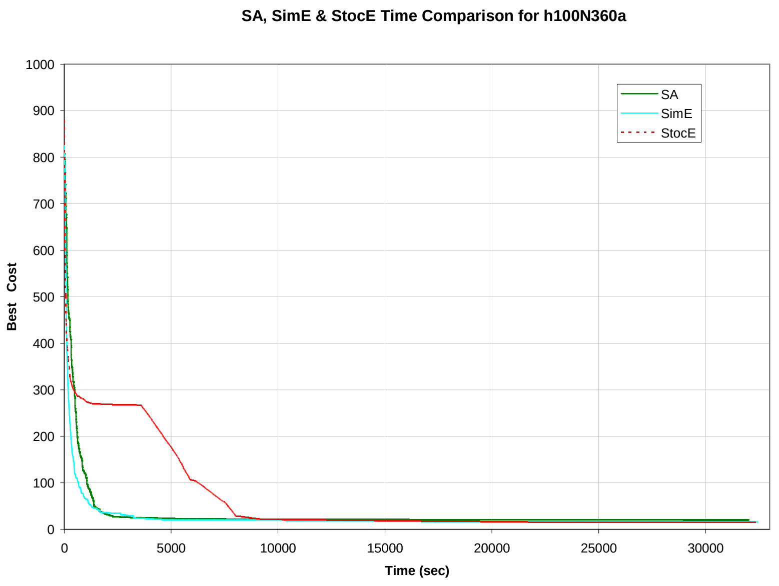


Figure 6.5: Time comparison of SA, SimE and StocE for h100N360a in 10 hours interval

Table 6.7: Cost-Time Comparison for SA, SimE & StocE for h100N360a

Time	**SACost**	**SimECost**	**StocECost**
	807.85	826.96	880.36
2mins	589.07	401.43	410.98
5mins 11sec	407.76	194.52	316.90
7mins 5sec	316.51	145.61	300.16
8mins 40sec	253.94	116.74	293.14
10mins 8sec	196.83	109.63	286.29
12mins	167.03	89.83	285.29
17mins	110.49	64.54	275.12
33mins 35sec	31.24	36.18	269.25
50mins	25.99	29.96	267.53
1hour	24.84	23.44	262.10
1hour 30mins	22.63	20.22	146.36
2hours	22.27	20.22	67.41
2hours 30mins	21.94	20.22	23.22
3hours	21.78	18.75	20.84
3hours 30mins	21.78	18.75	19.98
4hours	21.61	17.81	19.15
4hours 30mins	21.15	17.61	18.33
5hours	21.04	15.68	17.47
5hours 30mins	20.92	15.47	15.76
7hours	20.79	15.47	15.30
8hours	20.48	15.47	15.30
10hours	19.99	15.47	15.10

6.4.3 Comparisons for h100N360a

This section compares the cost, maximum utilization, number of congested links and percentage of extra load for one of the largest hierarchical topologies, h100N360a. From the table 6.8, it can be observed that InvCap, introduced by Cisco, performs the worst. Next in the series of deteriorating performance is SimE followed by Ericsson's GA, SA, Fortz & Thorup's Linear Search (LS), and Hybrid GA. The cost depends on the topology considered. LPLB is considered as the benchmark for comparing the costs across various algorithms. In this case, StocE outperforms the other algorithms in approximately 60% of the considered demands according to table 6.8. With the increase in demand, the difference in cost comparatively increases amongst the algorithms. However, StocE reaches this best cost at the expense of large amount of computation time. Every demand value in each experiment is run for 8 to 10 times and an average of these values is taken.

Table 6.8: Cost Comparison for h100N360a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F andT)**	**LPLB**	**SA**	**SimE**	**StocE**
1033.879	1.17	1.001	1.006	1	1	1.002	1.001	0.105
2067.757	1.17	1.002	1.008	1	1	1.003	1.002	0.111
3101.636	1.19	1.012	1.018	1.005	1	1.018	1.045	0.12
4135.515	1.27	1.043	1.048	1.028	1.02	1.044	1.023	0.145
5169.394	1.39	1.098	1.091	1.069	1.06	1.103	1.110	0.182
6203.272	1.53	1.16	1.142	1.128	1.1	1.17	1.152	0.564
7237.151	2.39	1.279	1.221	1.208	1.16	1.26	1.206	1.696
8271.03	10.66	1.441	1.331	1.312	1.25	1.406	1.314	3.15
9304.909	24.77	1.696	1.518	1.483	1.38	1.626	1.695	5.854
10338.79	53.24	2.536	2.063	2.077	1.76	2.562	2.365	8.609
11372.67	112.11	8.123	5.846	5.568	4.48	5.512	9.442	12.521
12406.54	**181.1**	**23.401**	**15.169**	**18.547**	**13.32**	**20.21**	**24.021**	**15.102**

Figure 6.6 represents the comparison of costs for the discussed algorithms.

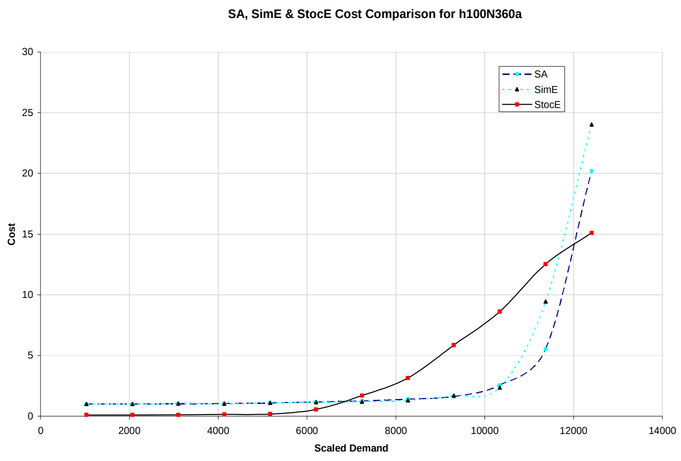


Figure 6.6: Cost comparison of SA, SimE and StocE for h100N360a

Figures 6.7, 6.8, and 6.9 represents the comparison of maximum utilization, number of congested links, and percentage of extra load, respectively. The cost increases progressively as the demand increases.

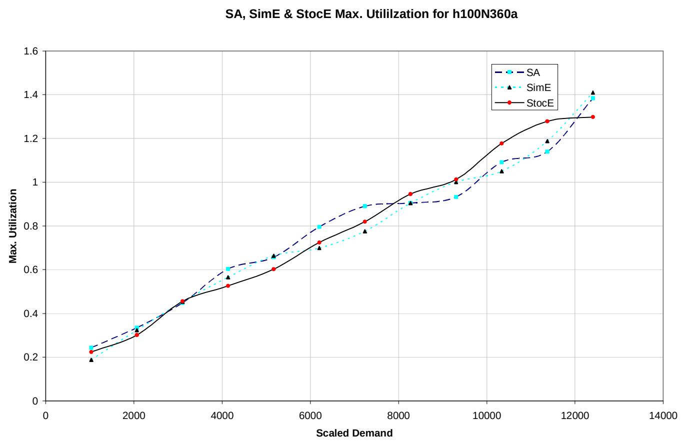


Figure 6.7: Maximum Utilization comparison of SA, SimE and StocE for h100N360a

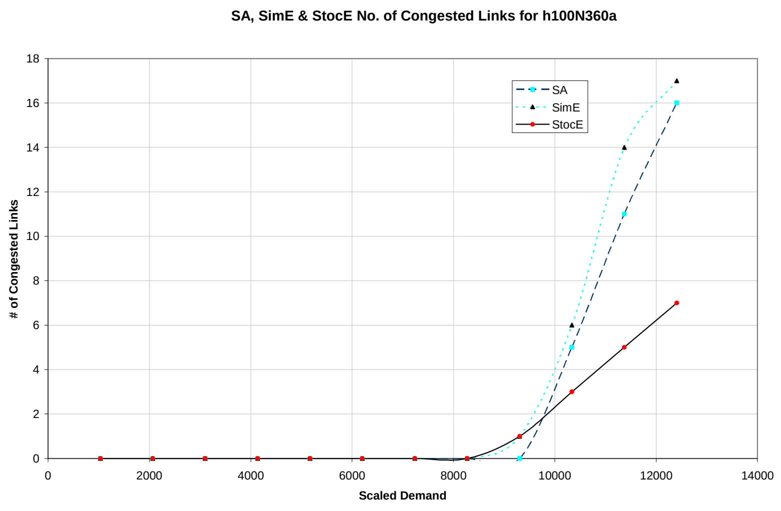


Figure 6.8: Number of congested links comparison of SA, SimE and StocE for h100N360a

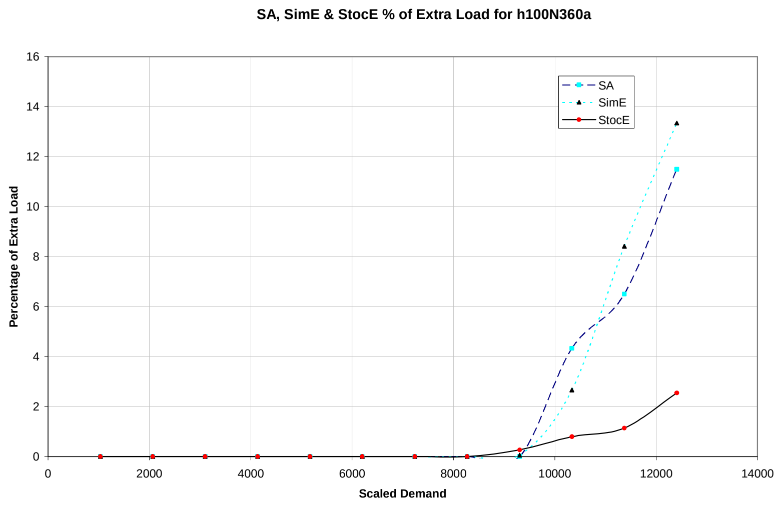


Figure 6.9: Percentage of Extra Load comparison of SA, SimE and StocE for h100N360a

6.4.4 Comparisons for r100N503a

This section compares the cost, maximum utilization, number of congested links and percentage of extra load for one of the largest random topologies, r100N503a. In this case, Fortz & Thorup's Linear Search outperforms the other algorithms. From the table 6.9, it can be observed that InvCap once again performs the worst. Next in the series of deteriorating performance are GA Eric, SA, StocE, SimE, and HGA Eric. The cost, maximum utilization, number of congested links and percentage of extra load comparisons are shown in Figures 6.10, 6.11, 6.12, and 6.13 respectively. This change in the trend could be attributed to the way the links are interconnected to nodes in this random graph.

Table 6.9: Cost Comparison for r100N503a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (FandT)**	**LPLB**	**SA**	**SimE**	**StocE**
8382.862	1	1.015	1.011	1	1	1.026	1.00	0.120
16765.72	1	1.023	1.011	1	1	1.037	1.02	0.15
25148.59	1.02	1.063	1.013	1	1	1.069	1.04	0.269
33531.45	1.09	1.172	1.041	1.011	1.01	1.173	1.12	1.14
41914.31	1.22	1.385	1.186	1.093	1.07	1.389	1.34	3.923
50297.17	1.4	1.698	1.358	1.236	1.2	1.626	1.41	6.139
58680.04	1.96	2.665	1.54	1.407	1.36	2.025	1.45	10.52
67062.9	8.13	4.89	1.773	1.605	1.54	2.497	1.67	18.932
75445.76	20.51	12.206	2.17	1.851	1.76	3.509	2.12	25.869
83828.68	48.85	30.794	2.788	2.286	2.1	8.503	3.13	34.968
92211.49	94.05	77.555	5.179	3.151	2.78	42.558	6.42	44.317
100594.3	**155.68**	**154.88**	**14.857**	**7.029**	**5.87**	**83.986**	**20.45**	**56.315**

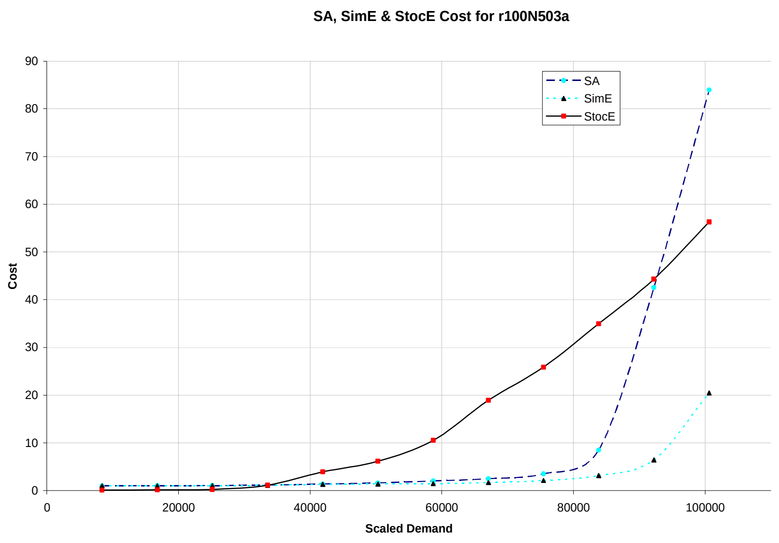


Figure 6.10: Cost comparison of SA, SimE and StocE for r100N503a

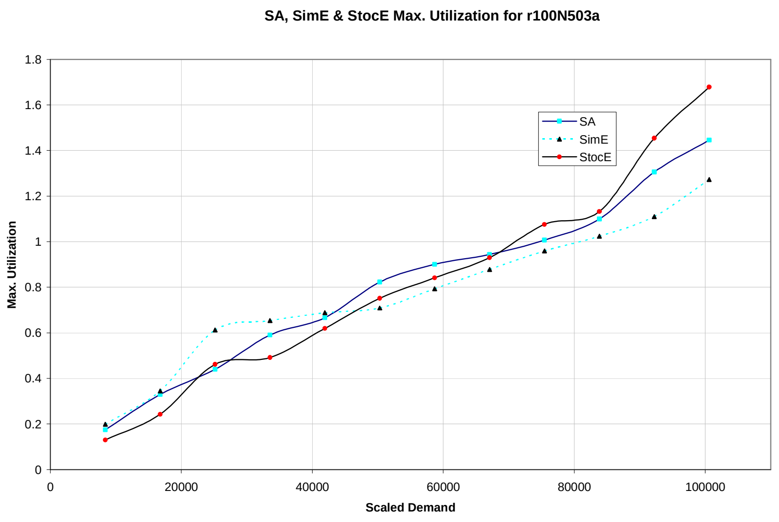


Figure 6.11: Maximum Utilization comparison of SA, SimE and StocE for r100N503a

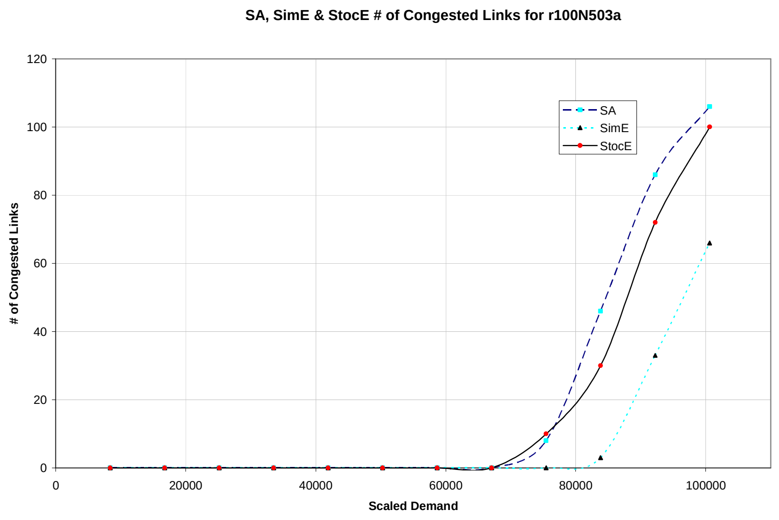


Figure 6.12: Number of congested links comparison of SA, SimE and StocE for r100N503a

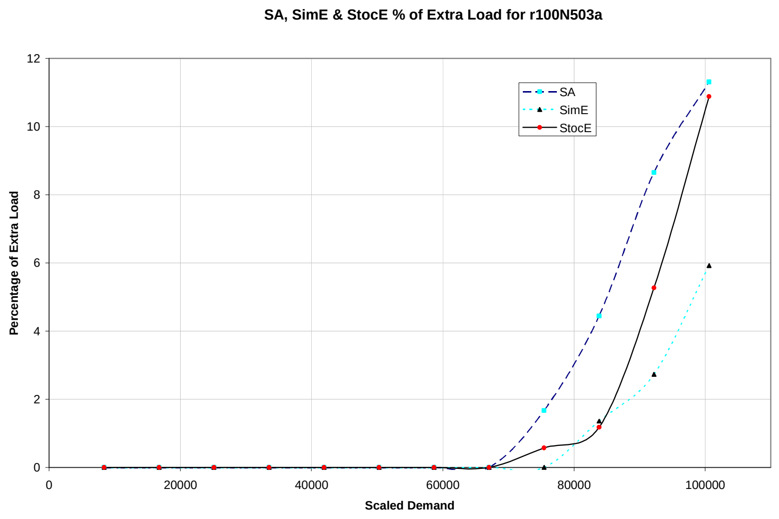


Figure 6.13: Percentage of Extra Load comparison of SA, SimE and StocE for r100N503a

6.5 Highest Demand Comparison

Table 6.10 portrays the comparison of all the algorithms for the highest demands used in this work with respect to StocE. The simulated results of SA, SimE, and StocE are hereby compared to the published results taking the highest demand (demand 12) into consideration. This result supports the core idea that in the majority of the cases, indeed, StocE is one of the best heuristics as it converges to the best cost for a given topology. In this case, the LSLB algorithm is considered to be the benchmark for comparing the remaining algorithms.

Table 6.10: Cost Comparison of all the Algorithms for the Highest Demand

Topologies	**Scaled Demand**	**InvCap**	**GAEric**	**HGAEric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
h50N148a	4927.689	180.299	20.968	15.123	16.837	14.4	23.01	37.15	**7.85436**
h50N212a	3362.693	159.848	12.318	4.221	**4.166**	**1.83**	13.49	47.07	4.48005
h100N280a	4605.21	114.55	19.73	19.214	19.245	19.06	20.777	27.81	**8.04038**
h100N360a	12406.54	181.1	23.401	15.169	18.547	**13.32**	20.21	24.03	**15.1025**
r50N228a	42281.17	264.611	27.375	**15.041**	15.585	**13.75**	32.35	53.15	22.7548
r50N245a	53561.54	357.045	123.604	50.631	53.301	**19.65**	151.2	154.75	**40.6343**
r100N403a	69296.84	238.56	108.485	25.283	**10.942**	**5.79**	69.05	54.14	47.7778
r100N503a	100594.3	155.68	154.88	14.857	**7.029**	**5.87**	84.23	20.45	56.3146
w50N169a	25411.46	139.521	13.633	12.849	12.883	**4.36**	14.56	16.35	**9.05255**
w50N230a	39446.6	78.084	11.588	11.288	11.281	**4.3**	11.92	12.41	**9.73927**
w100N391a	48473.73	54.56	12.647	11.575	**11.409**	**11.35**	12.07	12.40	14.7432
w100N476a	63493.11	103.43	26.675	20.213	**19.775**	**19.49**	24.35	22.35	25.5273

6.6 Results of the New Cost Function

As previously stated, the new cost function is designed to reduce the number of congested links especially for higher demands. Here, the same topology used for the comparison of Fortz's cost function, h100N360a, is considered for illustration. The new cost function has been designed with the main criteria of reducing the number of congested links and has the same impact on the algorithms considered. From the previous results, it has been deduced that in the majority of the cases, StocE outperforms the other heuristics when using the Fortz cost function. Similarly, with the new cost function, it has been deduced that StocE has an upper hand while achieving better results with a similar trade-off of larger computation time. However, the new cost function is designed to reduce the number of congested links at the expense of a marginal increase in maximum utilization. Figures 6.14, 6.15, and 6.16 represent the maximum utilization, number of congested links, and percentage of extra load comparisons of the Fortz's cost function and the new cost function for the various algorithms respectively.

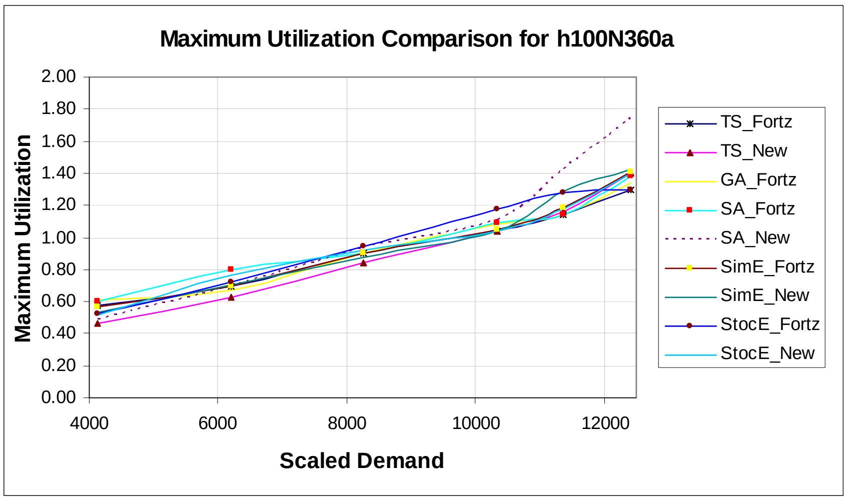


Figure 6.14: Maximum Utilization comparison of h100N360a using NewCF for various algorithms

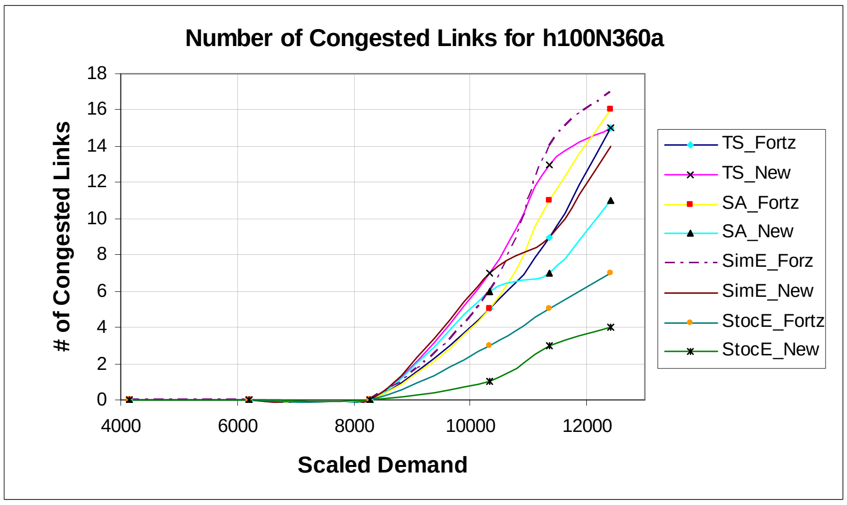


Figure 6.15: Number of congested links comparison of h100N360a using NewCF for various algorithms

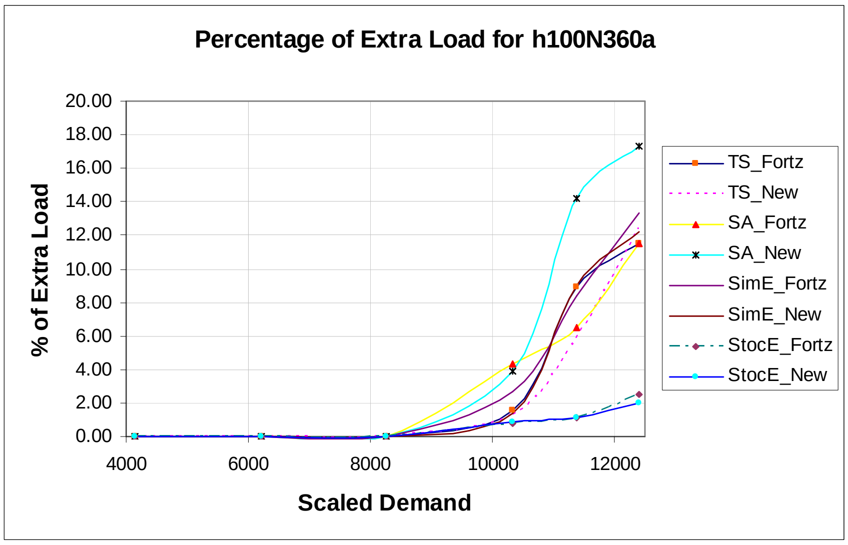


Figure 6.16: Percentage of Extra Load comparison of h100N360a using NewCF for various algorithms

Results of Tabu Search for Fortz's cost function and the new cost function are also compared with the other three, algorithms, i.e., SA, SimE, and StocE. The new cost function has been designed to minimize the number of congested links. This is reflected in the results presented in figure 6.15. The results of figures 6.14, 6.15, and 6.16 clearly support the logic behind the new cost function, i.e., the fewer number of congested links. This trend is seen in majority of the test cases considered.

Let us consider the topology h100N360a for the comparison of StocE's Fortz cost function and new cost function as depicted in figures 6.17 and 6.18. For the highest demand, demand 12, the maximum utilization for StocE Fortz cost function has increased from 1.3 to 1.4 for the StocE New cost function, and the number of congested links has reduced from 7 to 4 respectively.

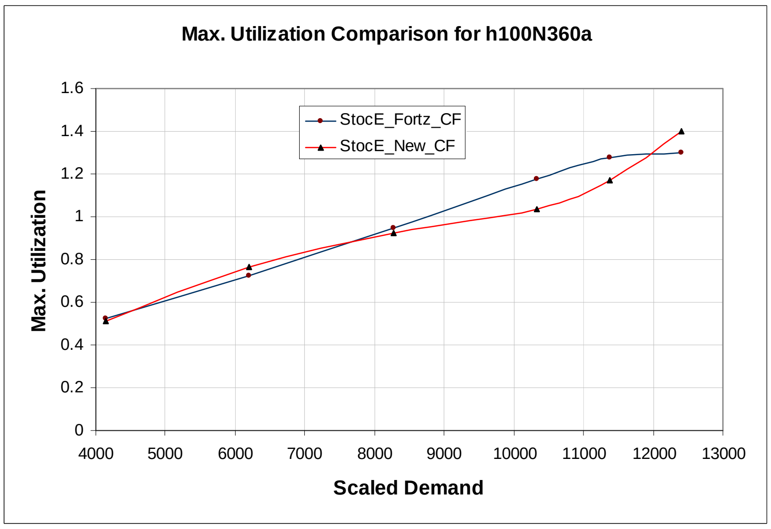


Figure 6.17: Maximum Utilization comparison of StocE Fortz Cost Function and the New Cost Function for h100N360a.

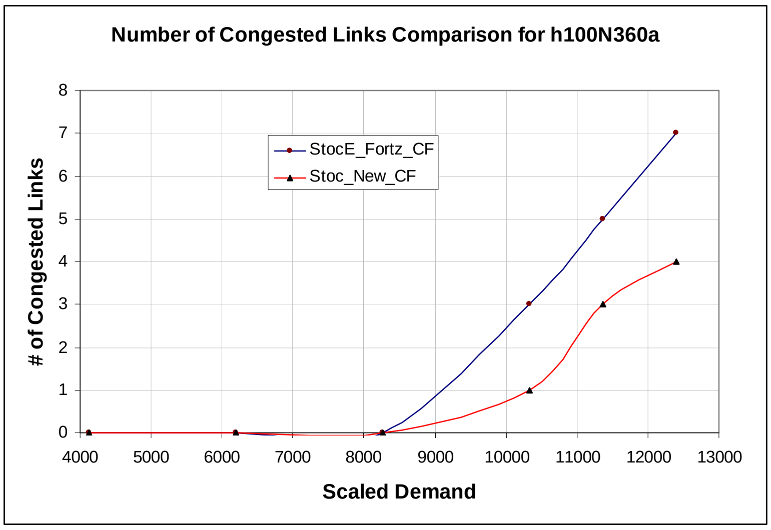


Figure 6.18: Number of congested links comparison of StocE Fortz Cost Function and the New Cost Function for h100N360a.

6.7 Overview of the Results of the Remaining Test Cases

Now, a brief overview of the other test cases, reported in appendix, for the four performance metrics, i.e., the cost, maximum utilization, percentage of extra load and number of congested links is presented. In h50N148a, h100N280a, r50N245a, w50N169a and w50N230a, the cost obtained through StocE has outperformed the other heuristics. In the case of h50N212a, the cost through StocE has outperformed Cisco's InvCap, Ericsson's GA (GA Eric), SA, and SimE. StocE's cost is comparable for other heuristics. In r50N228a, and r100N403a topology, StocE cost is better than InvCap, GA Eric, SA, and SimE, whereas HGA Eric and Linear Search were better than StocE. In w100N391a topology, StocE performed better than InvCap whereas the other heuristics did marginally better than StocE. In w100N476a test case, the LS algorithm outperforms the remaining algorithms. A clear insight on the remaining algorithms shows that there is a marginal variation in the cost values. As discussed above, the maximum utilization, the number of congested links, and the percentage of extra load follow a similar trend as their corresponding costs.

6.8 Phase II - Results and Discussion on Link-Failure

Here, the second phase of the thesis work is introduced. In this section, a comparison of Strategy1 (Strat1), Strategy2 (Strat2) and Strategy Original Heuristic (StratOH) are picturesquely represented and an approach of analyzing them is described. The details of these strategies are described in Chapter 4. The 12 test cases considered in phase I are reused in phase II.

In the case of without-link-failure (N+1), StratOH provides the best weight assignment because it is optimized for the (N+1) topology. Hence, the other two strategies, Strat1 and Strat2, are compared to StratOH. Initially, Strat1 is compared with StratOH for both with-link and without-link-failure. In the case of without-link-failures, the cost obtained in Strat1 is generally higher (i.e., worse) than that obtained in StratOH. Similarly, on comparing Strat2 with StratOH, 90% of the times of the given cases, the latter has comparatively lower cost than the former in the case of without-link-failures.

In case of a link-failure, Strat2 outperforms the other two strategies since the optimized cost is obtained for the topology with a link failure. In this case, Strat2 outperforms Strat1 and StratOH as predicted except for a few cases.

The following subsections portray the cost obtained with the help of these strategies and a comparison of the results is made. The same naming conventions as described in Chapter 4 is maintained throughout. For the sake of representation, higher demands such as demand 8 (d8) → demand 12 (d12) are considered. The reason for not representing the lower demands, demand 1→ demand 7 was that they had minimal impact on the results. The second column in the tables 6.11 → A.20 represent the scaled demand. The third and fourth columns represent strategies without-link-failure, whereas the sixth and seventh columns represent the same strategies with link-failure respectively. The fifth column (N+1) represents the cost increase (i.e., loss) while implementing a particular strategy with regard to no-link failures, in comparison to the benchmark strategy considered. The eighth column (N) similarly represents the cost gained (i.e., decrease) while implementing a particular strategy in terms of link failures in comparison to the benchmark strategy considered. The final column *(Loss:Gain)%*, the absolute value of the ratio of (N+1) to (N) times 100, gives an overall picture of the loss or gain obtained on implementing a particular strategy. When both (N+1) and (N) have positive values, there is an overall gain and when both (N+1) and (N) have negative values, then the topology is considered to have an overall loss. A low value here shows that the loss incurred in the case of a no-link-failure for a particular strategy is much smaller than the gain obtained on implementing the same strategy in the case of a link-failure. The bottom line is that through the last column, one can interpret the pros and cons of implementing a particular strategy.

Figures 6.19 → 6.22 give a pictorial representation of the topologies considered, both, in terms of without-link-failure (normal) as well as in terms of link-failure (failure). This clearly depicts that the usage of a particular strategy determines the overall loss or gain obtained.

6.8.1 Link Failure Comparison for h100N360a

Let us consider the hierarchical topology, h100N360a. This is one of the largest topologies of these kinds of graphs.

Table 6.11: Cost Comparison for h100N360a Strat1 & StratOH for with link and without link failure

	Topology	**Strat 1**	**Strat OH**	**(N+1)**	**Strat 1**	**Strat OH**	**N**		
	h100N360a	**(N+1)avg**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)avg**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	8271.03	1.54	1.36	-0.18	3.15	3.96	0.81	0.63	22.22
d9	9304.91	3.90	3.41	-0.49	5.48	6.85	1.38	0.89	35.51
d10	10338.79	6.26	5.68	-0.58	7.90	9.61	1.71	1.13	33.92
d11	11372.67	9.42	8.31	-1.11	13.33	15.52	2.19	1.08	50.68
d12	**12406.54**	**14.94**	**13.58**	**-1.36**	**15.87**	**18.10**	**2.24**	**0.88**	**60.71**

Table 6.11 represents the cost when strategy 1 (Strat1) is compared to the original heuristic strategy (StratOH). It can be seen that StratOH is performing better when there is no link-failure since this topology is optimized for the topology with no link-failure. The column [*G(+)/L(-)*] for (N+1) links shows the gain or the loss when Strat1 is implemented. However, in the case of link failure, Strat1 outperforms StratOH since Strat1 considers the average optimized cost as discussed in chapter 4. The column [*G(+)/L(-)*] for (N) links shows the gain or loss obtained while implementing Strat1 in comparison with StratOH. The column *Total Gain* gives the overall gain obtained by implementing strategy 1. The last column *(L:G)%* shows

the ratio of loss to gain obtained during the implementation of Strat1. In case of demand12, there is an overall loss to gain of 60.71%. This means that the loss incurred for using Strat1 instead of StratOH in the case of no link-failure represents only 60.71% of the gain obtained by Strat1 against StratOH in the case of a link-failure. In other terms, Srat1 provides better results in the case of a link-failure at the expense of some loss in the case of no-link-failure.

Table 6.12 represents the cost when strategy 2 (Strat2) is compared to the original heuristic strategy (StratOH). Even here, similar results are obtained in the case

Table 6.12: Cost Comparison for h100N360a Strat2 & StratOH for with link and without link failure

	Strat 2	**Strat OH**	**(N+1)**	**Strat 2**	**Strat OH**	**N**		
	(N)OH + Link	**(N+1)OH**	**[G(+)/L(-)]**	**(N)OH**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	2.12	1.36	-0.76	2.04	3.96	1.92	1.16	39.58
d9	3.40	3.41	0.01	4.91	6.85	1.94	1.95	G
d10	6.96	5.68	-1.28	7.32	9.61	2.28	1.00	56.14
d11	10.30	8.31	-1.98	11.27	15.52	4.25	2.27	46.59
d12	**15.92**	**13.58**	**-2.33**	**12.18**	**18.10**	**5.93**	3.60	**39.29**

of without as well as with a link failure when compared to StratOH. However, in the case of link failures, the ratio of loss to gain percentage, i.e., 39.39% obtained for the highest demand, demand 12, during the implementation of Strat 2 suggests that the loss here is less compared to Strat1. This point is well supported when comparing Strat1 and Strat2 as shown in table 6.13.

Figure 6.19 shows the comparison of Strat1 and Strat2 with StratOH in the case of no-link-failure. Generally, StratOH outperforms the other two strategies. But when comparing these two strategies, except for demand9, Strat1 does better than

Table 6.13: Cost Comparison for h100N360a Strat1 & Strat2 for with link and without link failure

	Topology	**Strat 1**	**Strat 2**	**(N+1)**	**Strat 1**	**Strat 2**	**N**		
	h100N360a	**(N+1)avg**	**(N)OH + Link**	**[G(+)/L(-)]**	**(N)avg**	**(N)OH**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	8271.03	1.54	2.12	0.58	3.15	2.04	1.10	1.68	G
d9	9304.91	3.90	3.40	-0.50	5.48	4.91	0.56	0.06	89.28
d10	10338.79	6.26	6.96	0.70	7.90	7.32	0.58	-0.12	L
d11	11372.67	9.42	10.30	0.87	13.33	11.27	2.06	1.19	G
d12	**12406.54**	**14.94**	**15.92**	**0.98**	**15.87**	**12.18**	**3.69**	**2.71**	**G**

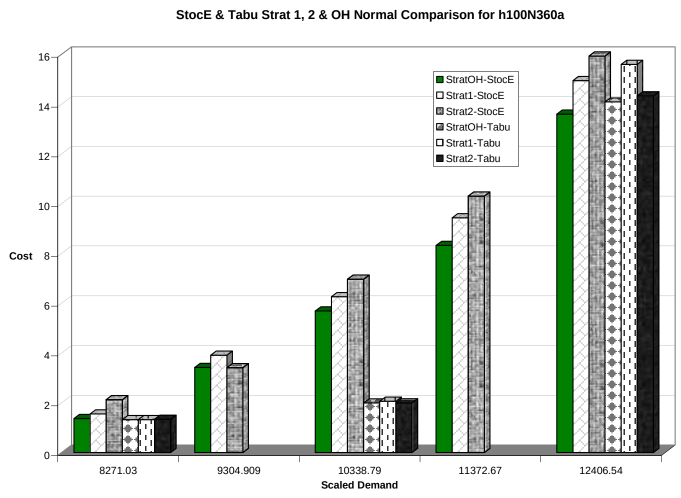


Figure 6.19: StocE & Tabu Cost Comparison of StratOH, Strat1 and Strat2 for h100N360a for normal case

Strat2. Figure 6.20 shows the comparison of Strat1 and Strat2 with StratOH in the case of a link failure. In this case, Strat2 outperforms both StratOH and Strat1. This figure rightly supports the implementation of Strat2 where a complete optimization of the weights is done for the links and only a single arc, i.e., the assumed failed link, has weights 1→20 and the best possible one is selected as described in detail in the Chapter 4.

A similar comparison is made with Tabu Search and the results are picturesquely illustrated in the figures 6.19 and 6.20. Since the stopping criteria is different

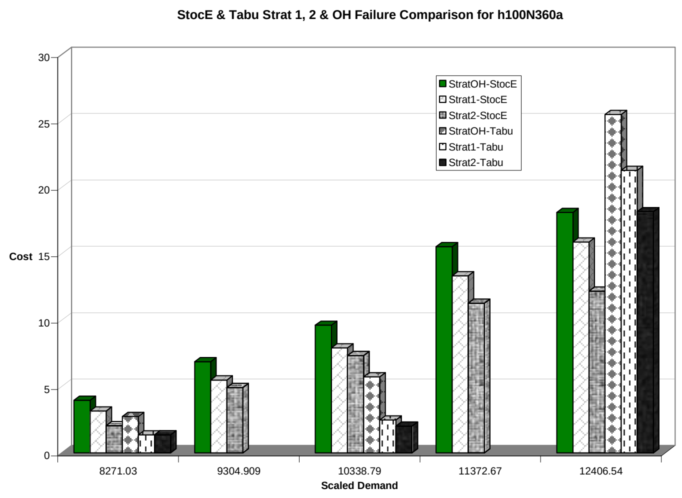


Figure 6.20: StocE & Tabu Cost Comparison of StratOH, Strat1 and Strat2 for h100N360a for Link Failure case

for both StocE and Tabu Search, here prominence is given to the comparison of the trends. StocE performs better compared to Tabu Search. For example, the difference between the cost obtained for StratOH, Strat1 and Strat2 through StocE is better compared to the difference obtained through Tabu Search. This illustrates that the gain obtained by implementing the two strategies in the case of link-failures is more in StocE compared to Tabu Search.

In the normal state, i.e., (N+1) topology, StratOH performs better than Strat1, and Strat1 is better than Strat2. In the failure state, Strat2 outperforms Strat1 and StratOH. In fact, Strat2 has a considerable gain in the case of link-failure and comparatively smaller loss in the case of topologies without-link-failures. In all the test cases considered, shown in figures 6.19, 6.20, 6.21, 6.22, A.50, and A.51, similar trends are found. In the case of topologies with lower demands, there is hardly any impact as the results are almost similar. This shows that there is a minimum effect on the network performance if load is kept low. Hence, in such cases, the original heuristic itself can be used.

6.8.2 Link Failure Comparison for h50N148a

Tables 6.14, 6.15, and 6.16 represent the comparisons among the three strategies and figures 6.21 and 6.22 show the corresponding normal and failure states performance for h50N148a topology respectively.

Table 6.14: Cost Comparison for h50N148a Strat1 & StratOH for with link and without link failure

	Topology	**Strat 1**	**Strat OH**	**(N+1)**	**Strat 1**	**Strat OH**	**N**		
	h50N148a	**(N+1)avg**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)avg**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	3285.13	0.44	0.26	-0.17	0.44	0.53	0.09	-0.08	188.00
d9	3695.77	0.96	0.75	-0.22	1.17	1.00	-0.17	-0.39	L
d10	4106.41	2.63	2.35	-0.28	3.54	4.43	0.89	0.61	96.55
d11	4517.05	4.57	4.41	-0.16	4.76	6.32	1.56	0.40	10.26
d12	**4927.69**	**7.85**	**7.76**	**-0.08**	**8.88**	**10.85**	**1.98**	**1.90**	**4.04**

Table 6.15: Cost Comparison for h50N148a Strat2 & StratOH for with link and without link failure

	Strat 2	**Strat OH**	**(N+1)**	**Strat 2**	**Strat OH**	**N**		
	(N)OH + Link	**(N+1)OH**	**[G(+)/L(-)]**	**(N)OH**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	0.35	0.26	-0.09	0.39	0.53	0.14	0.05	64.29
d9	0.95	0.75	-0.20	0.55	1.00	0.45	0.25	44.44
d10	3.21	2.35	-0.86	2.94	4.43	1.49	0.63	57.72
d11	4.67	4.41	-0.26	3.67	6.32	2.65	2.39	9.81
d12	**7.99**	**7.76**	**-0.22**	**6.83**	**10.85**	**4.02**	**3.80**	**5.47**

Table 6.16: Cost Comparison for h50N148a Strat1 & Strat2 for with link and without link failure

	Topology	**Strat 1**	**Strat 2**	**(N+1)**	**Strat 1**	**Strat 2**	**N**		
	h50N148a	**(N+1)avg**	**(N)OH + Link**	**[G(+)/L(-)]**	**(N)avg**	**(N)OH**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	3285.13	0.44	0.35	-0.09	0.44	0.39	0.05	-0.04	180.00
d9	3695.77	0.96	0.95	-0.02	1.17	0.55	0.62	0.60	3.26
d10	4106.41	2.63	3.21	0.58	3.54	2.94	0.60	1.18	G
d11	4517.05	4.57	4.67	0.10	4.76	3.67	1.08	1.18	G
d12	**4927.69**	**7.85**	**7.99**	**0.14**	**8.88**	**6.83**	**2.04**	**2.18**	**G**

Similarly, tables A.18, A.19, and A.20 represent the comparisons between the three strategies and figures A.50 and A.51 show the corresponding normal and failure states performance for r100N503a topology respectively.

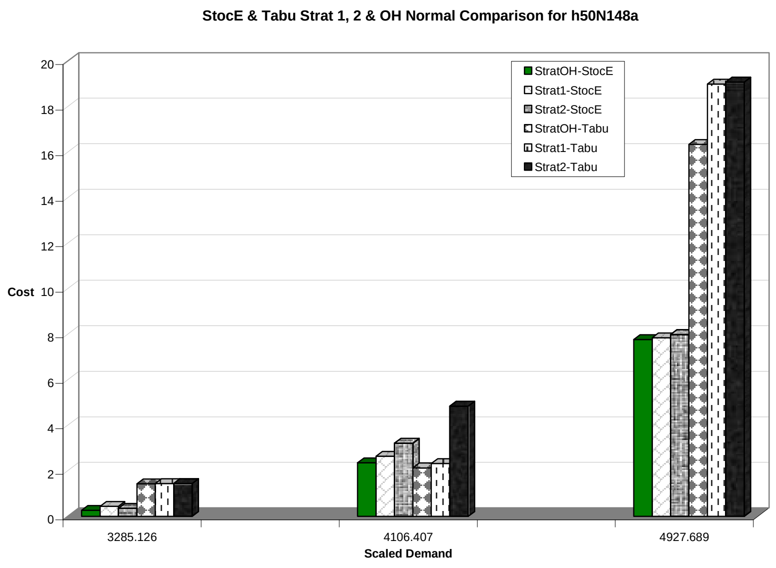


Figure 6.21: StocE & Tabu Cost Comparison of StratOH, Strat1 and Strat2 for h50N148a for normal case

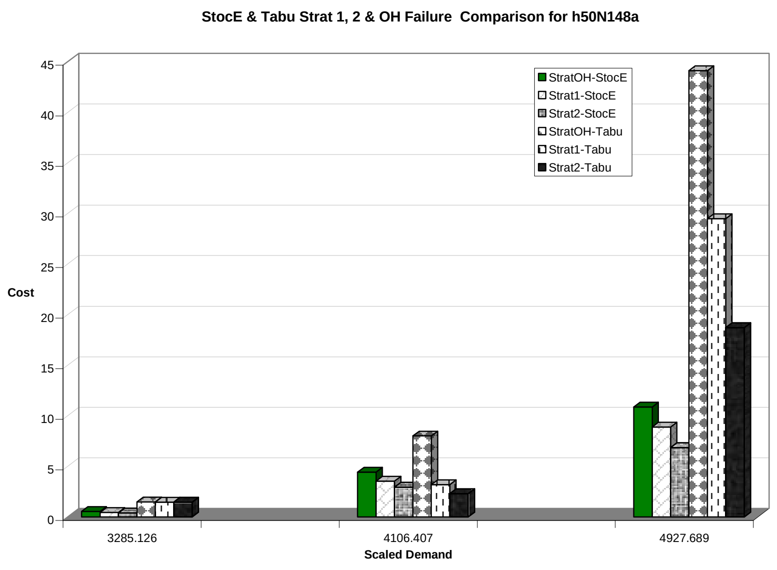


Figure 6.22: StocE & Tabu Cost Comparison of StratOH, Strat1 and Strat2 for h50N148a for Link Failure case

Chapter 7

Conclusion and Future Work

7.1 Conclusions

The efficient operation of the network depends on the configuration of the individual routers. The importance of the contribution of Traffic Engineering (TE) and the challenges the TE needs to cope with for a successful internetworking is described in this work. The TE problem has been modeled as a multi-commodity flow optimization (MCF) problem. The routing protocol classification is made, of which, OSPF, an example of IGP, forms the core of this thesis work. The ISP backbone network, its architecture, interfaces and links are briefly discussed, and a detailed description is presented in Appendix B.

An introduction to combinatorial optimization is illustrated and an overview of the iterative heuristics is discussed. In this thesis, an NP-hard problem has been addressed by using a Stochastic Evolution (StocE) algorithm that mimics the metaphor of natural biological evolution is adapted to the OSPFWS problem. A generic outline of StocE algorithm is portrayed with a clear explanation of its core modules such as *Perturb*, *Update* and *Make-State*. An implementation of StocE and its comparison to other widely known algorithms such as Simulated Annealing, Simulated Evolution, Genetic Algorithm, etc. are illustrated explicitly along with the convergence aspects.

A total of four parameters such as cost, maximum utilization, number of congested, and percentage of extra load were compared on twelve experimental network topologies. Throughout the simulations, in the majority of the cases, the cost obtained by StocE was better compared to other heuristics. This can be apprehended better from the results on the highest demand comparison for all the twelve networks considered. StocE outperformed SA and SimE, the other two algorithms resimulated alongside. Though, StocE can be used for large networks, it has a serious drawback. Compared with most approximate algorithms, StocE takes very large amount of CPU time especially for large problems as discussed in this thesis.

Another core issue addressed in this work is the improvement of the network by considering single link failure scenarios. In this thesis, we have proposed two strategies for OSPF/IS-IS networks. To the best of our knowledge, this is an innovative approach. Here, no-weight changes are propagated during link failures and the same weights that gave the optimal solution without a link failure are reused for the case of the link failure to get to an optimal or close to optimal solution. Simulation results show that if we assume a single link failure at a time, which accounts for large portion of network failures, then our strategies perform better in terms of cost for the majority of network topologies used in this thesis. One of the main obstacles to the first strategy discussed is the need for a great amount of computing power as each state enumerated would require a complete run of the simulation. This discouraging factor turned out to be a motivation for the implementation of the second strategy where initially, a simple computation of the shortest paths and resulting flows have alleviated the computation.

7.2 Future Work

By the problem specific initial solution, the size of the search space can be reduced by a great amount. However, to deal with cyclopean problems, the processing time has to be reduced drastically. Future work may consider other link and node failure models. Precomputing back-up paths that minimize the capacity requirements for multiple link failures is an interesting direction of study. AntNet, result of the application of Ant Colony Optimization (ACO) on the problem of Internet routing, has been implemented on smaller networks. It is an alternative routing algorithm for the OSPF protocol [75]. An inquisitiveness could be at the other end if this algorithm can be implemented for huge test cases as those used in this thesis.

Appendix A

Results of Remaining Test Cases

A.1 Iteration Comparison

Fig A.1, Fig A.2 and Fig A.3 represent the iteration comparisons for two topologies with different demands. Through these figures, it has been confirmed that $R = 20, P_{incr} = 3, and P_{zero} = 0$ is the best parameter combination

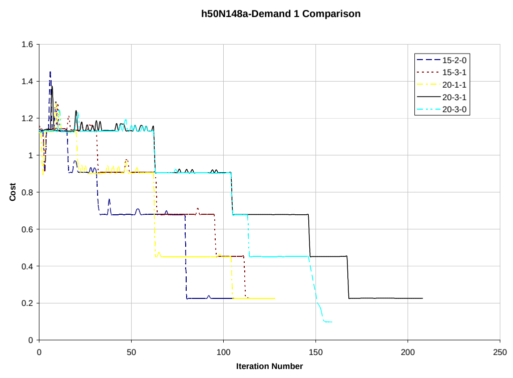


Figure A.1: Iteration comparison for h50N148a demand1 showing $R = 20, P_{incr} = 3, and P_{zero} = 0$ as the best combination

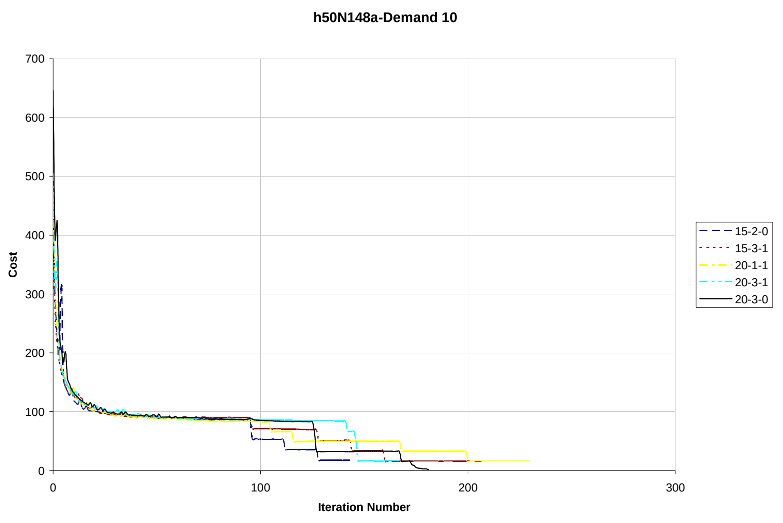


Figure A.2: Iteration comparison for h50N148a demand10 showing $R = 20, P_{incr} = 3, and P_{zero} = 0$ as the best combination

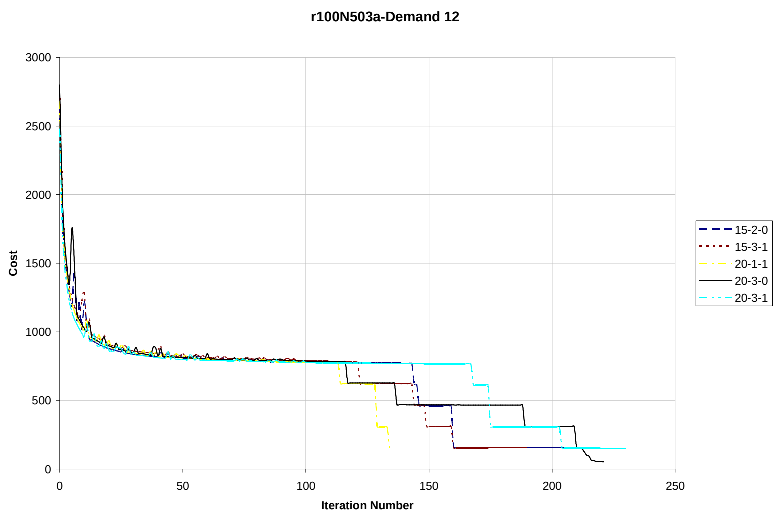


Figure A.3: Iteration comparison for r100N503a demand12 showing $R = 20, P_{incr} = 3, and P_{zero} = 0$ as the best combination

A.2 Time Comparison

As shown in the figures A.4 and A.5, a time comparison is made amongst StocE, SA and SimE for demands 4 and 12. Here the current cost is compared instead of best cost.

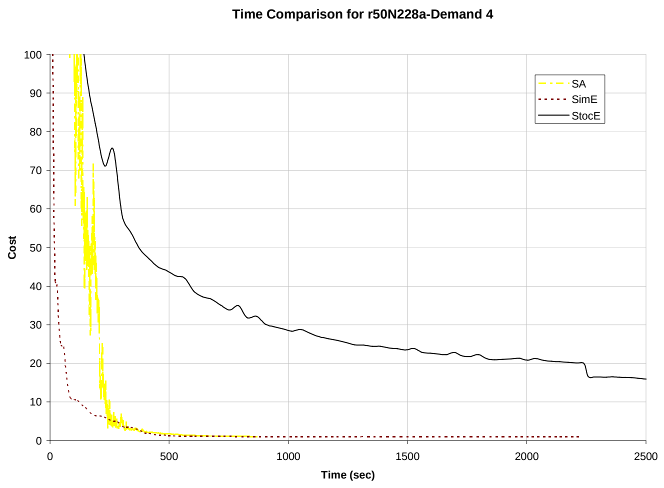


Figure A.4: Time comparison of SA, SimE and StocE for r50N228a demand4

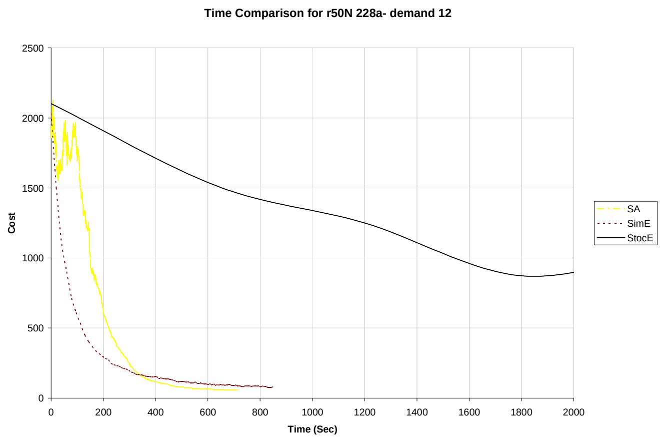


Figure A.5: Time comparison of SA, SimE and StocE for r50N228a demand12

A.3 Topologies Comparison

A.3.1 Comparisons for w100N476a

In this case, a comparison of cost, maximum utilization, number of congested links and percentage of extra load is considered for a waxman graph with 100 nodes and 476 arcs, largest waxman topology available. An interesting observation that can be made from table A.1 is that the StocE cost is marginally comparable to SA and SimE. Figures A.6, A.7, A.8, and A.9 portrays the comparison of cost, utilization, number of congested links and percentage of extra load on the links, respectively.

Table A.1: Cost Comparison for w100N476a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
5291.092	1	1.011	1.011	1	1	1.0183	1.001	0.1161
10582.19	1	1.012	1.011	1	1	1.0251	1.015	0.1265
15873.28	1.01	1.022	1.011	1.001	1	1.0351	1.034	0.1515
21164.37	1.06	1.064	1.031	1.015	1.02	1.0718	1.065	0.1792
26455.46	1.13	1.138	1.064	1.036	1.03	1.1541	1.145	0.3044
31746.55	1.25	1.259	1.169	1.105	1.09	1.2836	1.210	1.2492
37037.65	1.49	1.458	1.313	1.218	1.19	1.4624	1.243	2.7000
42328.74	3.38	1.679	1.468	1.354	1.32	1.6777	1.396	5.6383
47619.83	18.08	2.224	1.732	1.596	1.54	1.9988	1.685	10.3860
52910.92	38.94	3.832	2.682	2.53	2.41	3.3075	2.975	13.6403
58202.02	69.4	12.132	9.998	10.436	9.62	11.5609	11.363	18.6104
63493.11	**103.43**	**26.675**	**20.213**	**19.775**	**19.49**	**21.2365**	**22.327**	**25.5273**

A.3.2 Comparisons for r50N228a

This section considers a comparatively smaller random topology, r50N228a and compares the cost, maximum utilization, number of congested links and percentage of extra load. As usual InvCap performs the worst and StocE, best of SA and SimE

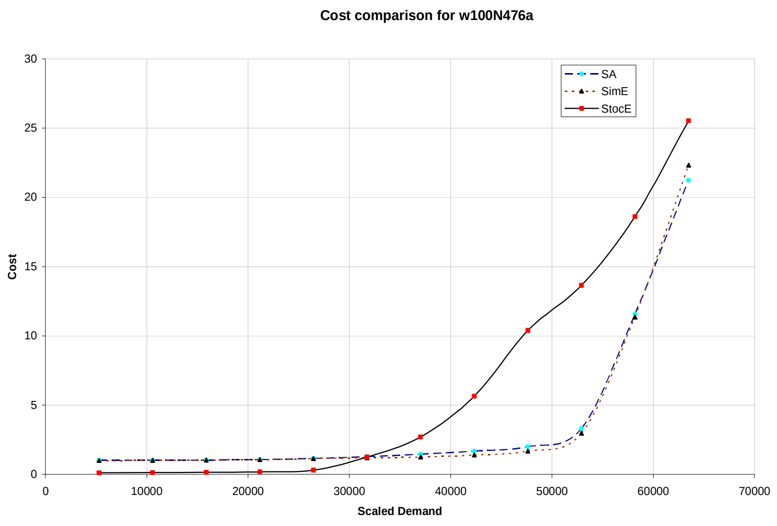


Figure A.6: Cost comparison of SA, SimE and StocE for w100N476a

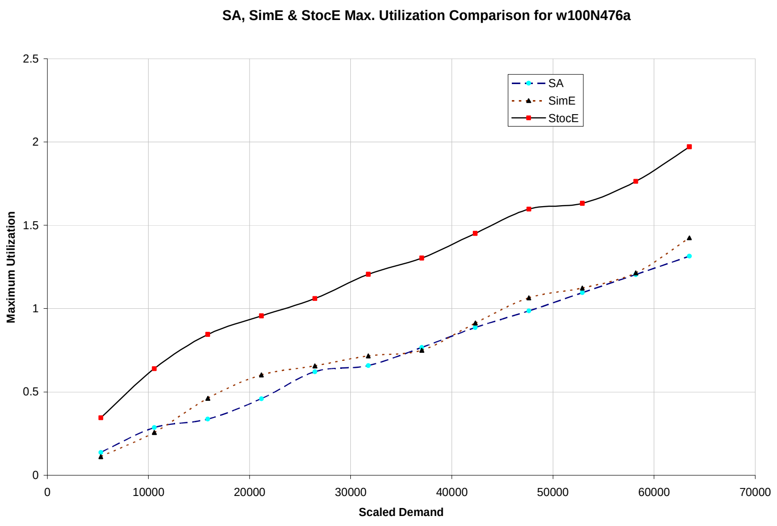


Figure A.7: Maximum Utilization comparison of SA, SimE and StocE for w100N476a

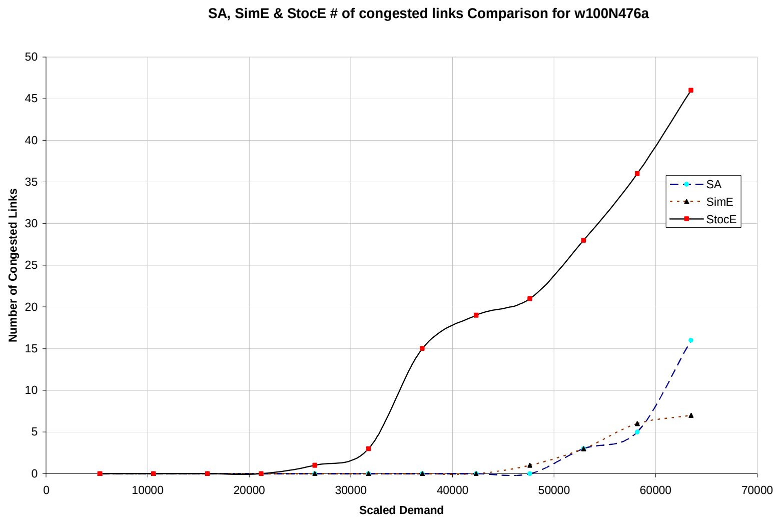


Figure A.8: Number of congested links comparison of SA, SimE and StocE for w100N476a

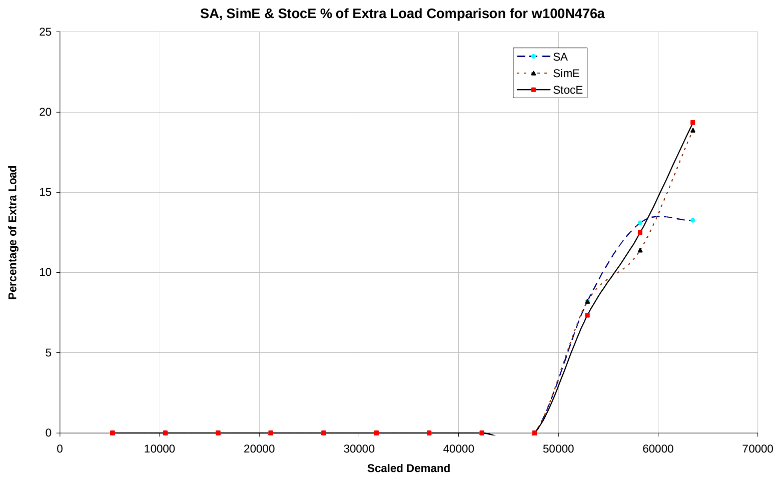


Figure A.9: Percentage of Extra Load comparison of SA, SimE and StocE for w100N476a

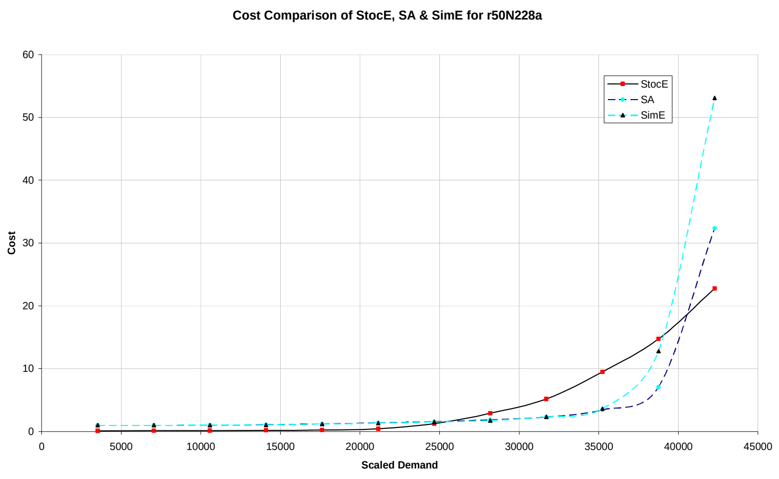


Figure A.10: Cost comparison of SA, SimE and StocE for r50N228a

as shown in table A.2. Even in this case, the utilization, number of congested links and percentage of extra load follows the similar methodology as was discussed in hierarchical topologies. These parameter comparisons are shown in figure A.10, representing the comparison of costs, where figures A.11, A.12, and A.13 representing the comparison of maximum utilization, number of congested links, and percentage of extra load, respectively.

Table A.2: Cost Comparison for r50N228a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F andT)**	**LPLB**	**SA**	**SimE**	**StocE**
3523.431	1	1	1	1	1	1.0004	1.0001	0.1109
7046.862	1	1	1	1	1	1.0008	1.0007	0.1224
10570.29	1.043	1.002	1.001	1.001	1	1.007	1.005	0.1470
14093.72	1.136	1.056	1.036	1.036	1.03	1.0771	1.04	0.1804
17617.15	1.296	1.179	1.151	1.144	1.13	1.2078	1.21	0.2340
21140.58	1.568	1.327	1.292	1.286	1.27	1.3715	1.3234	0.4473
24664.02	3.647	1.525	1.455	1.447	1.42	1.5691	1.5135	1.2531
28187.45	27.352	1.78	1.672	1.672	1.61	1.8623	1.7459	2.8980
31710.88	66.667	2.173	1.977	1.976	1.9	2.2932	2.3555	5.1675
35234.31	122.869	3.201	2.556	2.569	2.43	3.4546	3.6422	9.4956
38757.74	188.778	7.738	4.607	4.683	4.26	7.0602	12.7926	14.7259
42281.17	**264.611**	**27.375**	**15.041**	**15.585**	**13.75**	**32.3553**	**53.0953**	**22.7548**

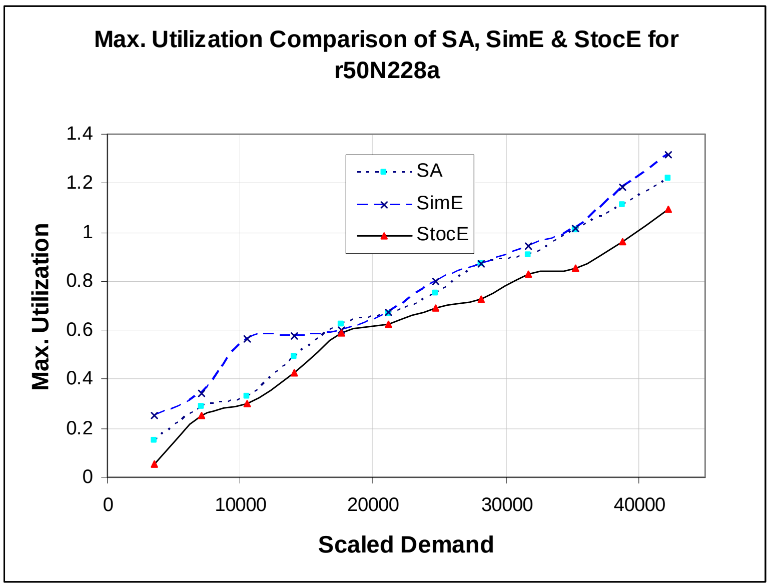


Figure A.11: Maximum Utilization comparison of SA, SimE and StocE for r50N228a

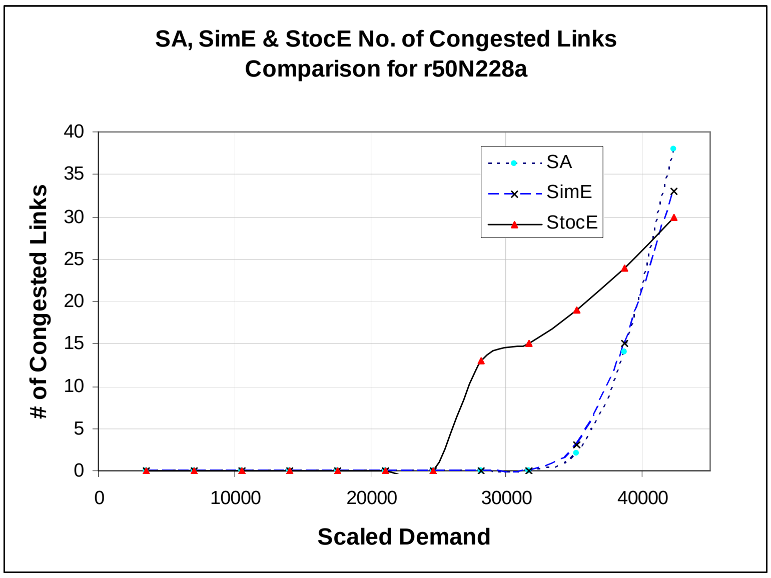


Figure A.12: Number of congested links comparison of SA, SimE and StocE for r50N228a

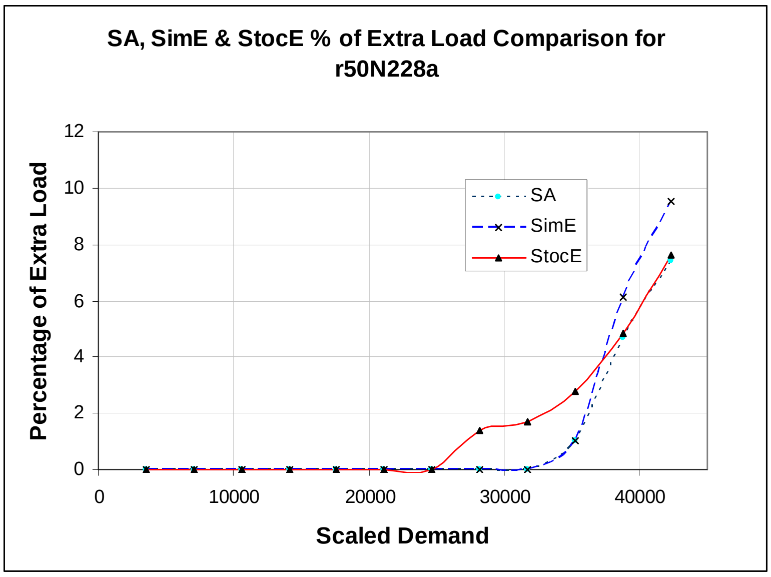


Figure A.13: Percentage of Extra Load comparison of SA, SimE and StocE for r50N228a

A.3.3 Comparisons for w50N169a

Here, a comparison of cost, maximum utilization, number of congested links and percentage of extra load is considered for a waxman graph with 50 nodes and 169 arcs. Table A.3 clearly illustrates that, as usual, InvCap performs the worst. On comparing with remaining algorithms, figure A.14, even in this case, StocE's best cost is the best amongst the others. The utilization of the links comparison, figure A.15, shows that for higher demands, StocE lies between SA and SimE, a point worth considering in this topology. Similar comparisons for number of congested links and percentage of extra load can be observed from figures A.16 and A.17, respectively.

Table A.3: Cost Comparison for w50N169a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
2117.622	1	1	1	1	1	0.9999	0.9994	0.1033
4235.244	1	1	1	1	1	0.9999	0.9992	0.105
6352.865	1.015	1	1	1	1	0.3332	1.0023	0.1163
8470.487	1.08	1.023	1.018	1.017	1.02	0.4200	1.014	0.1292
10588.11	1.171	1.1	1.088	1.086	1.08	1.1101	1.15	0.1465
12705.73	1.316	1.205	1.188	1.183	1.18	1.2176	1.2344	0.1793
14823.35	1.605	1.339	1.315	1.309	1.29	1.3514	1.3293	0.2536
16940.97	3.899	1.531	1.483	1.475	1.45	1.5505	1.5130	0.358
19058.6	20.63	1.798	1.756	1.754	1.72	1.8519	1.8163	1.0408
21176.22	45.769	2.498	2.373	2.361	2.31	2.5121	2.5555	2.1472
23293.84	84.409	6.28	5.988	5.998	3.27	6.3000	7.2413	4.6645
25411.46	**139.521**	**13.633**	**12.849**	**12.883**	**4.36**	**14.5527**	**16.3181**	**9.0525**

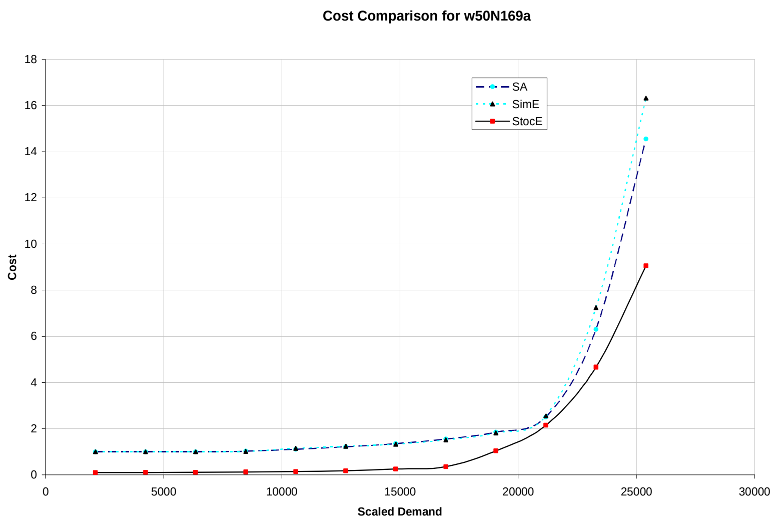


Figure A.14: Cost comparison of SA, SimE and StocE for w50N169a

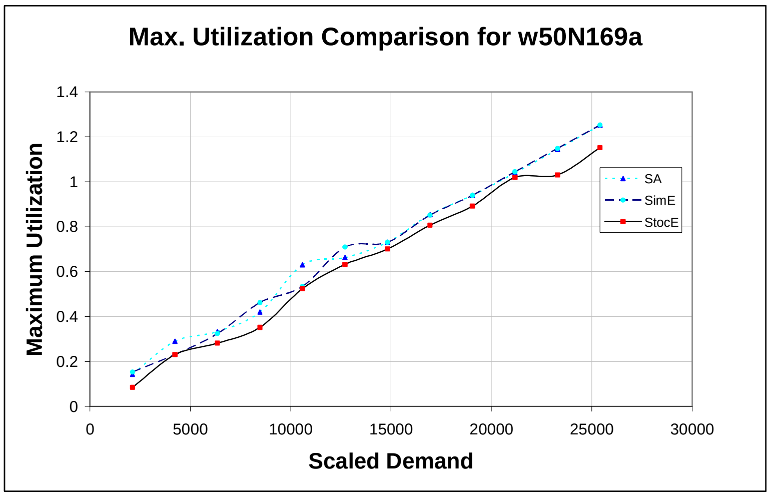


Figure A.15: Maximum Utilization comparison of SA, SimE and StocE for w50N169a

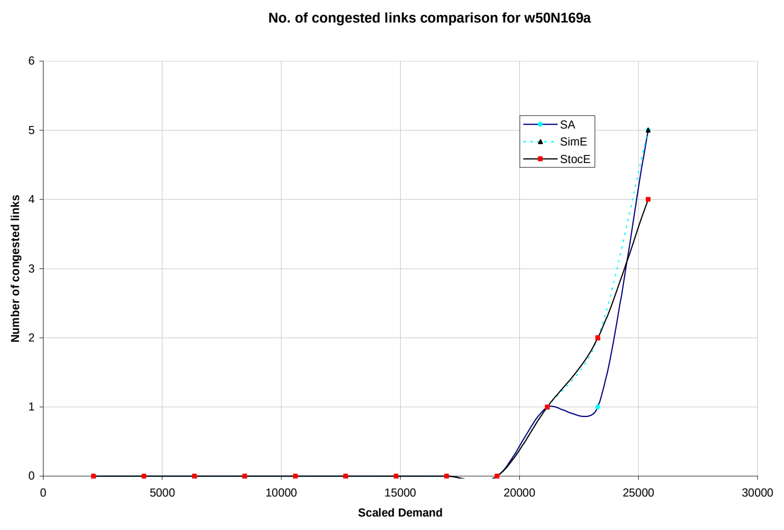


Figure A.16: Number of congested links comparison of SA, SimE and StocE for w50N169a

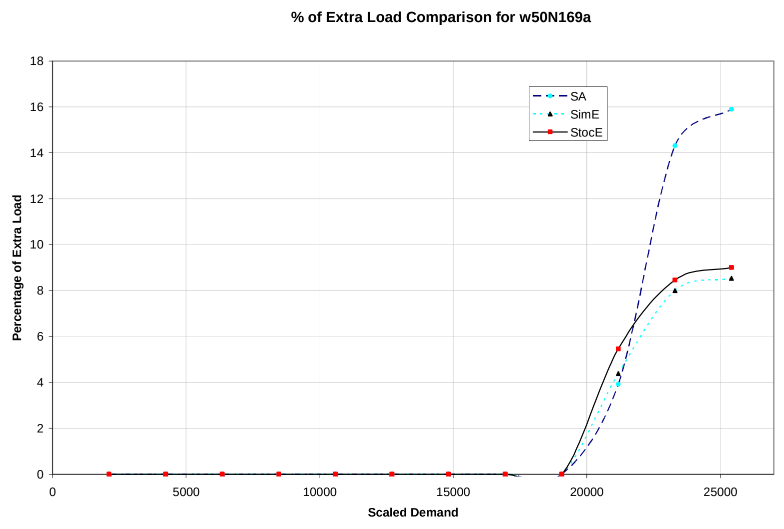


Figure A.17: Percentage of Extra Load comparison of SA, SimE and StocE for w50N169a

A.3.4 Comparisons for h50N212a

This section illustrates the cost, maximum utilization, number of congested links and percentage of extra load comparisons for the hierarchical topology, h50N212a. Table A.4 shows the cost comparison

Table A.4: Cost Comparison for h50N212a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
280.2244	1.005	1	1	1	1	0.9999	0.9999	0.1036
560.4488	1.012	1.001	1.001	1.001	1	1.0014	1.021	0.1021
840.6732	1.039	1.018	1.017	1.017	1.01	1.0184	1.0342	0.1094
1120.898	1.11	1.058	1.054	1.054	1.03	1.0589	1.0734	0.1218
1401.122	1.268	1.098	1.092	1.092	1.06	1.1043	1.1234	0.1401
1681.346	6.281	1.146	1.137	1.137	1.11	1.1458	1.2097	0.1790
1961.571	27.661	1.227	1.208	1.206	1.16	1.2295	1.2674	0.1848
2241.795	44.14	1.352	1.319	1.316	1.24	1.3628	1.4812	0.2435
2522.02	63.905	1.52	1.453	1.452	1.35	1.5129	1.7879	0.3689
2802.244	95.131	1.875	1.718	1.691	1.47	1.891	2.3828	1.0394
3082.468	128.351	3.153	2.264	2.205	1.61	3.2177	7.0065	3.069
3362.693	**159.848**	**12.318**	**4.221**	**4.166**	**1.83**	**13.4499**	**46.8936**	**4.4801**

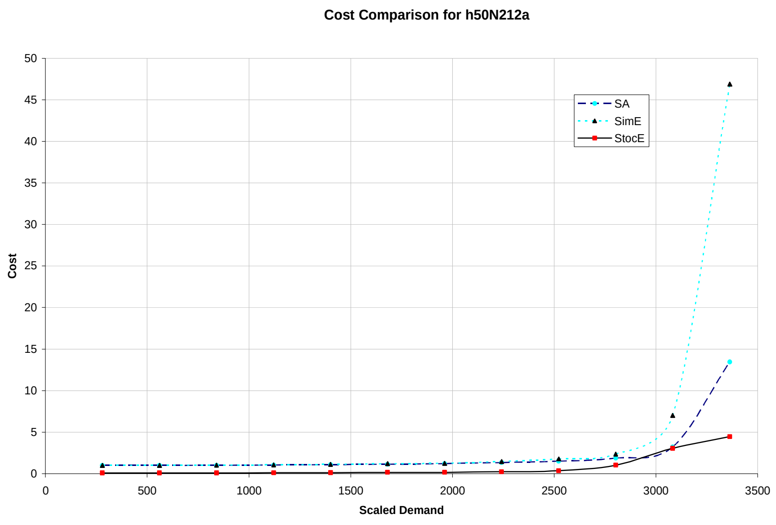


Figure A.18: Cost comparison of SA, SimE and StocE for h50N212a

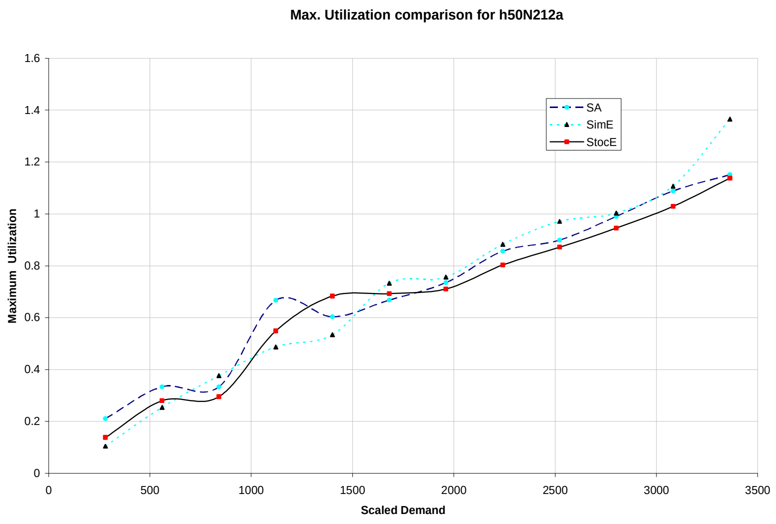


Figure A.19: Maximum Utilization comparison of SA, SimE and StocE for h50N212a

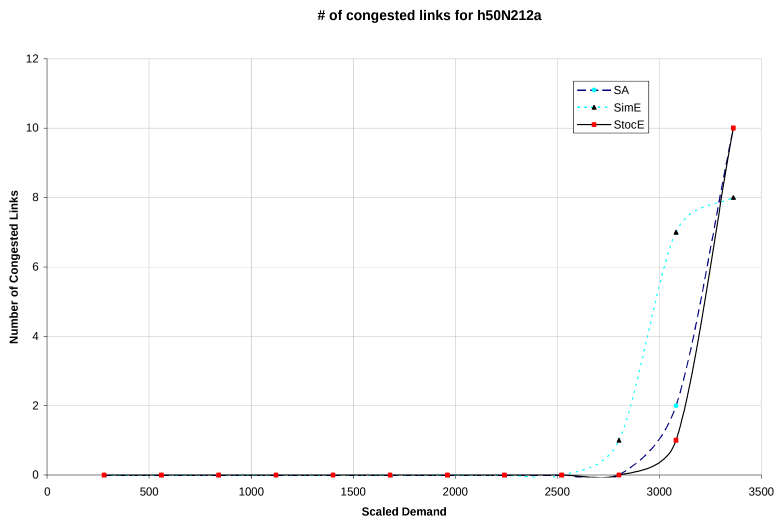


Figure A.20: Number of congested links comparison of SA, SimE and StocE for h50N212a

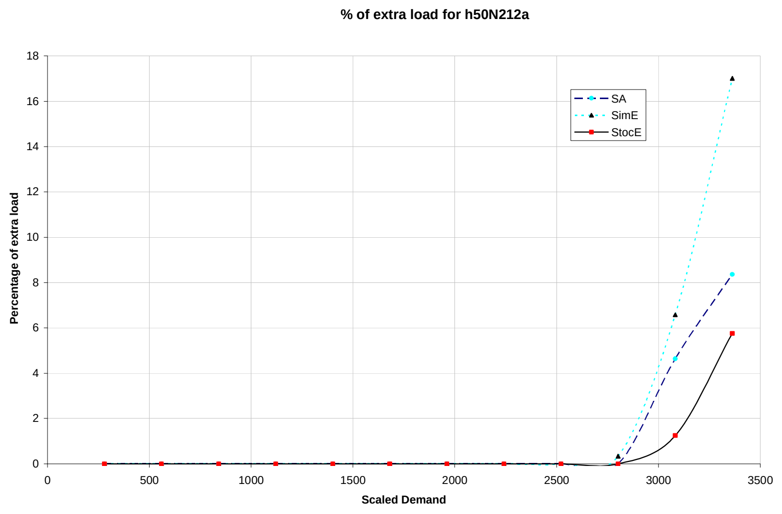


Figure A.21: Percentage of Extra Load comparison of SA, SimE and StocE for h50N212a

A.3.5 Comparisons for h100N280a

Table A.5: Cost Comparison for h100N280a

Scaled Demand	InvCap	GA Eric	HGA Eric	LS (F and T)	LPLB	SA	SimE	StocE
383.7675	1.02	1	1	1	1	1	1	0.097912
767.7351	1.02	1	1.001	1	1.001	1.001	1.000753	0.097981
1151.303	1.03	1.006	1.007	1.005	1.008	1.007	1.008486	0.102367
1535.07	1.09	1.036	1.037	1.033	1.03	1.039	1.02	0.110653
1918.838	1.17	1.083	1.078	1.076	1.06	1.095	1.087	0.12181
2302.605	1.59	1.143	1.135	1.132	1.11	1.153	1.221	0.141985
2686.373	8.87	1.234	1.22	1.217	1.2	1.260	1.234	0.184965
3070.14	17.5	1.337	1.313	1.311	1.28	1.368	1.33	0.320152
3453.908	24.93	1.602	1.557	1.558	1.52	1.643	1.633	1.528942
3837.675	38.54	3.343	3.244	3.258	3.18	3.427	3.420	2.073076
4221.443	70.25	11.976	11.812	11.823	11.71	12.027	13.245	4.62106
4605.21	**114.55**	**19.73**	**19.214**	**19.245**	**19.06**	**20.439**	**27.075**	**8.04038**

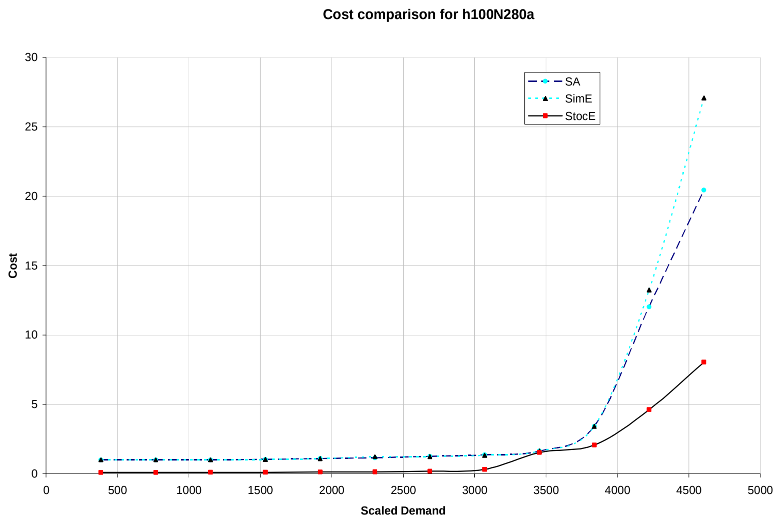


Figure A.22: Cost comparison of SA, SimE and StocE for h100N280a

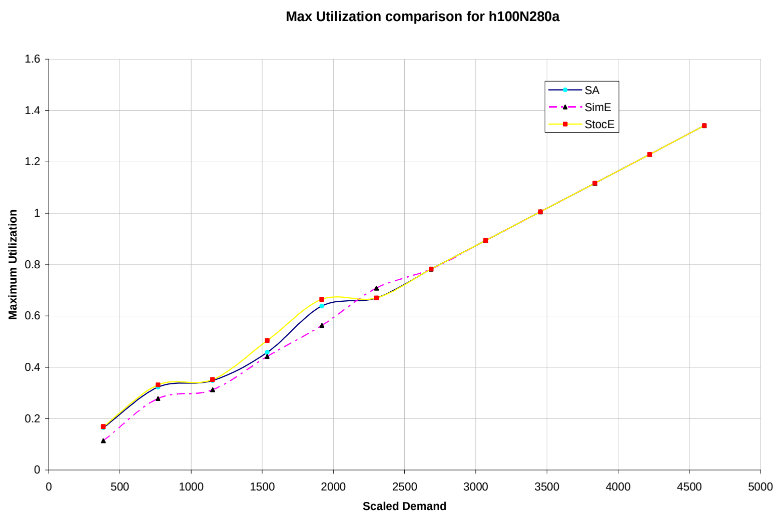


Figure A.23: Maximum Utilization comparison of SA, SimE and StocE for h100N280a

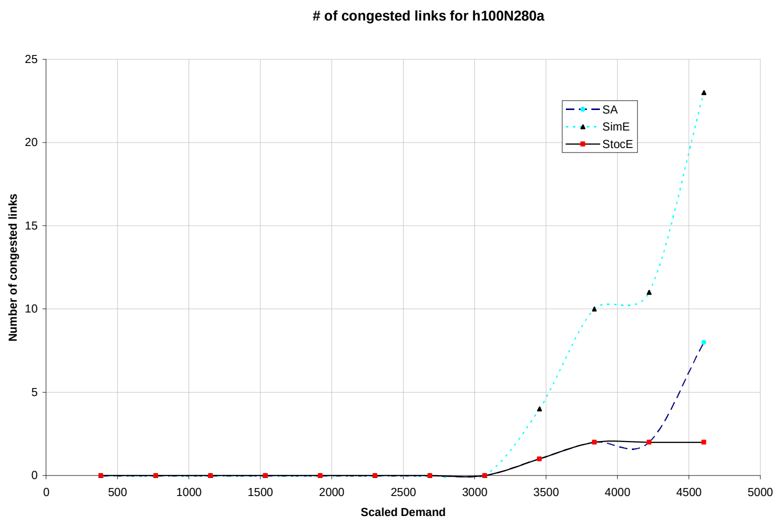


Figure A.24: Number of congested links comparison of SA, SimE and StocE for h100N280a

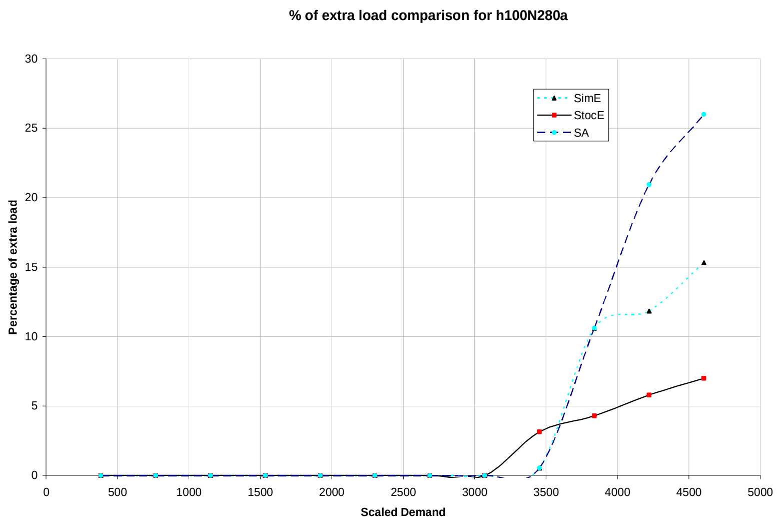


Figure A.25: Percentage of Extra Load comparison of SA, SimE and StocE for h100N280a

A.3.6 Comparisons for r100N403a

Table A.6: Cost Comparison for r100N403a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
5774.737	1	1.005	1.006	1	1	1.0101	1.001	0.1299
11549.47	1	1.008	1.006	1	1	1.0090	1.003	0.1270
17324.21	1.04	1.034	1.013	1.001	1	1.0338	1.0343	0.1659
23098.95	1.13	1.119	1.061	1.036	1.03	1.1218	1.1314	0.2697
28873.69	1.31	1.299	1.217	1.156	1.14	1.2898	1.299	1.1640
34648.42	1.68	1.546	1.394	1.312	1.29	1.5043	1.3342	3.3674
40423.16	9.25	1.932	1.601	1.507	1.47	1.7508	1.5535	6.8061
46197.9	37.22	2.849	1.882	1.757	1.71	2.2072	1.8765	10.1498
51972.63	71.52	4.375	2.32	2.112	2.02	3.1115	2.4295	17.3434
57747.37	115.26	13.822	3.131	2.703	2.46	6.3049	3.982	26.3635
63522.11	173.79	41.105	6.729	4.175	3.27	16.8293	21.8429	36.8023
69296.84	**238.56**	**108.485**	**25.283**	**10.942**	**5.79**	**68.6698**	**54.115**	**47.7778**

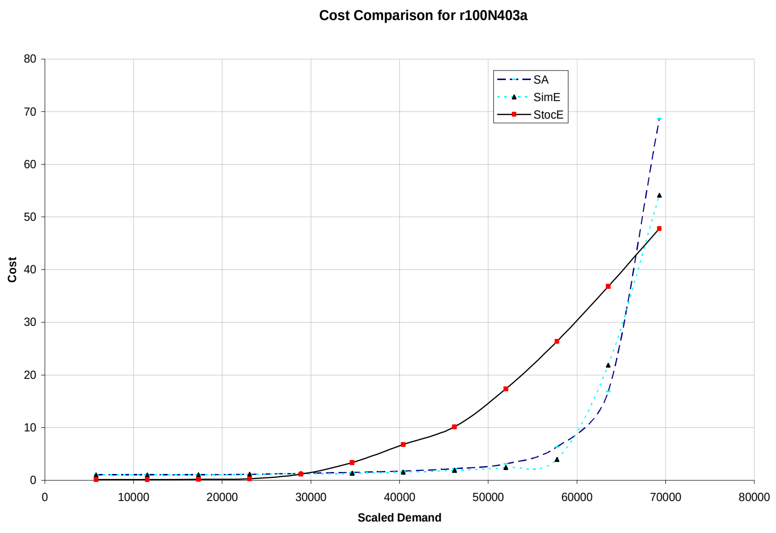


Figure A.26: Cost comparison of SA, SimE and StocE for r100N403a

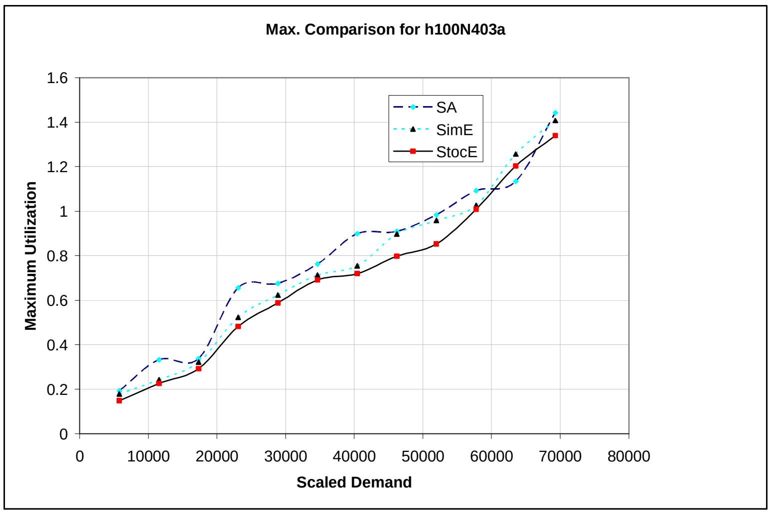


Figure A.27: Maximum Utilization comparison of SA, SimE and StocE for r100N403a

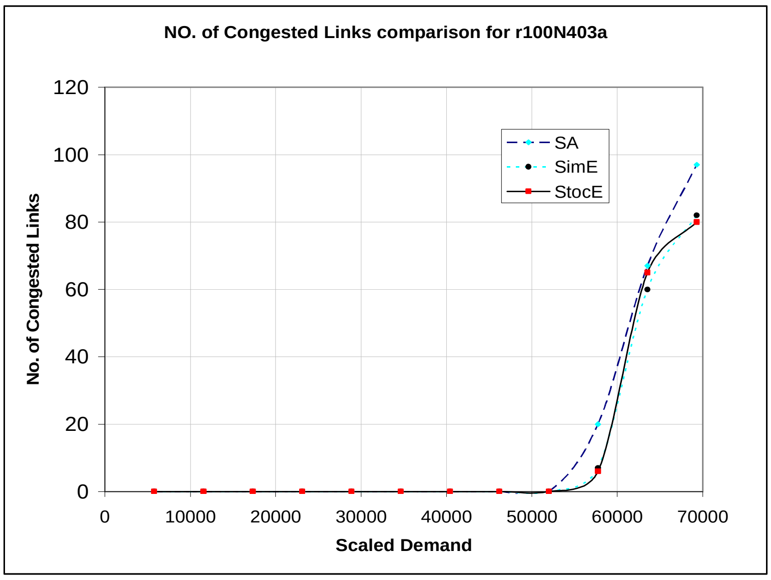


Figure A.28: Number of congested links comparison of SA, SimE and StocE for r100N403a

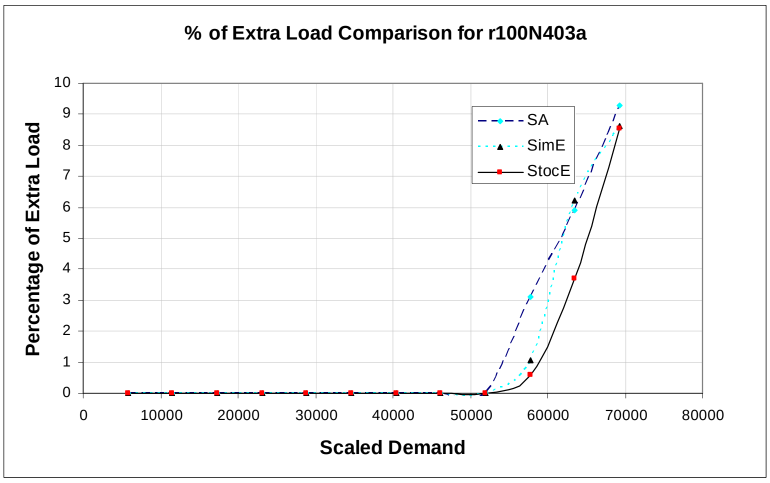


Figure A.29: Percentage of Extra Load comparison of SA, SimE and StocE for r100N403a

A.3.7 Comparisons for w50N230a

Table A.7: Cost Comparison for w50N230a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
3287.217	1	1	1	1	1	1	1.	0.112
6574.434	1	1	1	1	1	1.001	1	0.126
9861.651	1.002	1	1	1	1	1.001	1.001	0.135
13148.87	1.049	1.01	1.009	1.009	1.01	1.018	1.02	0.157
16436.08	1.129	1.05	1.031	1.028	1.03	1.066	1.05	0.2
19723.3	1.23	1.137	1.108	1.103	1.09	1.174	1.235	0.278
23010.52	1.393	1.25	1.218	1.21	1.19	1.297	1.243	0.466
26297.74	1.634	1.398	1.357	1.349	1.32	1.483	1.392	0.917
29584.95	2.706	1.593	1.514	1.51	1.48	1.676	1.576	2.188
32872.17	12.816	1.98	1.885	1.875	1.83	2.012	2.016	3.286
36159.39	38.708	4.042	3.874	3.87	2.84	4.086	4.383	6.106
39446.6	**78.084**	**11.588**	**11.288**	**11.281**	**4.3**	**11.938**	**12.382**	**9.739**

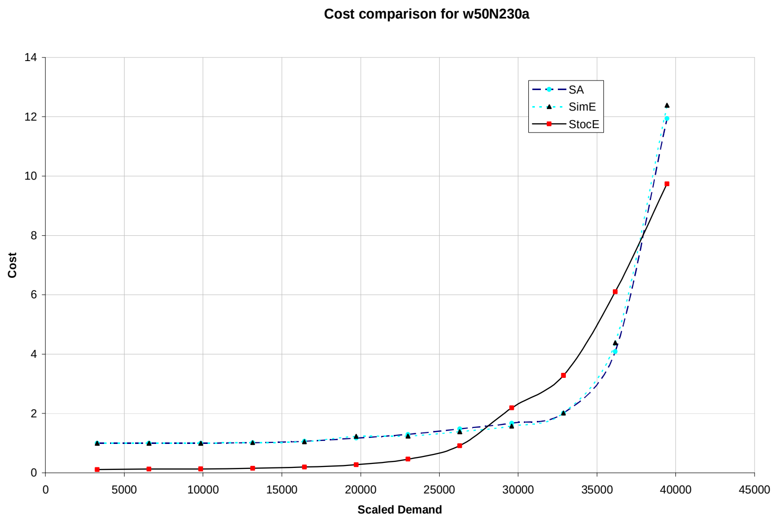


Figure A.30: Cost comparison of SA, SimE and StocE for w50N230a

A.3.8 Comparisons for w100N391a

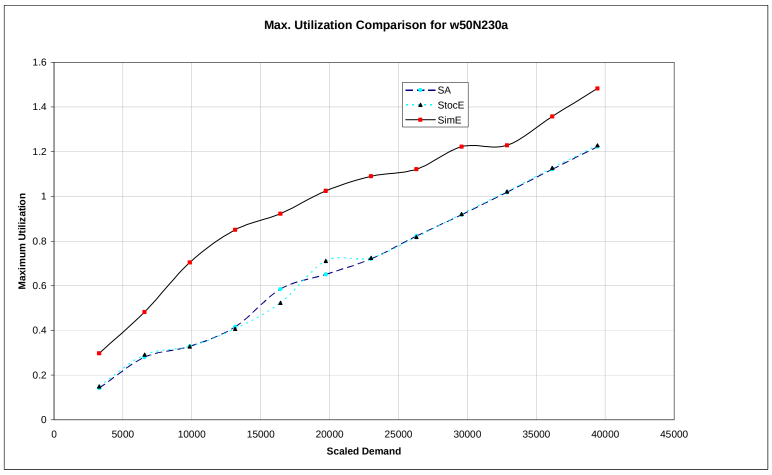


Figure A.31: Maximum Utilization comparison of SA, SimE and StocE for w50N230a

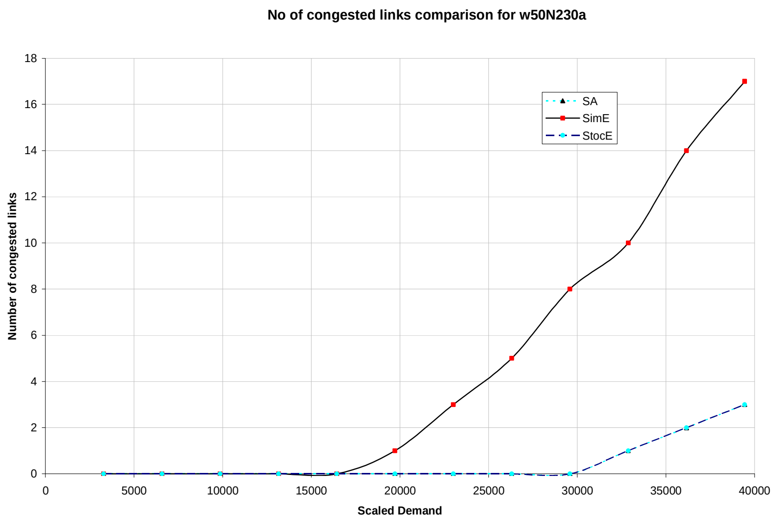


Figure A.32: Number of congested links comparison of SA, SimE and StocE for w50N230a

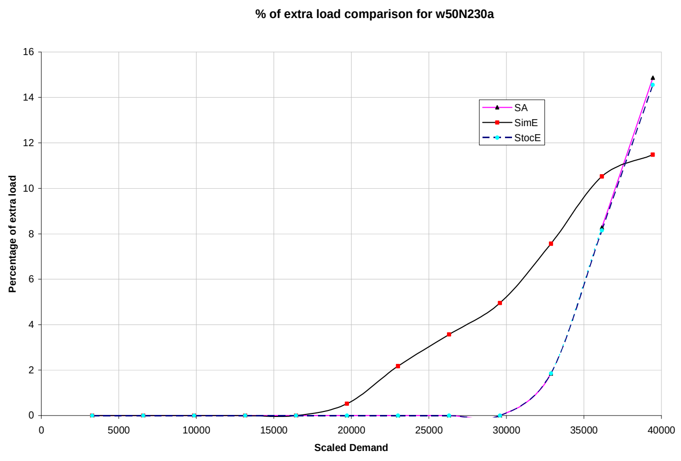


Figure A.33: Percentage of Extra Load comparison of SA, SimE and StocE for w50N230a

Table A.8: Cost Comparison for w100N391a

Scaled Demand	**InvCap**	**GA Eric**	**HGA Eric**	**LS (F and T)**	**LPLB**	**SA**	**SimE**	**StocE**
4039.477	1	1.005	1.006	1	1	1.010	1.023	0.111
8078.954	1	1.006	1.005	1	1	1.014	1.013	0.117
12118.43	1.01	1.01	1.007	1.002	1	1.022	1.034	0.131
16157.91	1.03	1.029	1.014	1.006	1.01	1.031	1.105	0.145
20197.39	1.09	1.065	1.029	1.012	1.01	1.068	1.079	0.193
24236.86	1.24	1.136	1.075	1.048	1.04	1.183	1.1	0.379
28276.34	4.66	1.268	1.186	1.13	1.11	1.284	1.152	0.956
32315.82	12.37	1.435	1.324	1.247	1.23	1.451	1.303	1.37
36355.29	23.02	1.835	1.698	1.599	1.57	1.803	1.76	4.966
40394.77	32.23	4.807	4.501	4.391	4.36	4.711	4.778	6.782
44434.25	40.64	8.953	8.317	8.191	8.14	8.616	8.571	10.637
48473.73	**54.56**	**12.647**	**11.575**	**11.409**	**11.35**	**12.065**	**12.388**	**14.743**

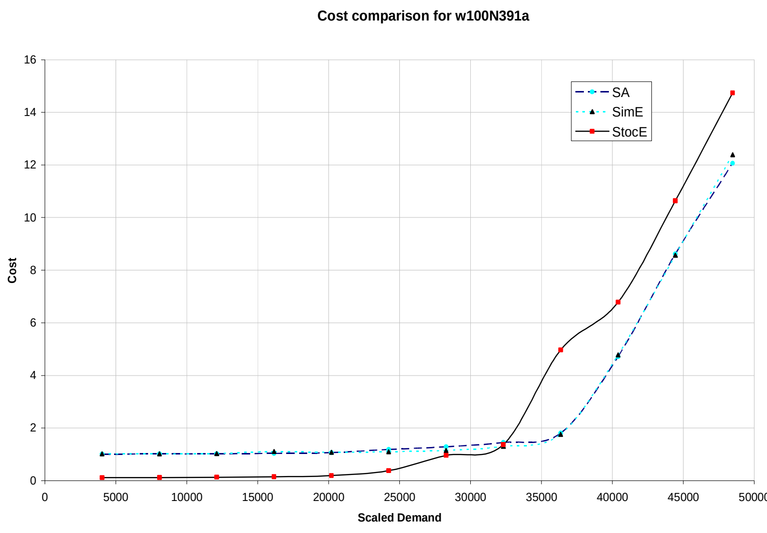


Figure A.34: Cost comparison of SA, SimE and StocE for w100N391a

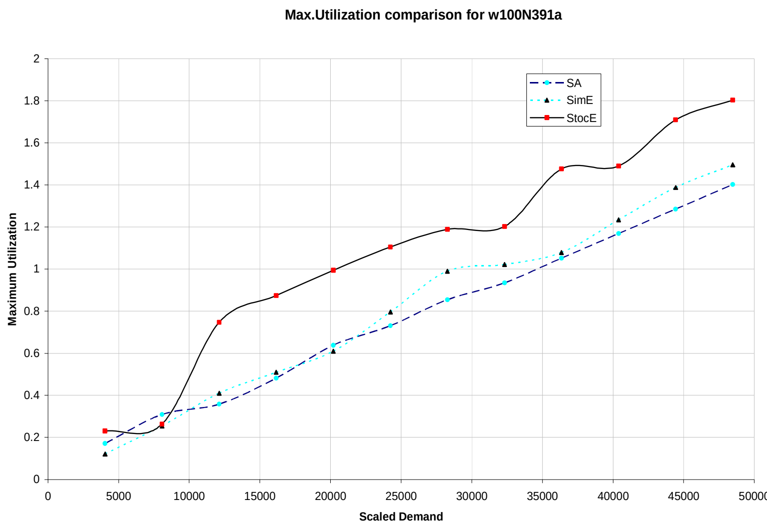


Figure A.35: Maximum Utilization comparison of SA, SimE and StocE for w100N391a

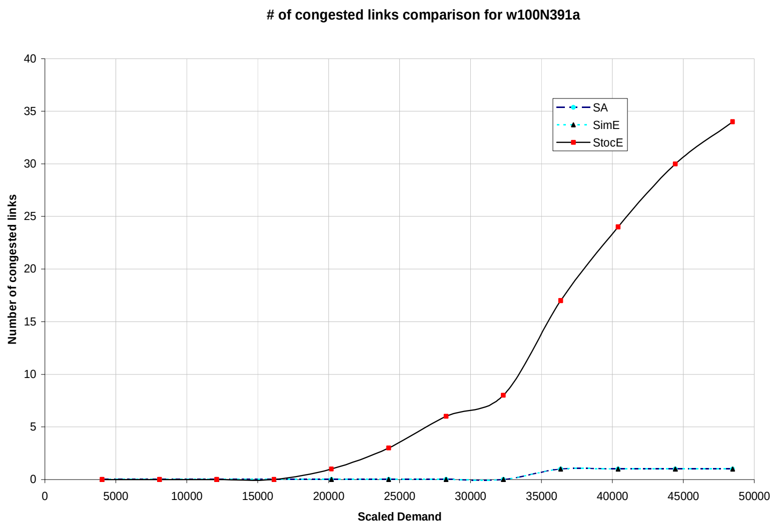


Figure A.36: Number of congested links comparison of SA, SimE and StocE for w100N391a

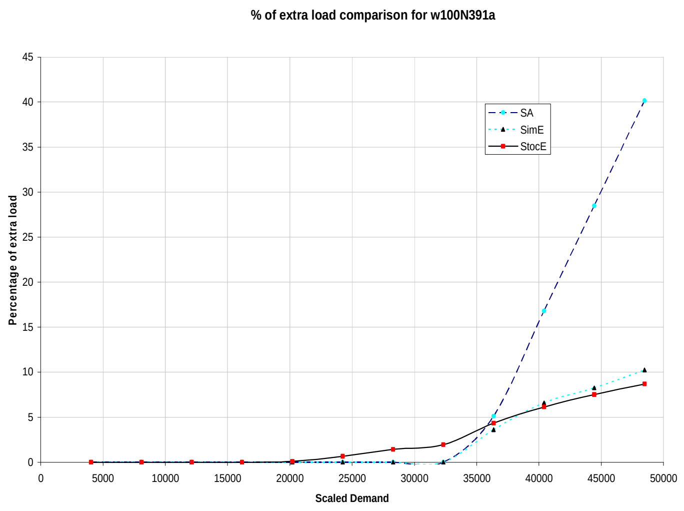


Figure A.37: Percentage of Extra Load comparison of SA, SimE and StocE for w100N391a

A.3.9 New Cost Function Comparisons

Comparisons for w100N476a:

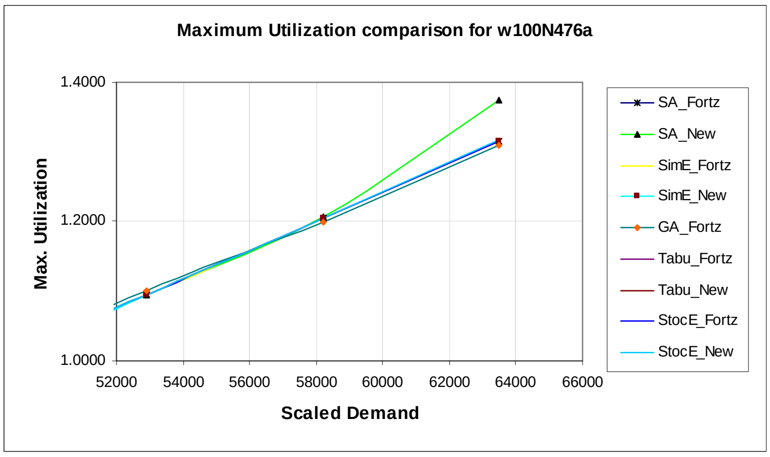


Figure A.38: Maximum Utilization comparison of w100N476a using NewCF for various algorithms

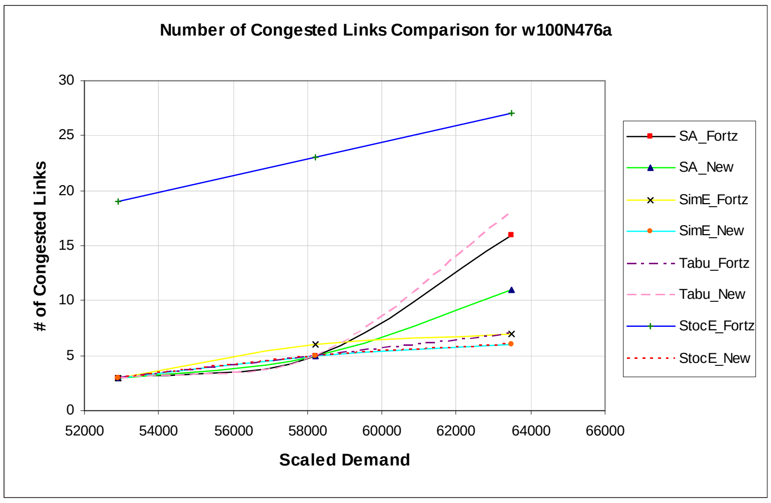


Figure A.39: Number of congested links comparison of w100N476a using NewCF for various algorithms

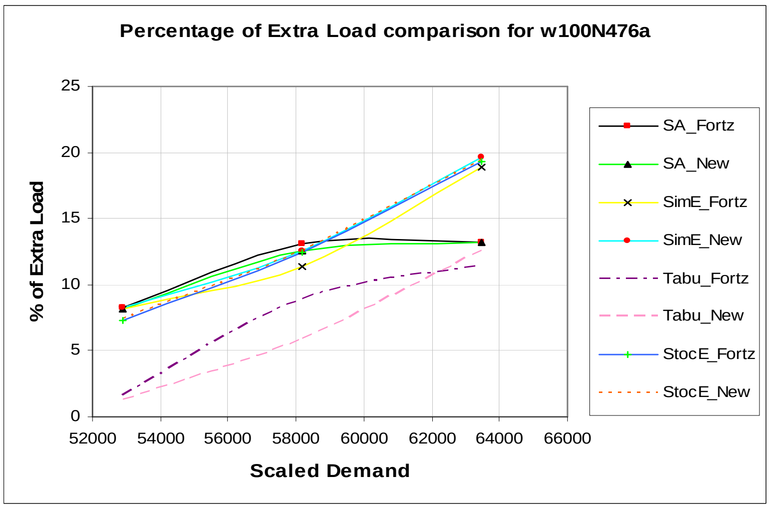


Figure A.40: Percentage of Extra Load comparison of w100N476a using NewCF for various algorithms

Comparisons for r50N245a:

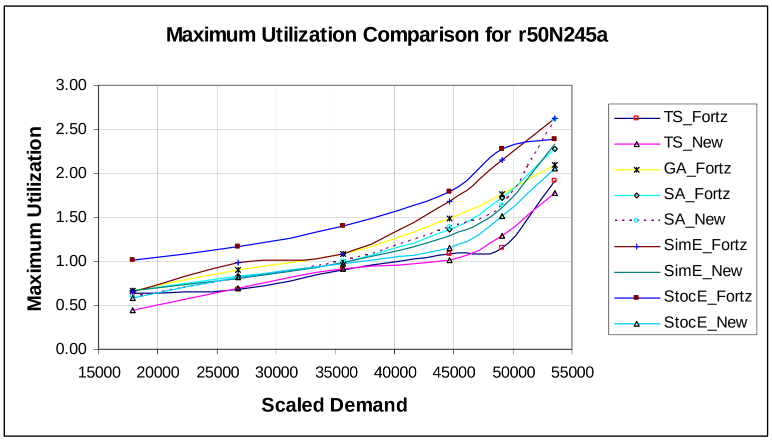


Figure A.41: Maximum Utilization comparison of r50N245a using NewCF for various algorithms

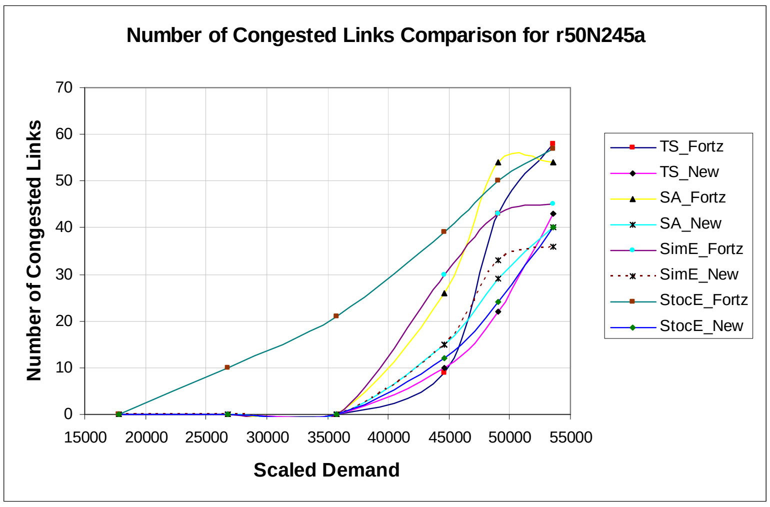


Figure A.42: Number of congested links comparison of r50N245a using NewCF for various algorithms

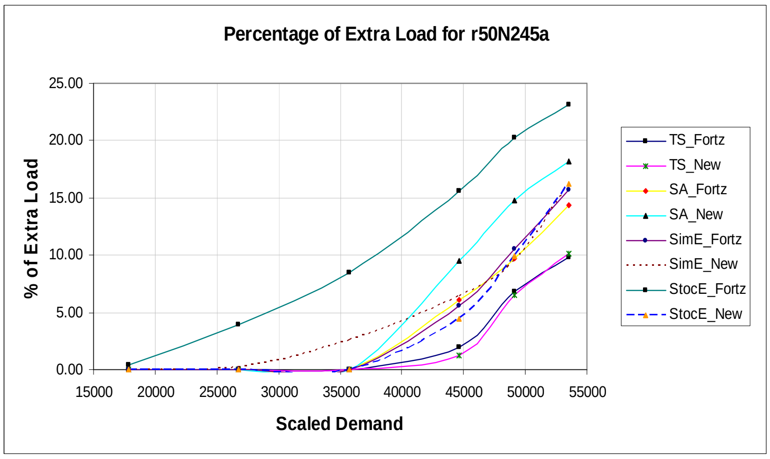


Figure A.43: Percentage of Extra Load comparison of r50N245a using NewCF for various algorithms

A.3.10 Link Failure Comparison for r50N228a

Table A.9: Cost Comparison for r50N228a Strat1 & StratOH for with link and without link failure

	Topology	**Strat 1**	**Strat OH**	**(N+1)**	**Strat 1**	**Strat OH**	**N**		
	r50N228a	**(N+1)avg**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)avg**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	3285.13	0.44	0.26	-0.17	0.44	0.53	0.09	-0.08	188.00
d9	3695.77	0.96	0.75	-0.22	1.17	1.00	-0.17	-0.39	L
d10	4106.41	2.63	2.35	-0.28	3.54	4.43	0.89	0.61	96.55
d11	4517.05	4.57	4.41	-0.16	4.76	6.32	1.56	1.40	10.26
d12	**4927.69**	**7.85**	**7.76**	**-0.08**	**8.88**	**10.85**	**1.98**	**1.90**	**4.04**

Table A.10: Cost Comparison for r50N228a Strat2 & StratOH for with link and without link failure

Topology	**Strat 2**	**Strat OH**	**(N+1)**	**Strat 2**	**Strat OH**	**N**		
	(N)OH + Link	**(N+1)OH**	**[G(+)/L(-)]**	**(N)OH**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	1.68	1.38	-0.30	2.41	3.90	1.48	1.18	0.21
d9	3.16	2.67	-0.49	4.67	5.17	0.49	0.00	100.00
d10	9.10	7.97	-1.12	8.95	10.50	1.54	0.42	72.72
d11	14.99	13.00	-1.99	13.46	17.73	4.26	2.27	46.71
d12	**21.49**	**19.42**	**-2.07**	**20.30**	**27.75**	**7.45**	**5.38**	27.79

Table A.11: Cost Comparison for r50N228a Strat1 & Strat2 for with link and without link failure

	Topology	**Strat 1**	**Strat 2**	**(N+1)**	**Strat 1**	**Strat 2**	**N**		
	r50N228a	**(N+1)avg**	**(N)OH + Link**	**[G(+)/L(-)]**	**(N)avg**	**(N)OH**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	28187.45	1.94	1.68	-0.26	2.55	2.41	0.14	-0.12	185.71
d9	31710.88	3.42	3.16	-0.26	4.18	4.67	-0.50	-0.76	L
d10	35234.31	8.89	9.10	0.20	8.06	8.95	-0.89	-0.69	22.47
d11	38757.74	14.02	14.99	0.96	15.03	13.46	1.57	2.53	G
d12	**42281.17**	**20.65**	**21.49**	**0.84**	**24.30**	**20.30**	**4.00**	**4.84**	**G**

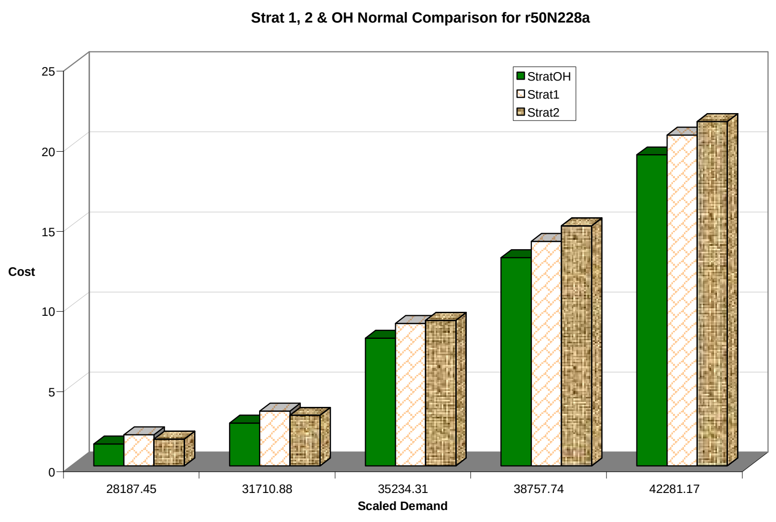


Figure A.44: Cost Comparison of StratOH, Strat1 and Strat2 for r50N228a for normal case

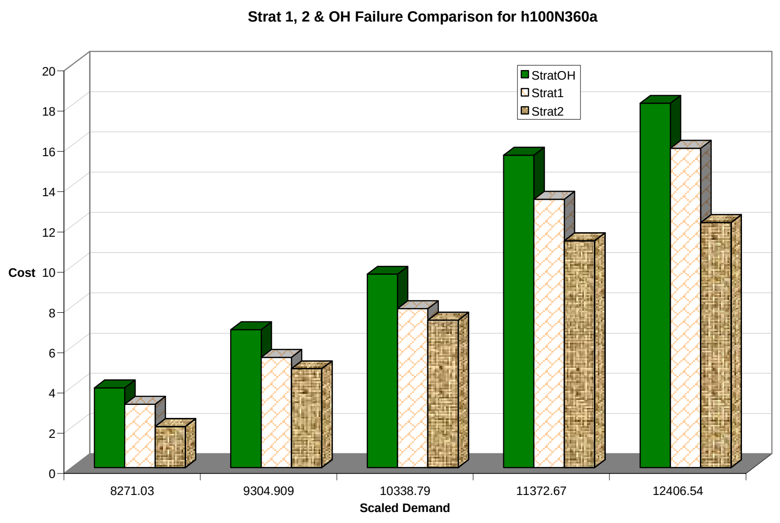


Figure A.45: Cost Comparison of StratOH, Strat1 and Strat2 for r50N228a for Link Failure case

A.3.11 Link Failure Comparison for w50N169a

Table A.12: Cost Comparison for w50N169a Strat1 & StratOH for with link and without link failure

	Topology	**Strat 1**	**Strat OH**	**(N+1)**	**Strat 1**	**Strat OH**	**N**		
	w50N169a	**(N+1)avg**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)avg**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	16940.97	0.53	0.30	-0.23	0.85	0.36	-0.50	-0.73	L
d9	19058.60	0.76	0.45	-0.31	0.98	2.04	1.06	0.75	29.25
d10	21176.22	1.83	0.71	-1.12	2.94	4.15	1.21	0.09	92.56
d11	23293.84	4.48	3.07	-1.41	5.97	8.66	2.69	1.28	52.42
d12	**25411.46**	**9.69**	**7.22**	-2.47	**11.77**	**16.05**	4.29	1.82	47.57

Table A.13: Cost Comparison for w50N169a Strat2 & StratOH for with link and without link failure

Topology	**Strat 2**	**Strat OH**	**(N+1)**	**Strat 2**	**Strat OH**	**N**		
w50N169a	**(N)OH + Link**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)OH**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	0.39	0.30	-0.10	0.39	0.36	-0.03	-0.13	L
d9	0.84	0.45	-0.39	0.72	2.04	1.32	0.93	29.55
d10	1.83	0.71	-1.12	1.99	4.15	2.16	1.04	51.85
d11	4.41	3.07	-1.34	3.60	8.66	5.06	3.72	26.48
d12	**10.31**	**7.22**	-3.09	**8.69**	**16.05**	**7.37**	**4.28**	**41.93**

Table A.14: Cost Comparison for w50N169a Strat1 & Strat2 for with link and without link failure

	Topology	**Strat 1**	**Strat 2**	**(N+1)**	**Strat 1**	**Strat 2**	**N**		
	w50N169a	**(N+1)avg**	**(N)OH + Link**	**[G(+)/L(-)]**	**(N)avg**	**(N)OH**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	16940.97	0.53	0.39	-0.14	0.85	0.39	0.47	0.33	29.79
d9	19058.60	0.76	0.84	0.08	0.98	0.72	0.26	0.34	G
d10	21176.22	1.83	1.83	0.00	2.94	1.99	0.95	0.95	G
d11	23293.84	4.48	4.41	-0.07	5.97	3.60	2.37	2.30	2.95
d12	**25411.46**	**9.69**	**10.31**	0.62	**11.77**	**8.69**	**3.08**	**3.70**	**G**

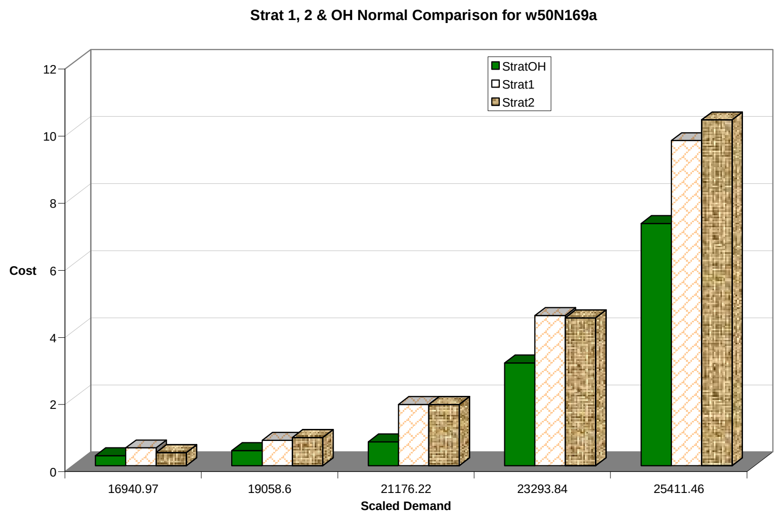


Figure A.46: Cost Comparison of StratOH, Strat1 and Strat2 for w50N169a for normal case

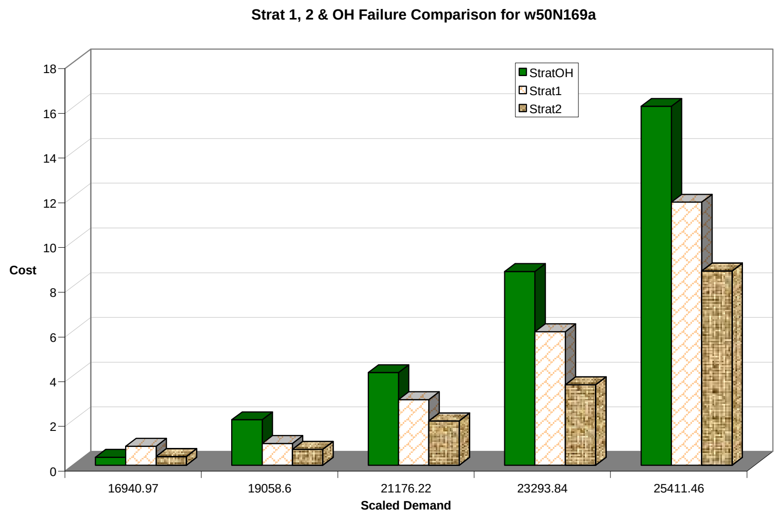


Figure A.47: Cost Comparison of StratOH, Strat1 and Strat2 for w50N169a for Link Failure case

A.3.12 Link Failure Comparison for w100N476a

Table A.15: Cost Comparison for w100N476a Strat1 & StratOH for with link and without link failure

	Topology	**Strat 1**	**Strat OH**	**(N+1)**	**Strat 1**	**Strat OH**	**N**		
	w100N476a	**(N+1)avg**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)avg**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	42328.74	3.54	3.09	-0.45	5.54	6.64	1.10	0.65	40.91
d9	47619.83	7.53	6.86	-0.67	8.59	8.49	-0.11	-0.78	L
d10	52910.92	11.88	10.74	-1.13	13.46	15.64	2.19	1.06	51.60
d11	58202.02	17.44	15.97	-1.48	19.86	22.61	2.75	1.27	53.82
d12	63493.11	**25.00**	**23.02**	-1.98	**27.60**	**31.53**	**3.92**	**1.94**	**50.51**

Table A.16: Cost Comparison for w100N476a Strat2 & StratOH for with link and without link failure

Topology	**Strat 2**	**Strat OH**	**(N+1)**	**Strat 2**	**Strat OH**	**N**		
w100N476a	**(N)OH + Link**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)OH**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	3.58	3.09	-0.49	5.41	6.64	1.22	0.73	40.16
d9	7.46	6.86	-0.60	8.79	8.49	-0.31	-0.91	L
d10	11.88	10.74	-1.14	10.93	15.64	4.71	3.57	24.20
d11	17.64	15.97	-1.67	15.11	22.61	7.50	5.83	22.27
d12	**25.81**	**23.02**	-2.79	**22.86**	**31.53**	5.87	11.45	32.22

Table A.17: Cost Comparison for w100N476a Strat1 & Strat2 for with link and without link failure

	Topology	**Strat 1**	**Strat 2**	**(N+1)**	**Strat 1**	**Strat 2**	**N**		
	w100N476a	**(N+1)avg**	**(N)OH + Link**	**[G(+)/L(-)]**	**(N)avg**	**(N)OH**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	42328.74	3.54	3.58	0.04	5.54	5.41	0.13	0.17	G
d9	47619.83	7.53	7.46	-0.07	8.59	8.79	-0.20	-0.27	L
d10	52910.92	11.88	11.88	0.00	13.46	10.93	2.53	2.53	G
d11	58202.02	17.44	17.64	0.19	19.86	15.11	4.75	4.94	G
d12	63493.11	**25.00**	**25.81**	0.81	**27.60**	**22.86**	4.74	5.55	G

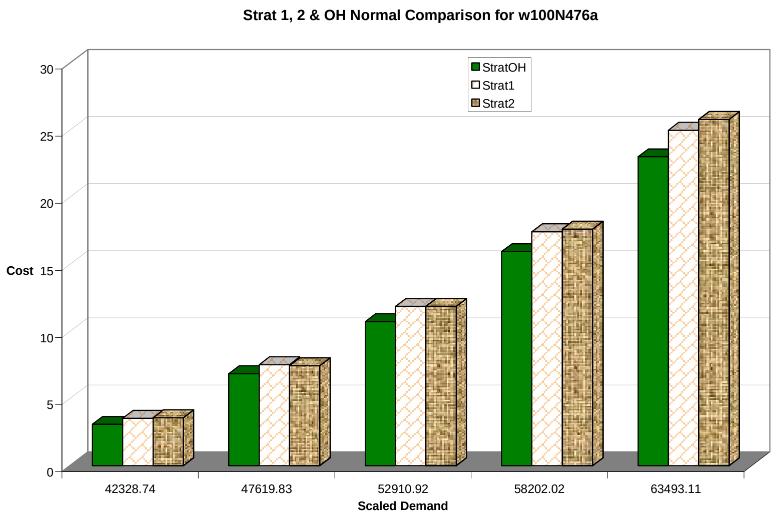


Figure A.48: Cost Comparison of StratOH, Strat1 and Strat2 for w100N476a for normal case

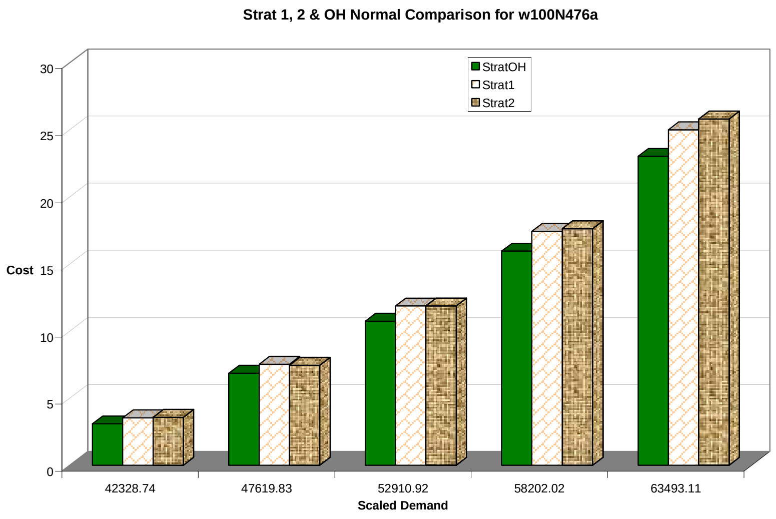


Figure A.49: Cost Comparison of StratOH, Strat1 and Strat2 for w100N476a for Link Failure case

A.3.13 Link Failure Comparison for r100N503a

Table A.18: Cost Comparison for r100N503a Strat1 & StratOH for with link and without link failure

	Topology	Strat 1	Strat OH	(N+1)	Strat 1	Strat OH	N		
	r100N503a	**(N+1)avg**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)avg**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	67062.90	17.18	16.33	-0.85	18.12	19.23	1.11	0.26	76.58
d9	75445.76	23.35	22.36	-0.99	25.89	24.87	-1.02	-2.01	L
d10	83828.68	33.59	32.08	-1.51	33.69	37.27	3.57	2.06	42.30
d11	92211.49	42.31	40.27	-2.04	45.39	49.32	3.93	1.89	51.91
d12	**100594.30**	**53.99**	**51.80**	**-2.19**	**55.31**	**60.31**	**5.01**	**2.82**	**43.71**

Table A.19: Cost Comparison for r100N503a Strat2 & StratOH for with link and without link failure

Topology	Strat 2	Strat OH	(N+1)	Strat 2	Strat OH	N		
r100N503a	**(N)OH + Link**	**(N+1)OH**	**[G(+)/L(-)]**	**(N)OH**	**[(N+1)OH-Link]**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	17.07	16.33	-0.74	17.63	19.23	1.60	0.86	46.25
d9	23.43	22.36	-1.07	22.78	24.87	2.09	1.02	51.20
d10	33.74	32.08	-1.66	31.90	37.27	5.37	3.71	30.91
d11	42.47	40.27	-2.20	40.49	49.32	8.83	6.63	24.92
d12	**54.44**	**51.80**	**-2.64**	**50.15**	**60.31**	**10.16**	**7.52**	**23.98**

Table A.20: Cost Comparison for r100N503a Strat1 & Strat2 for with link and without link failure

	Topology	Strat 1	Strat 2	(N+1)	Strat 1	Strat 2	N		
	r100N503a	**(N+1)avg**	**(N)OH + Link**	**[G(+)/L(-)]**	**(N)avg**	**(N)OH**	**[G(+)/L(-)]**	**Total Gain**	**L:G%**
d8	67062.90	17.18	17.07	-0.11	18.12	17.63	0.49	0.38	22.45
d9	75445.76	23.35	23.43	0.08	25.89	22.78	3.11	3.19	G
d10	83828.68	33.59	33.74	0.15	33.69	31.90	1.80	1.95	G
d11	92211.49	42.31	42.47	0.16	45.39	40.49	4.90	5.06	G
d12	**100594.30**	**53.99**	**54.44**	**0.44**	**55.31**	**50.15**	**5.16**	**5.60**	**G**

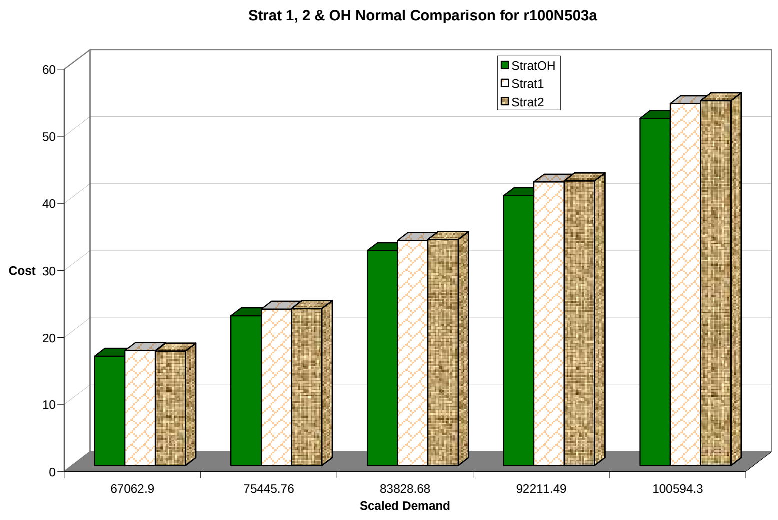


Figure A.50: Cost Comparison of StratOH, Strat1 and Strat2 for r100N503a for normal case

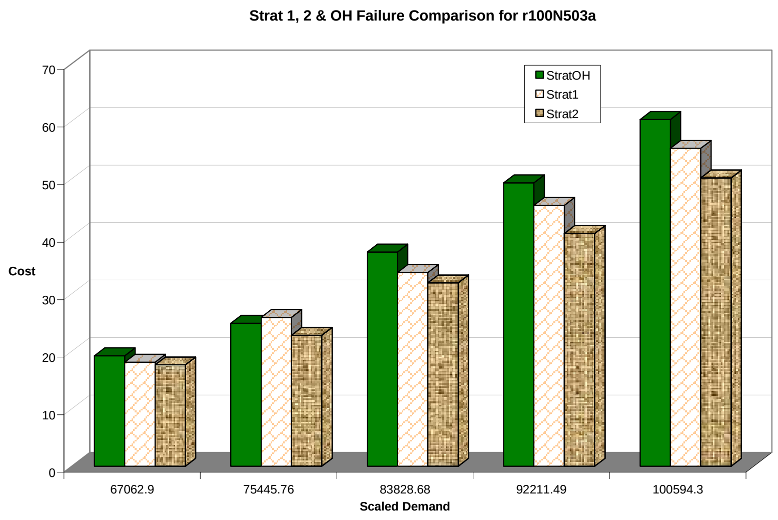


Figure A.51: Cost Comparison of StratOH, Strat1 and Strat2 for r100N503a for Link Failure case

Appendix B

Open shortest Path First(OSPF) protocol

B.1 Introduction to OSPF

The Open Shortest Path First (OSPF) protocol, defined in RFC 2328, is an Interior Gateway Protocol used to distribute routing information within a single Autonomous System. OSPF is a link-state protocol. The link can be considered as an interface on the router. The state of the link is a description of that interface and of its relationship to its neighboring routers. A description of the interface would include, for example, the IP address of the interface, the mask, the type of network it is connected to, the routers connected to that network and so on. The collection of all these link-states would form a link-state database.

In order to construct and calculate the shortest path to all known destinations, OSPF uses a Link-State algorithm. The various steps of this algorithm are:

1. Upon initialization or due to any change in routing information, a router will generate a link-state advertisement. This advertisement will represent the collection of all link-states on that router.

2. All routers will exchange link-states by means of flooding. Each router that receives a link-state update should store a copy in its link-state database and then propagate the update to other routers.

3. After the database of each router is completed, the router will calculate a Shortest Path Tree to all destinations. The router uses the Dijkstra algorithm to calculate the shortest path tree. The destinations, the associated cost and the next hop to reach those destinations will form the IP routing table.

4. In case no changes in the OSPF network occur, such as cost of a link or a network being added or deleted, OSPF should be very quiet. Any changes that occur are communicated via link-state packets, and the Dijkstra's algorithm is recalculated to find the shortest path.

The Link State Database (LSDB) contains the link state advertisements sent around the 'Area' and each router holds an identical copy of this LSDB. The router then creates a Shortest Path First (SPF) tree using Dijkstra's algorithm on the

LSDB and a routing table can be derived from the SPF tree which now contains the best route to each router.

The Shortest Path First (SPF) routing algorithm is the basis for OSPF operations. When an SPF router is powered up, it initializes its routing-protocol data structures and then waits for indications from lower-layer protocols that its interfaces are functional. It uses cost as its routing metric. A link state database is constructed of the network topology which is identical on all routers in the area.

B.2 OSPF vs RIP

The rapid growth and expansion of today's networks has pushed Routing Information Protocol (RIP) to its limits. RIP has certain limitations that could cause problems in large networks:

1. RIP has a limit of 15 hops.

2. RIP cannot handle Variable Length Subnet Masks (VLSM). Given the shortage of IP addresses and the flexibility VLSM gives in the efficient assignment of IP addresses, this is considered a major flaw.

3. Periodic broadcasts of the full routing table will consume a large amount of bandwidth. This is a major problem with large networks especially on slow links and WAN clouds.

4. RIP converges slower than OSPF. In large networks convergence gets to be in the order of minutes.

5. RIP has no concept of network delays and link costs. Routing decisions are based on hop counts. The path with the lowest hop count to the destination is always preferred even if the longer path has a better aggregate link bandwidth and slower delays.

6. RIP networks are flat networks. There is no concept of areas or boundaries.

RIP2 addresses the issues of VLSM, authentication, and multicast routing updates. RIP2 is not a big improvement over RIP (now called RIP 1) because it still has the limitations of hop counts and slow convergence which are essential in todays large networks.

OSPF, on the other hand, addresses most of the issues presented above:

1. With OSPF, there is no limitation on the hop count.

2. The intelligent use of VLSM is very useful in IP address allocation.

3. OSPF uses IP multicast to send link-state updates. This ensures less processing on routers that are not listening to OSPF packets. Also, updates are only sent in case routing changes occur instead of periodically. This ensures a better use of bandwidth.

4. OSPF has better convergence than RIP. This is because routing changes are propagated instantaneously and not periodically.

5. OSPF allows for better load balancing.

6. OSPF allows for a logical definition of networks where routers can be divided into areas. This will limit the explosion of link state updates over the whole network. This also provides a mechanism for aggregating routes and cutting down on the unnecessary propagation of subnet information.

7. OSPF allows for routing authentication by using different methods of password authentication.

8. OSPF allows for the transfer and tagging of external routes injected into an Autonomous System. This keeps track of external routes injected by exterior protocols such as BGP.

9. OSPF is an open standard, not related to any particular vendor.

B.2.1 Choice between RIP and OSPF

In many places, RIP is still used in TCP/IP networks that have not been upgraded to OSPF. It is also used on OSPF networks as an end-station-to-router protocol. OSPF addresses all the deficiencies of RIP, without affecting connectivity to RIP based networks. Fast growing networks must be designed properly if the capabilities of

OSPF are to be fully exploited. Because of its ability to handle variable networking masks, OSPF also helps to reduce waste of today's precious IP addresses. Ideally, network design should include a consistent enterprise-wide IP address assignment policy that lends itself to the creation of OSPF areas and address summarization. If correct design and router-tuning takes place, OSPF will allow networks to scale to very large topologies, while maintaining high levels of availability and performance.

B.3 Disadvantages of OSPF

- OSPF is very processor intensive.
- OSPF maintains multiple copies of routing information, increasing the amount of memory needed.
- Using areas, OSPF can be logically segmented (this can be a good thing and a bad thing).
- OSPF is not as easy to learn as some other protocols.
- In the case where an entire network is running OSPF, and one link within it is "bouncing" every few seconds, OSPF updates would dominate the network by informing every other router every time the link changed state.

OSPF is perhaps the most widely used IGP in large networks. It can operate securely, using MD5 to authenticate peers before forming adjacencies, and before accepting link-state advertisements (LSA). A natural successor to the Routing Information Protocol (RIP), it was VLSM-capable or classless from its inception. Multicast extensions to OSPF, the Multicast Open Shortest Path First (MOSPF) protocols, have been defined, but these are not widely used at present. OSPF can "tag" routes, and propagate the tags along with the routes.

Routers in the same broadcast domain or at each end of a point-to-point telecommunications link form adjacencies when they have detected each other. The routers elect a designated router (DR) and a backup designated router (BDR) which act as a hub to reduce traffic between routers. OSPF uses both unicast and multicast to send "hello packets" and link state updates. Multicast addresses 224.0.0.5 and 224.0.0.6 are reserved for OSPF. In contrast to the Routing Information Protocol (RIP) or the Border Gateway Protocol (BGP), OSPF does not use TCP or UDP but uses IP directly, via IP protocol 89.

B.4 OSPF Network Type

OSPF has three network types:

1. Broadcast networks (Ethernet, Token Ring, FDDI)

2. Nonbroadcast multiaccess networks (SMDS, Frame Relay, X.25, Classic IP Over ATM)

3. Point-to-multipoint networks (HDLC, PPP)

Broadcast is typically used on networks where broadcast and multicast are supported. The idea is to send out one OSPF multicast that reaches multiple receivers. This can be appropriate with X.25 and Frame Relay if they have a full mesh of PVC's. Loss of a PVC can cause problems with OSPF broadcast networks since some routers with a common subnet will no longer be able to transmit directly to each other.

Non-broadcast is used where broadcast/multicast is not feasible. The drawback to this interface type is that neighbors must be explicitly configured, which can get to be a nuisance.

Point-to-multipoint is for use in partial mesh situations, such as typical Frame Relay or ATM networks. It causes the generation of multiple host routers. It saves from having to configure neighbor statements. It allows to use the NBMA addressing model, where the FR or ATM cloud is one IP subnet. And it tolerates loss of virtual circuits.

Types of Multi-access networks

These are typically Frame Relay, ATM or X.25 networks that have no broadcast capability but have many routers connected. There are three types:

1. Hub and Spoke - a central router has links to other routers in a star arrangement. A spoke can only talk to other spokes via the hub.

2. Full Mesh - each router has a link to every other router providing full resilience.

3. Partial Mesh - not all routers have links to the central site.

Point-to-Point and Multipoint-to-Point networks have no need for DR/BDRs and form adjacencies with their neighbours automatically and quickly without the need for static neighbours being configured.

In a hub-spoke network operating in Broadcast mode the DR really needs to be the hub router in order for it to maintain contact with all the routers. It is therefore important to make sure that none of the other routers can become the DR by setting their interface priorities to 0 or raising the hub router's interface priority to be the highest.

The Non-Broadcast Multi-Access (NBMA) network has all the router interfaces in the same subnet, in addition the neighbours have to be statically defined because there is no facility for broadcasts. You can also configure sub-interfaces to allow separate subnets and therefore separate NBMA networks to exist. Rather than use a NBMA network where you have to statically configure the neighbours you can configure a Point-to-Multipoint network for Partial Mesh networks. In this case there is no DR and each link is treated as a separate Point-to-Point. A Point-to-Multipoint network can exist in one subnet.

There are some Point-to-Multipoint networks such as Classic IP over ATM that do not support broadcasts. For these networks you can configure a Point-to-

Multipoint Non-broadcast mode that requires the configuration of static neighbors since they cannot be discovered dynamically.

B.5 OSPF packets

B.5.1 OSPF packet header

All OSPF packets have a common 24 byte header that contains all the information necessary to determine whether OSPF should accept the packet or not as shown in the Table B.1.

Table B.1: OSPF Packet Format

Field Length in Bytes	**Description**
1	Version Number
1	Type
2	Packet Length
4	RouterID
4	AreaID
4	CheckSum
4	Authentication Type
8	Data

Version number identifies the OSPF version used. This can be either 2 or 3.

Type identifies the OSPF packet type.

Packet length specifies the packet length, including the OSPF header, in bytes.

Router ID identifies the source of the packet.

Area ID identifies the area to which the packet belongs. All OSPF packets are associated with a single area.

Checksum checks the entire packet contents for any damage suffered in transit.

Authentication type contains the authentication type. All OSPF protocol exchanges are authenticated. The authentication type is configurable on per-area basis. This is valid for OSPFv2 only.

OSPF Hello Packets

Routers periodically send Hello packets on all interfaces, including virtual links, to establish and maintain neighbor relationships. Hello packets are multicast on physical networks that have a multicast of broadcast capability, which enables dynamic discovery of neighboring routers. On brodcast networks, dynamic neighbor discovery is not possible. So, all neighbors have to be statically using the neighbor statement.

Hello packets consist of the OSPF header plus the following fields:

- Network mask
- Hello Interval
- Options
- Router Priority

Table B.2: Packet type and its code

Type Code	**Packet Type**
1	Hello
2	Database Description
3	Link State Request
4	Link State Update
5	Link State Acknowledgement

- Router Dead Interval
- Designated Router(DR)
- Backup Designated Router(BDR)
- Neighbor.

Within the OSPF header the packet type is indicated by way of a type code as shown in Table B.2.

Database Description Packets

When initializing an adjacency, OSPF exchanges database description packets, which describe the contents of the topological database. These packets consist of the OSPF header, packet sequence number, and the link-state advertisement's header.

Link-State Request Packets

When a router detects that portions of its topological database are out of date, it sends a link-state request packet to a neighbor requesting a precise instance of the database. These packets consist of the OSPF header plus fields that uniquely identify the database information that the router is seeking.

Link-State Update Packets

Link-state update packets carry one or more link-state advertisements one hop farther from their origin. The router multicasts (floods) these packets on physical networks that support multicast or broadcast mode. The router acknowledges all link-state update packets and, if retransmission is necessary, sends the retransmitted advertisements unicast.

Link-state update packets consist of the OSPF header plus the following fields:

- Number of advertisementsNumber of link-state advertisements included in this packet.

- Link-state advertisementsThe link-state advertisements themselves.

Link-State Acknowledgment Packets

The router sends link-state acknowledgment packets in response to link-state update packets to verify that the update packets have been received successfully. A single acknowledgment packet can include responses to multiple update packets. Link-state acknowledgment packets consist of the OSPF header plus the link-state advertisement header.

Link-State Advertisement Packet Types

Link-state request, link-state update, and link-state acknowledgment packets are used to reliably flood link-state advertisement packets. OSPF sends the following types of link-state advertisements:

- Router link advertisements
- Network link advertisements
- Summary link advertisements
- AS external link advertisement

Each link-state advertisement type describes a portion of the OSPF routing domain. All link-state advertisements are flooded throughout the AS. Each link-state advertisement packet begins with a common 20-byte header.

B.6 OSPF Areas

In OSPF, a single AS can be divided into smaller groups called areas. This reduces the number of link-state advertisements and other OSPF overhead traffic sent on the network, and it reduces the size of the topological database that each router must maintain. Within a network multiple Areas can be created to help ease CPU use in SPF calculations, memory use and the number of LSAs being transmitted. 60-80 routers are considered to be the maximum to have in one area. The Areas are defined on the routers and then interfaces are assigned to the areas. The default area is 0.0.0.0 and should exist even if there is only one area in the whole network (which is the default situation).

An area is a set of networks and hosts within an AS that have been administratively grouped together. Routers that are wholly within an area are called internal routers. All interfaces on internal routers are directly connected to networks within the area. The topology of an area is hidden from the rest of the AS, thus significantly reducing routing traffic in the AS. Also, routing within the area is determined only by the area's topology, providing the area with some protection from bad routing data. All routers within an area have identical topological databases.

Area Border Routers

Routers that belong to more than one area are called area border routers. They maintain a separate topological database for each area to which they are connected.

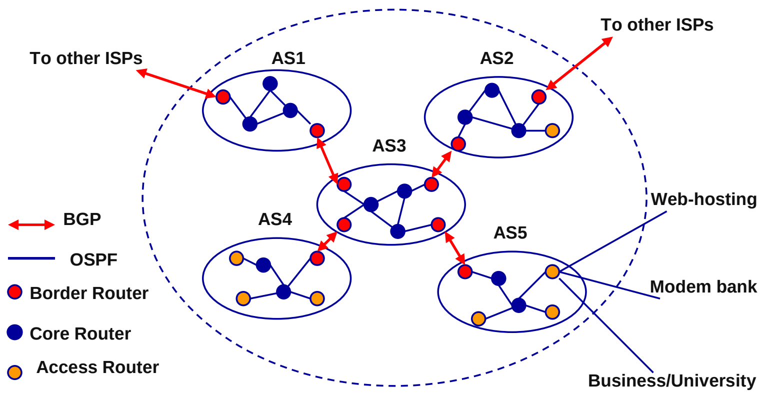


Figure B.1: Backbone Topology showing OSPF Hierarchy

Backbone Areas

An OSPF backbone area consists of all networks in area ID 0.0.0.0, their attached routers, and all area border routers. The backbone itself does not have any area border routers. The backbone distributes routing information between areas. The backbone is simply another area, so the terminology and rules of areas apply: a router that is directly connected to the backbone is an internal router on the backbone, and the backbone's topology, as shown in Figure B.1 is hidden from the other areas in the AS.

AS Boundary Routers

Routers that exchange routing information with routers in other ASs are called AS boundary routers. They advertise externally learned routes throughout the AS. Any router in the ASan internal router, an area border router, or a backbone routercan be an AS boundary router. Every router within the AS knows the path to the AS boundary routers.

Stub Areas

Stub areas are areas through which or into which AS external advertisements are not flooded. Creating stub areas when much of the topological database consists of AS external advertisements reduces the size of the topological databases and therefore the amount of memory required on the internal routers in the stub area. When an area border router is configured for a stub area, the router automatically advertises a default route in place of the external routes that are not being advertised within the stub area so that routers in the stub area can reach destinations outside the area. The following restrictions apply to stub areas: *A virtual link cannot be created through a stub area, and a stub area cannot contain an AS boundary router.*

Not-So-Stubby Areas

An OSPF stub area has no external routes in it, so you cannot redistribute from another protocol into a stub area. A not-so-stubby area (NSSA) allows external routes to be flooded within the area. These routes are then leaked into other areas. However, external routes from other areas still do not enter the NSSA.

Transit Areas

Transit areas are used to pass traffic from one adjacent area to the backbone (or to another area if the backbone is more than two hops away from an area). The traffic does not originate in, nor is it destined for, the transit area.

B.7 Virtual Links

If an area has been added to an OSPF network and it is not possible to connect it directly to the backbone or two organisations that both have a backbone area having merged, then a virtual link is required. The link must connect two routers within a common area called a Transit Area and one of these routers must be connected to the backbone. A good example of its use could be when two organizations merge and two Area 0s must be connected i.e. 'patching the backbone'. Virtual links cannot be used to patch together a split area that is not the backbone area. Instead a tunnel must be used, the IP address of which is in one of the areas.

B.8 Metrics

The process of selecting a new path requires:

- Knowledge of the flow requirements and characteristics.
- Information about availability of resources in networks.

- Evaluate the amount of resources that has to be allocated to support the new flow. This, because it can be decided not to accept a new flow, even when resources are available, if the cost of the path is deemed too high.

The metrics involved in the path selection process are:

1. Link available bandwidth

2. Link propagation delay

3. Hop count

4. Weight assignment based on cost function.

In the first case, the relevant metric to accept a new flow is the current amount of available (i.e., unallocated) bandwidth. Changes in this metric need to be advertised as part of one extended LSA, so that accurate information is available to the path selection algorithm.

In the second case, Link propagation delay has to be considered to be able to identify high latency links, i.e., satellite links, which may be unsuitable for the request. This information has to be flooded as part of one extended LSA with the advantage that the timely dissemination is not critical, since this parameter is unlike to change significantly over time.

In the third case, smaller number of hops is preferable because it consumes fewer network resources; then the selection algorithm will attempt to find the minimum hop path capable of satisfying the given request. Fortunately, this is a metric that does not affect LSAs because it is used already implicitly as part of the path selection algorithm.

B.8.1 External Metrics

When OSPF exports route information from external ASs, it includes a cost, or external metric, in the route. There are two types of external metrics: Type 1 and Type 2. Type 1 external metrics are equivalent to the link-state metric; that is, the cost of the route used in the internal AS. Type 2 external metrics are greater than the cost of any path internal to the AS.

B.9 Designated Router

Each multiaccess network has a designated router, which performs two main functions:

- Originate network link advertisements on behalf of the network.
- Establish adjacencies with all routers on the network, thus participating in the synchronizing of the link-state databases.

The OSPF hello protocol elects a designated router for the network based on the priorities advertised by all the routers. In general, when an interface first becomes functional, it checks whether the network currently has a designated router. If there is one, the router accepts that designated router regardless of its own router priority. Otherwise, if the router has the highest priority on the network, it becomes the designated router. If router priorities tie, the router with the highest router ID (which is typically the router's IP address) is chosen as the designated router.

B.10 OSPF Authentication

All OSPF protocol exchanges are authenticated. The Authentication Type field available in the OSPF packet header identifies the authentication algorithm.[1] The authentication type is configurable on a per-interface basis. Three values are defined in the RFC 2328 standard:

Null Authentication Use of null authentication type means that routing exchanges over the network/subnet are not authenticated. The 64 bit Authentication field in the OSPF header can contain anything; it is not examined on packet reception. When null authentication is used, the entire contents of each OSPF packet (other than 64 bit Authentication field) are checksummed to detect data corruption.

Simple Password Authentication Using a simple password authentication type,

[1]Chapter 4: Routing Protocol Security, **Designing Network Security**, second edition, Cisco Systems.

a 64 bit field is configured on a per network basis. All packets sent on a particular network must have this configured value in their OSPF header. This essentially serves as a "clear" 64 bit password. In addition, the entire contents of each OSPF packet (other than 64 bit Authentication field) are checksummed to detect data corruption.

Plaintext authentication uses a shared secret key known to all the routers on the network segment. When a sending router builds an OSPF packet, it signs the packet by placing the key as plaintext in the OSPF header. The receiving router than compares the received key against the key in memory. If the keys match, the router accepts the packet. Otherwise, the router rejects the packet.

Cryptographic authentication In Cryptographic authentication, a shared secret key is configured in all routers attached to a common network/subnet. For each OSPF packet, the key is used to generate/verify a "message digest" that is appended to the end of the OSPF packet. Because the secret key is never sent over the network in the clear, protection is provided against passive attacks where intruders can eavesdrop on a network.

The algorithms used to generate and verify the message digest are specified implicitly by the secret key. Most implementations use the MD5 algorithm. To protect against replay attacks, a nondecreasing sequence number is included in each OSPF protocol packet. This provides long term protection; however resulting in OSPF replay attack until the sequence number changes. To implement this feature, each neighbor data structure contains a new field called the Cryptographic Sequence Number. This field is initialized to zero, and is also set to zero when the neighbor's state transitions to "down". Whenever an OSPF packet is accepted as authentic, the Cryptographic Sequence Number is set to the received packet's sequence number. Because the Cryptographic Sequence Number field is 32 bits in length, rollover issues should not be a problem unless a vendor has a particularly poor implementation.

B.11 OSPF Version 3

OSPFv3 is a modified version of OSPF that supports Internet Protocol version 6 (IPv6) addressing. OSPFv3 differs from OSPFv2 in the following ways:

1. All neighbor ID information is based on a 32-bit router ID.
2. The protocol runs per link rather than per subnet.
3. Router and network link-state advertisements (LSAs) do not carry prefix information.
4. Two new LSA types are included:
    (a) link-LSA and
    (b) intra-area-prefix-LSA.

5. Flooding scopes are as follows:
   (a) Link-local
   (b) Area
   (c) AS
6. Link-local addresses are used for all neighbor exchanges except virtual links.
7. Authentication is removed; the IPv6 authentication header relies on the IP layer.
8. The packet format has changed as follows:
   (a) Version number 2 is now version number 3.
   (b) The db option field has been expanded to 24 bits.
   (c) Authentication information has been removed.
   (d) Hello messages do not have address information.
   (e) Two new option bits are included: R and V6.

- Type 3 summary LSAs have been renamed inter-area-prefix-LSAs.
- Type 4 summary LSAs have been renamed inter-area-router-LSAs.

B.11.1 OSPFv3 Authentication

For providing inherent security to OSPFv3, AH/ESP extension headers (that is IPsec) is required. All OSPFv3 packets must be authenticated using either

AH or ESP and confidentiality can be used as an opinion.

OSPFv3 requires transport mode security associations to be used because the protocol packets are exchanged between routers that act like end hosts. ESP with NULL encryption in transport mode is required, which will provide authentication to only higher layer protocol data and not to the IPv6 headers, extension headers and options. AH in transport mode can optionally be provided and will provide authentication to higher layer protocols, selected version of the IPv6 header, selected portions of extension headers, and selected options. OSPF packets received in clear text or received with an incorrect AH integrity check value are required to be dropped when authentication is enabled.

HMAC MD5-96 must be implemented as the authentication algorithm and DES-CBC must be implemented as the encryption algorithm.

Bibliography

[1] Sadiq M. Sait and Habib Youssef. *Interative Computer Algorithms and their Applications*. IEEE Computer Society Press, Dec 1999.

[2] Bernard Fortz. On the Evaluation of Reliability of OSPF Routing in IP Networks. *http://www.poms,ucl.as.be/staff/bf/..IAG2601.pdf*, 2002.

[3] L. S. Buriol, M. G. CResende, C. C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing. June 2003.

[4] Oliver Heckmann. *The Competitive Internet Service Provider: Network Architecture, Interconnection, Traffic Engineering and Network Design*. Prentice Hall Series, 1992.

[5] Bernard Fortz, J. Rexford, and M. Thorup. Traffic Engineering with traditional IP routing protocols. *IEEE Communications Magazine*, pages 118–124, 2002.

[6] D. Awduche, A. Chiu, A. Elwalid, and X. Xiao. Overview and Priniciples of Internet Traffic Engineering. *RFC 3272*, May 2002.

[7] D. O. Awduche. MPLS and Traffic Engineering in IP Networks. *IEEE Communications Magazine*, pages 42–47, Dec 1999.

[8] X. Xiao, A. Hannan, B. Bailey, and L. Ni. Traffic Engineering with MPLS in the Internet. *IEEE Networks Magazine*, pages 28–33, Mar 2000.

[9] L. Qiu, Y. R. Yang, Y. Zhang, and S. Shenker. Selfish Routing in Internet like Environments. In *Proceedings of ACM SIGCOMM'03*, Karlsruhe, Germany, Aug 2003.

[10] M. Caesar and J. Rexford. BGP Routing Policies in ISP Networks. *Proceedings of IEEE Networks Magazine*, Nov 2005.

[11] T. C. Bressoud, R. Rastogi, and M. A. Smith. Optimal Configuration for BGP Route Selection. *Proceedings of IEEE INFOCOMM'02*, Jun 2002.

[12] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang. BGP Routing stability of popular destinations. In *Proceedings of Internet Measurement Workshop*, Marceille, France, Nov 2002.

[13] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancyn of Internet path properties. In *Proceedings of Internet Measurement Workshop*, San Francisco, CA, Nov 2001.

[14] R. Teixeira, S. Agarwal, and J. Rexford. BGP Routing Changes: Merging views from two ISPs. In *ACM SIGCOMM Computer Communications Review*, Oct 2005.

[15] S. Agarwal, C. N. Chuah, S. Bhattacharya, and C. Diot. The impact of BGP dynamics on intradomain traffic. In *Proceedings of Joint International Conference on Measurement and Modelling of Computer Systems (SIGMETRICS)*, NY, New York, Jun 2004.

[16] R. Teixeira, T. Griffin, A. Shaikh, and G. Voelker. Network Sensitivity to Hot-Potato Disruptions. In *Proceedings of ACM SIGCOMM'04*, Portland, OR, Aug 2004.

[17] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of Hot-Potato Routing in IP Networks. In *Proceedings of Joint International Conference on Measurement and Modelling of Computer Systems (SIGMETRICS)*, NY, New York, Jun 2004.

[18] J. Wu, Z. M. Mao, J. Rexford, and J. Wang. Finding a needle in haystack: Pinpointing significant BGP Routing changes in an IP Network. In *Proceedings of USENIX/ACM Symposium on Networked Systems Design and Implementation (NSDI'05)*, San Francisco, CA, May 2005.

[19] N. Feamster, J. Borkenhagen, and J. Rexford. Guidelines for Interdomain Traffic Engineering. In *ACM SIGCOMM Computer Communications Re-*

view, Oct 2003.

[20] N. Feamster and J. Rexford. Network wide BGP route prediction for traffic engineering. In *Proceedings of ITCOM*, Boston, MA, Aug 2002.

[21] N. Feamster, J. Borkenhagen, and J. Rexford. A model of BGP routing for network engineering. In *Proceedings of Joint International Conference on Measurement and Modelling of Computer Systems (SIGMETRICS)*, pages 331–342, NY, New York, Jun 2004.

[22] S. Agarwal, A. Nucci, and S. Bhattacharya. Measuring the shared fate of IGP Engineering and Interdomain traffic. In *Proceedings of* 13*th International Conference on Network Protocols(ICNP)'05*, Boston, MA, Nov 2005.

[23] T. Bressoud, R. Rastogi, and M. Smith. Optimal Configuration for BGP Route Selection. In *Proceedings of IEEE INFOCOMM'03*, San Francisco, CA, Apr 2003.

[24] C. Zhang, Z. Ge. J. Kurose, Y. Liu, and D. Towsley. On Optimal Routing with multiple traffic matrices: Tradeoff between average case and worst case performance. In *Proceedings of* 13*th International Conference on Network Protocols(ICNP)'05*, Boston, MA, Nov 2005.

[25] C. Zhang, Y. Liu, W. Gong, R. Moll, J. Kurose, and D. Towsley. On Optimal Routing with multiple traffic matrices. In *Proceedings of IEEE*

INFOCOMM'05, Maimi, FL, Apr 2005.

[26] Y. Zhang and Z. Ge. Finding critical traffic matrices. In *Proceedings of DSN'05*, Japan, Jun 2005.

[27] D. Bertsimas, D. Pachamanova, and M. Sim. Robust linear optimization under general norms. Technical report, Operations Research Letters, 2004.

[28] A. Elwalid, C. Jin, S. Low, and I. Widjaja. MATE: MPLS adaptive traffic engineering. In *Proceedings of IEEE INFOCOMM'01*, Anchorag, AK, Apr 2001.

[29] S. Kandula, D. Katabi, B. Davie, and A. Charny. Walking the tightrope: Responsive yet stable traffic engineering. In *Proceedings of ACM SIGCOMM'05*, Philadelphia, PA, Aug 2005.

[30] D. Applegate, L. Breslau, and E. Cohen. Coping with network failures: Routing strategies for Optimal demand oblivious restoration. In *Proceedings of Joint International Conference on Measurement and Modelling of Computer Systems (SIGMETRICS)*, NY, New York, Jun 2004.

[31] R. Zhang-Shen and N. McKeown. Designing a predictable Internet backbone network. In *Proceedings of Third Workshop on Hot Topics in Networks (HotNets-III)*, SanDiego, CA, Nov 2004.

[32] D. Applegate and E. Cohen. Making intradomain routing robust to changing and uncertain traffic demands: Understanding fundamental tradeoffs. In *Proceedings of ACM SIGCOMM'03*, Karlsruhe, Germany, Aug 2003.

[33] Bernard Fortz and Mikkel Thorup. Increasing Internet Capacity using Local Search. Technical report, Technical Report, IS-MG, 2000.

[34] D. O. Awduche, J. Malcolm, J. Agogbua, M. O'Dell, and J. Mc Manus. Requirements for Traffic Engineering over MPLS. Technical report, Technical Report, RFC 2702, 1999.

[35] Uyless Black. *IP Routing Protocols.* Prentice Hall Series, 2000.

[36] T. M. Thomas II. *OSPF Network Desing Solutions.* Cisco Press, 1998.

[37] J. T. Moy. *OSPF: Anatomy of Internet Routing Protocol.* Addison Wesley, 1999.

[38] F. Baker. Requirements for IP version 4 Routers. Technical report, Request for comments 1812, Jun 1995.

[39] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP source addres spoofing. Technical report, Request for Comments 2267, Jan 1998.

[40] Euglene L. Lawyer. *Combinatorial Optimization: Networks and Matroids.* Holt, Rinehart and Winston, 1976.

[41] Carlos A. S. Oliveira. Approximation Algorithms for Combinatorial Optimization. *ACM Symposium on Theory of Computing Archive*, Nov 2004.

[42] Youssef G. Saab and Vasant B. Rao. Stochastic Evolution: A Fast Effective Heuristic For Some Generic Layout Problems. *27th ACMIIEEE Design Automation Conference*, 1990.

[43] B. Sanso and F. Soumis. Communication and Transportation Network Reliabilty Using Routing Models. *IEEE Transactions on Reliability*, 40:29–37, 1991.

[44] C. Alaettinogly and S. Casner. Is-is routing on the qwest backbone: A recipe for subsecond isis convergence. NANOG,24, Feb 2002.

[45] Mukul Goyal, K. K. Ramakrishna, and Wu-Chi Feng. Achieving Faster Failure Detection in OSPF Networks. *IEEE International Conference on Communications*, 1:296–300, 2003.

[46] Bernard Fortz and Mikkel Thorup. Robust Optimization of OSPF/IS-IS Weights. *http://www.poms,ucl.as.be/staff/bfen/inoc.pdf*, 2000.

[47] B. Fortz and M. Thorup. Optimizing ospf/is-is weights in a changing world. volume 20 of *4*, pages 756–767. IEEE journal on selected areas in Communications, 2002.

[48] A. Nucci, B. Schroeder, S. Bhattacharya, N. Taft, and C. Diot. IGP link weight assignment for transient link failures. *to appear in ITC 18*, 2003.

[49] A. Markopoulo, G. Iannaccone, S. Bhattaacharya, C. Chuah, and C. Diot. Charaterization of Link Failures in an IP Backbone. INFOCOM, 2004.

[50] Ashwin Sridharan and Roch Guerin. Making IGP robust to link failures. *Technical report on Networking*, 2005.

[51] C. Alaettinoglu, V. Jacobson, and H. Yu. Toward millisecond IGP convergence. In *NANOG 20*, Oct 2000.

[52] A. Shaikh, L. Kalampoukas, R. Dube, and A. Varma. Routing Stability in Congested Networks: Experimentation and Analysis. In *Proceedings of ACM SIGCOMM*, Aug 2000.

[53] A. Basu and J. Riecke. Stability Issues In OSPF Routing. In *Proceedings of ACM SIGCOMM*, Aug 2001.

[54] P. Fraciosa, D. Frigioni, and M. Giaccio. Semi-dynamic shortest paths and breadth first search in digraphs. In *Proceedings of 14th Symp. Theoretical Aspects of Computer Science*, pages 113–124, 1997.

[55] D. Frigioni, M. Ioffreda, U. Nanni, and G. Pasqualone. Experimental analysis of dynamic algorithms for the single source shortest path problem. In *ACM Journal of Experimental Algorithmics*, volume 3 of *5*, 1998.

[56] G. Ramalingam and T. Reps. An incremental algorithm for a generalization of the shortest path problem. In *Journal of Algorithms*, volume 21 of *2*, pages 267–305, 1996.

[57] Mikkel Thorup. Fortifying OSPF/IS-IS against Link Failure. *NATO ASI Series: Ser.F, Computer and System Sciences, Springer-Verlag*, 2001.

[58] P. Narvaez, K. Y. Siu, and H. Y. Tzeng. Local Restoration Algorithm for Link State Routing Protocols. ICCCN, 1999.

[59] Y. Liu and A. L. Narsimha Reddy. A fast rerouting scheme for ospf/is-is networks. Technical report, AM University, Dept of Electrical Engineering, Texas, AM University, 2004.

[60] D. Awduche, A. Chiu, A. Elwalid, and X. Xiao. A Framework for Internet Traffic Engineering. Network working group. *RFC 3272*, 2000.

[61] F. Y. S. Lin and J. L. Wang. Minimax OSPF Routing Algorithms in Nnetworks Supporting the SMDS Services. *Proceedings of IEEE International Conference Communications(ICC)*, 2:666–670, 1993.

[62] H. Frank and I. T. Frisch. *Communication, Transmission and Transportation Networks.* Addison Wesley, 1971.

[63] H. Frank and W. Chou. Network Properties of the ARPA Computer Network. *Networks*, 4:213–239, 1974.

[64] H. Frank, R. E. Kahn, and L. Klienrock. Computer Communication Network Design - Experience with Theory and Practice. *AFIPS Proceedings*, 40:255–270, 1972.

[65] Bernard Fortz. On the evaluation of the reliability of OSPF routing in IP networks. *Proc. of SSGRR 2002w International Conference-Advances in Infrastructure for e-Business, e-Education, e-Science and e-Medicine on the Internet*, 2002.

[66] Dmitri Bertsekas and Robert Gallager. *Data Networks.* Prentice Hall Series, 1992.

[67] E. Dijkstra. A node on two problems in connection of graphs. *Numerical Mathematics*, 1959.

[68] L. S. Buriol, M. G. C. Resende, C. C. Ribeiro, and M. Thorup. A Hybid Genetic Algorith for the Weight Setting Problem in OSPF/IS-IS Routing. *Networks*, 46(1):36–56, 2005.

[69] E. W. Zegura. Gt-itm:georgia tech internetwork topology models (software). http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm/gt-itm.tar.gz, 1996.

[70] K. Calvert, M. Doar, and E. W. Zegura. Modeling internet topology. volume 35 of *6*, pages 160–163. IEEE Communications Magazine, 1997.

[71] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to model an Internetwork. *Proc.15th IEEE Conf on Computer Communications(INFOCOM)*, pages 594–602, 1996.

[72] P. L. Toint. Transportation Modelling and Operations Research: A fruitful connection. *NATO ASI Series: Ser.F, Computer and System Sciences, Springer-Verlag*, 166:1–27, 1998.

[73] Mohammed H. Sqalli, Sadiq M. Sait, and Mohammed Aijaz Mohiuddin. An Enhanced Estimator to Multi-Objective OSPF Weight Setting Problem. *IEEE/IFP Network Operations and Mangement Symposium (NOMS)*, Apr 2006.

[74] Sadiq M. Sait, Mohammed H. Sqalli, and Mohammed Aijaz Mohiuddin. Engineering Evolutionary Algorithm to Solve Multi-Objective OSPF Weight Setting Problem. *Australian Conference on Aritificial Intelligence*, pages 950–955, 2006.

[75] V. Verstraete, M. Strobbe, E. Van Breusegem, Jan Coppens, M. Pickavet, and Piet Demeester. AntNet: ACO Routing Algorithm in Practice. *Proceedings of the 8E Informs Telecommunications Conference.*

Vitae

- Shaik Muzibur Rehman
- Received Bachelor of Technology (B.Tech)degree in Electronics & Communications Engineering from Jawaharlal Nehru Technological University, Hyderabad,India in 2001
- Completed M.S. degree requirements in Computer Engineering at KFUPM, Saudi Arabia in 2007