Preventive Start-Time Network Optimization for Energy Saving against Link Failure with Unpredictable Traffic Demand

RAVINDRA SANDARUWAN **RANAWEERA**
1231105
INFORMATION AND COMMUNICATION SYSTEMS PROGRAM
DEPARTMENT OF COMMUNICATION ENGINEERING AND INFORMATICS
THE UNIVERSITY OF ELECTRO-COMMUNICATIONS

A DISSERTATION PRESENTED TO THE
DEPARTMENT OF COMMUNICATION ENGINEERING AND INFORMATICS
IN CANDIDACY FOR THE MASTER'S DEGREE

RECOMMENDED FOR ACCEPTANCE BY THE
DEPARTMENT OF COMMUNICATION ENGINEERING AND INFORMATICS

ADVISORS: EIJI **OKI** AND NAOTO **KISHI**
PROFESSORS, UEC
JANUARY 27, 2014

専攻主任印	主指導教員印	指導教員印

修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院　情報理工学研究科　情報・通信工学専攻　博士前期課程		
氏　　名	ラヴィンドラ　サンダルワン　ラナヴィーラ Ravindra Sandaruwan Ranaweera	学籍番号	1231105
論 文 題 目	Preventive Start-Time Network Optimization for Energy Saving against Link Failure with Unpredictable Traffic Demand (不確かなトラヒック需要に対するリンク故障を考慮したネットワークエネルギーの最適化)		

要　旨

インターネットの普及により、ネットワークに流れるトラヒック量が増加し、ネットワーク管理者が抱えている問題が複雑になりつつある。ネットワークのエネルギー消費の増加、リンク故障の対応やトラヒックの変動が大きな問題である。本論文では、不確かなトラヒック需要に対するリンク故障を考慮し、OSPF (Open Shortest Path First)リンクウエイトを最適化することでネットワークエネルギーを最小化する方式について提案する。

ルーチングに必要なネットワークリソースを最小化することによってネットワークのエネルギー消費が低減できる。利用するネットワークリソースを最小化するように OSPF ネットワークの経路を選択すればエネルギー消費の低減になる。OSPF ネットワークでは、各リンクに割り当てられているリンクウエイトの合計が最小となるような経路を選んでデータ通信を行っている。リンクウエイト最適化問題は NP ハード問題で[10]、近似解が得られるような発見的手法が必要である。

スタートタイム最適化（SO: Start-time Optimization)、ランタイム最適化（RO: Run-time Optimization)、予防的なスタートタイム最適化（PSO: Preventive Start-time Optimization）は主なリンクウエイト最適化ポリシーである。SO はネットワーク故障に弱い、RO はネットワークを不安定にさせるといった欠点があるため、予めリンク故障を想定し、最悪な場合に対応させる PSO が提案された。これらのポリシーはすべてのノード間のトラヒック需要が正確に分かっているパイプモデルに対して提案されているが、すべてのノード間のトラヒック需要の推測、測定が非常に困難である。従って、各ノート間のトラヒック需要が正確に分からなくても、各ノードから出入りするトラヒック量だけが分かっている、フレキシブルなホースモデルが提案されている。

本論文では、PSO ポリシーをネットワークリソースの利用を最小となるよう、ホースモデルに適用する方式を提案する。この発見的手法は、最悪パフォーマンストラヒック需要を選択し、リンク故障を考慮しながらリンク利用率の合計が最小となるようにリンクウエイトを選ぶ。提案方式は、最悪パフォーマンストラヒック需要を選択してリンクウエイト空間を検索しているのでブルートフォース方式よりも計算量が少なくなる。数値計算結果により、提案方式が他の方式に比べ、利用するネットワークリソース低減に適していることが分かる。

Preventive Start-Time Network Optimization for Energy Saving against Link Failure with Unpredictable Traffic Demand

RAVINDRA SANDARUWAN **RANAWEERA**

1231105

A DISSERTATION PRESENTED TO THE
DEPARTMENT OF COMMUNICATION ENGINEERING AND INFORMATICS
IN CANDIDACY FOR THE MASTER'S DEGREE

RECOMMENDED FOR ACCEPTANCE BY THE
DEPARTMENT OF COMMUNICATION ENGINEERING AND INFORMATICS

ADVISORS: EIJI **OKI** AND NAOTO **KISHI**

PROFESSORS, UEC

JANUARY 27, 2014

Contents

Ravindra Sandaruwan Ranaweera
January 2014

Abstract

With the rapid development in emerging Internet applications traffic on backbone networks is increasing every second leaving the network operators with challenging traffic engineering problems. Network energy consumption, link failure, and traffic demand uncertainty are some of the major problems. This work studies how to relax network energy consumption increase, link failure, and traffic uncertainty by optimizing OSPF (Open Shortest Path First) link weights.

The network energy consumption can be reduced by minimizing the total network resources used for routing. By implementing the most appropriate set of routes, we can minimize the total network resource usage. In OSPF, link weights are used to find the routes from all sources to destinations. Optimizing link weights in an OSPF network is a challenging traffic engineering problem to reduce the total network resources used.

Most previous studies have focused on the application of Start-time Optimization (SO), Run-time Optimization (RO), and Preventive Start-time Optimization (PSO) on pipe model where the exact traffic demand between each source and destination pair needs to be specified. In reality, measuring or predicting the exact traffic demand is almost impossible. Therefore, a more flexible model called hose model was introduced. In the hose model the total ingress and egress traffic is known which makes it more flexible for traffic demand fluctuations.

This dissertation proposes a PSO policy for the hose model to optimize link weights against link failures aiming at reducing the total network resources used. The proposed scheme employs a heuristic algorithm to determine a suitable set of link weights to reduce the sum link utilizations for any single link failure. It efficiently selects the worst-case performance traffic matrix and reduces the worst-case total network resource usage as compared to a brute-force scheme that is computationally expensive when searching the link weight space against all the possible traffic matrices and topologies created by single link failures. The numerical results show that the proposed scheme is more effective in the reduction of worst-case total network resource usage than the schemes utilizing SO or minimum-hop routing.

Acknowledgments

I would like to express my gratitude to my advisor Eiji OKI, professor at the University of Electro-Communications for his invaluable guidance and advices. It was quite comfortable working with him since he allowed me to conduct the research in my own pace. Also, my heartiest gratitude goes to my sub-advisor Naoto KISHI, professor at the University of Electro-Communications for his guidance.

I would like to acknowledge all the advices and guidance by Mr. Kamrul Islam. His research experiences was a great deal of help. And I would also like to thank my fellow lab members for their suggestive ideas and cooperation. Without them this work would not have been possible.

Also, I would like to extend my gratitude to *Tokyu Foundation for Foreign Students* for the support they have given me providing a scholarship, which pave the way to concentrate on my research.

Finally, I would like to thank my friends all over the world. They have always been my family more than just friends. And at last but not the least, my parents and family members for the support and encouragement they give.

CHAPTER 1

Introduction

1.1. Routing

In the Internet, most forwarding is done as hop-by-hop. This means that each router is only responsible for forwarding a datagram to another router. This process continues until the datagram reaches its destination or times-out because its path is too long. Random-walk routing techniques are not particularly reliable, and so it is important that the routers in a network have a coordinated approach to deciding which is the next hop along the path to a destination. The distance vector and path vector techniques essentially pass information between neighboring routers and use this data to build the shortest paths to all destinations. These are then installed in a routing table and passed on to the router's neighbors. The path computation model deployed in distance vector protocols and path vector protocols is iterative and distributed.

Link state protocols ensure that all routers in the network have the same view of the network's resources, but do not distribute path information. It is the responsibility of each router to use the link state information that it receives to calculate the shortest paths through the network.

1.2. Open Shortest Path First (OSPF)

OSPF [2] is an interior gateway protocol that routes Internet Protocol (IP) packets solely within a single routing domain (autonomous system). It gathers link state information from available routers and constructs a topology map of the network. The topology determines the routing table presented to the Internet Layer which makes the routing decisions based solely on the destination IP address found in IP packets.

OSPF detects changes in the topology, such as link failures, very quickly and converges on a new loop-free routing structure within seconds. It computes the shortest path tree for each route using a method based on Dijkstra's algorithm, a shortest path first algorithm.

1.3. Computing Paths

Before discussing how to select the shortest paths across a network, we must establish what we mean by "shortest". As we have seen in distance vector protocols and link state protocols, the advertised connecitivity is associated with a metric as shown in Table 1. In most simple cases this metric is set to 1 for each hop or link, so the total matrix for a path is the number of hops. This paths computed utilizing this method is called minimum-hop (min-hop) routing. It is possible, however, to set the metric for any one hop to be greate than 1; this increases the distance, as it were, between a pair of adjacent router. Another way to look at this is to say that an increased cost has been assigned to the link. In this case, when the total metric for a path is computed, it is the sum of the link metrics along the path.

Open Shortest Path First (OSPF) computation selects the least cost path across a network from source to destination using only active links. The Dijkstra algorithm [4] is a relatively efficient and significantly simple way to build a routing table from a set of link state information for every node. It is an important point that the algorithm fully populates the routing table in one go by simultaneously calculating the shortest paths to all destinations.

The Internet is a collection of nodes and links with the purpose of delivering IP datagrams from a source node to a destination node. The source does not, in general, care how the data is delivered as long as it arrives in a timely and a reliable way. In SPF (Shortest Path First) routing, each datagram is routed on the shortest path between the source and the destination.

In an otherwise unused network, SPF routing is ideal: datagrams are delivered expeditiously with the least use of network resources. However, as network traffic increases it may be the case that a link or router is saturated and cannot handle all of the traffic that it receives. When this happens, data will be lost either in a random way through data collisions, or through an organized scheme within routers.

Traffic engineering (TE) is the process of predetermining the path through the network that various flows of data will follow. These paths can diverge from the shortest paths for a variety of reasons, including operator choice or the application of constraints to the routing

decision. Fundamental to the way TE works is the fact that packets are not routed at each router in the network, but the path is specified from the outset and provided as a series of IP addresses (links) that the traffic must traverse.

The choice of path computation is forced by the decision to use a distance vector, path vector or link state routing protocol. The first two employ a distributed computation technique, but link state protocols require some form of full computation on each node. The advantages of the link state approach are that the protocol has a full view of the network and can perform sophisticated policy based routing, and even compute paths to remote nodes. On the other hand, the amount of information required can quickly grow large, which may place an effective limit on the size of the network that can be supported. At the same time, the complexity of the path computation does not scale well and needs a serious CPU in a large network.For the majority of applications, a simple shortest path first approach provided by any of the three routing techniques is adequate. Each can be enhanced by simple metrics to skew the cost of traversing specific links.

In OSPF routing there are costs assigned to all of the links. This link cost (link weight) can be used to introduce traffic engineering in OSPF routing model. The link weights must be calculated considering network topology, network resources etc.. So far, the link weight calculating policies were developed only considering the network congestion. An optimal set of link weights would reduce the network congestion ratio which will allow more traffic to be entered into the network [5].

With the development in emerging Internet applications, such as Voice-over-IP (VoIP), teleconference, video-on-demand, has led to a rapid increase in the energy consumption of the backbone networks. Recent studies have introduced policies to optimize the link weights considering the network energy consumption while guaranteeing quality of service (QoS).

1.4. Network Energy Consumption

With the development in emerging Internet applications, such as Voice-over-IP (VoIP), teleconferences, video-on-demand, and multi- party games, the data on backbone networks has increased rapidly. This has led to the growth in network energy consumption with significant economic and environmental impacts. The rapid increase in backbone energy consumption and network failures are two main challenges network operators face today.

Based on the report in [6], the global amount of energy needed by Internet in 2007 has been 5.5% of the total electricity consumption in the world. The increase rate is estimated around 20-25% per year. The total power consumption of routers in Japan was 7.3 TW-h in 2004 which increased to 13 TW-h in 2010 [7]. Though the core network equipments currently occupy 20% of the total energy consumption, it is estimated to be increasing dramatically in the future [8]. The energy consumption in backbone and aggregation networks will soon become the same amount as it is now consuming in access networks [9]. In [9], they estimate the energy consumption of IP networks in 2017 will be twelve times of the energy consumption in 2009. By implementing energy aware traffic engineering in backbone networks it is possible to reduce the energy consumed by backbone networks.

CHAPTER 2

Link Weight Optimization

2.1. Link Weight Optimization

In commonly used packet routing protocols in communication networks, such as OSPF (Open Shortest Path First) in IP networks, each link in the network is assigned a weight. Based on weights of links in the network, routing protocols find the end-to-end path for each source-destination pair such that the sum of link weights on the path is minimized, so called the shortest path. Hence, to find a 'good' path for each source and destination pair that satisfies some network constraints such as the quality of service (QoS) of each session, energy consumption, and the target utilization of each link, it is important to assign the appropriate weight for each link in the network. A simple default weight setting policy suggested by CISCO [3] is to make the weight of a link inversely proportional to its capacity. This setting of weight makes the more capable link to be used the most in the network. Which means more the capability of a link is greater the network uses that link. But this simple weight setting policy do not emphasize on traffic matrix (traffic demand) or required traffic over link. So still this does not give the most optimal performance for the oblivious routing.

A more efficient way is optimizing link weights for a given topology considering the traffic over the network which will reduce a certain objective function. This scheme is called as Start-time Optimization (SO). Optimization provides network operators with a powerful method for traffic engineering. Its general objective is to distribute traffic flows evenly across available network resources in order to avoid network congestion and quality of service (QoS) degradation. The simple method is to check for every possible set of link weights for all the links and calculate an objective function and then set the weight for which the objective function is minimized. But the time complexity of this straight-forward algorithm is x^m, where x is the higher limit of weights and m is the number of links in the network. As this is a high complexity algorithm many researches have been done to

find out a heuristic algorithm usable for larger network instances. B. Fortz and M. Thorup has found a heuristic algorithm to reduce the time complexity [10]. In this study they uses Tabu Search [11] to find out the optimal weight for links.

2.2. Tabu Search

The abundance of difficult optimization problems encountered in practical settings such as telecommunications, logistics, financial planning, transportation, and production has motivated the development of powerful optimization techniques. These techniques are usually the result of adapting ideas from a variety of research areas. The philosophy of Tabu Search (TS) is to derive and exploit a collection of principles of intelligent problem solving. In this sense, it can be said that TS is based on selected concepts that unite the fields of artificial intelligence and optimization.

Tabu search is a meta-heuristic local search algorithm that can be used for solving combinatorial optimization problems (problems where an optimal ordering and selection of options is desired - an example, the traveling salesman problem). Tabu search uses a local or neighborhood search procedure to iteratively move from one potential solution x to an improved solution x' in the neighborhood of x, until some stopping criterion has been satisfied (generally, an attempt limit or a score threshold). Local search procedures often become stuck in poor-scoring areas or areas where scores plateau. In order to avoid these pitfalls and explore regions of the search space that would be left unexplored by other local search procedures, Tabu Search carefully explores the neighborhood of each solution as the search progresses. The solutions admitted to the new neighborhood, N^x, where N is the number of nodes, are determined through the use of memory structures. Using these memory structures, the search progresses by iteratively moving from the current solution x to an improved solution x' in N^x.

These memory structures form what is known as the Tabu list, a set of rules and banned solutions used to filter which solutions will be admitted to the neighborhood N^x to be explored by the search. In its simplest form, a Tabu list is a short-term set of the solutions that have been visited in the recent past (less than n iterations ago, where n is the number of previous solutions to be stored - n is also called the tabu tenure).

Tabu search can be used to find a satisfying solution for the traveling salesman problem (that is, a solution that satisfies an adequacy criterion, rather than the absolutely optimal solution). First, the Tabu Search starts with an initial solution, which can be generated randomly or according to some sort of nearest neighbor algorithm. To create new solutions, the order that two cities are visited in a potential solution is swapped. The total traveling distance between all the cities is used to judge how ideal one solution is compared to another.

Figure 2.1 shows an example of how to get the optimal link weights using Tabu Search. As the initial conditions link weights shown in Figure 2.1(a) are given. According to those link weights we can decide the shortest paths from nodes $1, 2, 3, 7$ to 6. As shown in Figure 2.1(a) the link between nodes $4 - 6$ is congested. We increase its link weight until a new routing is achieved. When the link weight of nodes $4 - 6$ is increased to 4, a new routing is achieved. We again calculate the shortest paths, find the congested link, and increase its weight. This process is repeated until an optimal set of link weights is achieved.

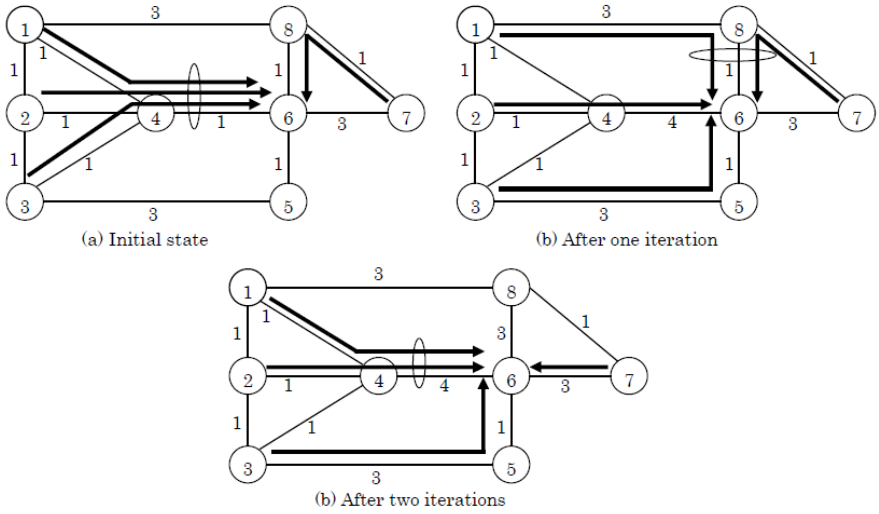

FIGURE 2.1. Link weight optimization using tabu search

2.3. Simulated Annealing

Simulated annealing [12] is also a method to used to solve optimization problems in a large search space. It is often used when the search space is discrete (e.g., all tours that visit a given set of cities). For certain problems, the simulated annealing may be more efficient than other methods. The simulated annealing algorithm was originally inspired

from the process of annealing in metal work. Annealing involves heating and cooling a material to alter its physical properties due to the changes in its internal structure. As the metal cools its new structure becomes fixed, consequently causing the metal to retain its newly obtained properties. In simulated annealing we keep a temperature variable to simulate this heating process. We initially set it high and then allow it to slowly 'cool' as the algorithm runs. While this temperature variable is high the algorithm will be allowed, with more frequency, to accept solutions that are worse than our current solution. This gives the algorithm the ability to jump out of any local optimums it finds itself in early on in execution. As the temperature is reduced so is the chance of accepting worse solutions, therefore allowing the algorithm to gradually focus in on a area of the search space in which hopefully, a close to optimum solution can be found. This gradual 'cooling' process is what makes the simulated annealing algorithm remarkably effective at finding a close to optimum solution when dealing with large problems which contain numerous local optimums. The nature of the traveling salesman problem makes it a perfect example.

The simulated annealing is proved to give better performance over other optimization solving methods, such as, simple hill climber. Although hill climbers can be surprisingly effective at finding a good solution, they also have a tendency to get stuck in local optimums. The simulated annealing algorithm is excellent at avoiding this problem and is much better on average at finding an approximate global optimum.

A hill climber algorithm will simply accept neighbor solutions that are better than the current solution. When the hill climber cannot find any better neighbors, it stops.

FIGURE 2.2. Hill climber algorithm

In the example shown in Figure 2.2 the hill climber is started at the red arrow and it works its way up the hill until it reaches a point where it can't climb any higher without first

descending. In this example we can clearly see that it's stuck in a local optimum. If this were a real world problem we wouldn't know how the search space looks so unfortunately we wouldn't be able to tell whether this solution is anywhere close to a global optimum.

Simulated annealing works slightly differently than this and will occasionally accept worse solutions. This characteristic of simulated annealing helps it to jump out of any local optimums it might have otherwise got stuck in.

To understand how simulated annealing is able to avoid these local optimus, it is important to take a look at how the algorithm decides which solutions to accept. First we check if the neighbor solution is better than our current solution. If it is, we accept it unconditionally. If however, the neighbor solution is not better we need to consider a couple of factors. Firstly, how much worse the neighbor solution is; and secondly, how high the current 'temperature' of our system is. At high temperatures the system is more likely accept solutions that are worse. Basically, the smaller the change in energy (the quality of the solution), and the higher the temperature, the more likely it is for the algorithm to accept the solution.

2.4. Network Failure

There are many reasons for which a network node may out of order. A node may be go down or a link can be inactive and can not use. These are called node failure and link failure, respectively. Only making a routing scheme smarter is not the solution of making the routing stronger. Network failure must be considered when new smarter routing schemes are studied. A recent study of routing updates in the Sprint's IP Backbone network shows that 80% of all failures are unexpected and of those 70% affect a single link at a time. Network disruption is of three types: node failure, link failure and link congestion. Every network has a mechanism to recover these failures. From the End-user point of view this recovery process should be fast enough so that the service interruption time is either unnoticeable or minimal. As the Internet applications are growing fast, network failure occurs frequently and recovery is the main challenge to establish a sound network.

When link failure occurs traffic related to that link should be re-routed through other active links, which can create an excess load to any other link and causes congestion in the network. Link failure is considered as a failure but if we consider it as a Topology change

in the network, we can take precautions for this problem in routing. When a link fails, the network takes a new shape. So we can take topology as a variable which can change with time because of link failure. For every possible link failure we can get a new topology. If we consider a network topology as shown in Figure 2.3, all the new topologies created by single link failure will be shown as Figure 2.4.

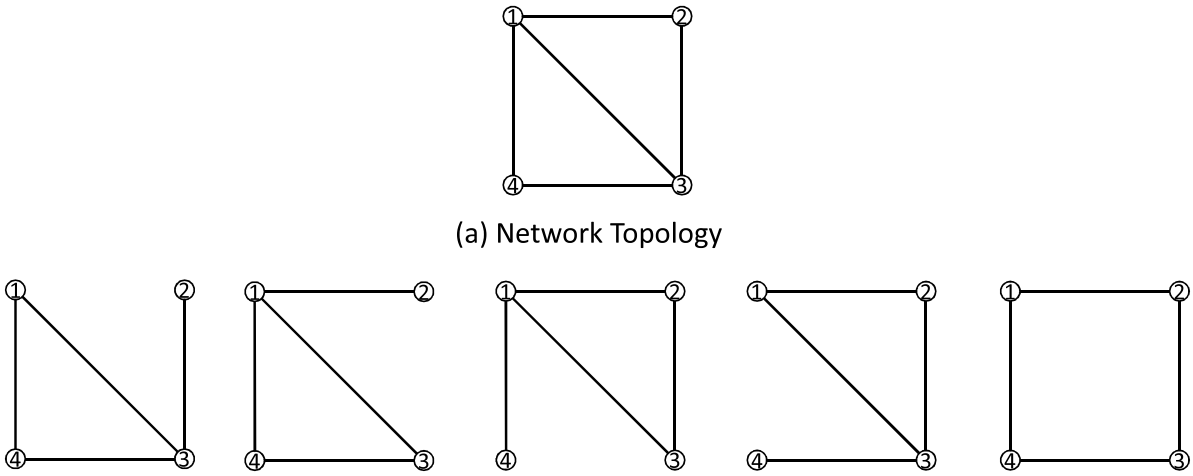

FIGURE 2.3. A simple network topology and topologies after single link failure

2.5. Link Weight Optimization against Link Failure

As discussed earlier, finding an appropriate link weight set can reduce network congestion, total resources used, power consumption etc.. There are three major link weight optimization policies used at present. They are,

- Start-time Optimization (SO)
- Run-time Optimization (RO)
- Preventive Start-time Optimization (PSO).

These policies are introduced to overcome the weaknesses other policies have.

2.6. Start-time Optimization (SO)

A start-time optimization scheme optimizes a set of OSPF link weights considering the topology information based on given traffic demands under the condition that no link failure occurs. However, SO is not able to cope with network topology changes caused by network

failures, such as link failures. Whenever a link failure occurs, each router needs to reroute the data packets in order not to lose any packets, which may lead to unexpected network congestion or increase in network resources used. SO considers the link weight assignment problem as a static problem and does not consider dynamic network topology changes at run time. Thus, link failures are considered to be one of the main challenges faced by network operators because link failures occur on a daily basis in large IP backbones [13].

2.7. Run-time Optimization (RO)

The weakness of SO can be overcome by computing a new optimal set of link weights whenever the topology is changed. It can be said that this approach, called Run-time Optimization (RO) provides the best routing performance after link failure. However, updating link weights in any case may not be practical for two reasons. The first reason is that changing link weights frequently causes network instability. In order to recompute the shortest paths to all the other routers, the updated link weights should be flooded to all routers in the network. As a result, the performance of Transport Control Protocol (TCP) connections may be degraded due to the arrival of packets in disorder [14]. The network will take longer time to achieve stability if link weights are updated often, because packets are sent back and forth between routers due to frequent updates to the routing table. The second reason lies in the short-lived nature of most link failures. As mentioned in [13], 80% of the link failures in their tested networks last less than 10 minutes, and 50% of the link failures last less than one minute. They thus define transient failures as those failures that last for not more than 10 minutes. Transient failures can create rapid congestion that is harmful to the network. However, they leave the human operator with insufficient time to reassign link weights before the failed link is restored. The study in [13] also examined the frequency of single-link and multiple-link failures. They observed that more than 70% of transient failures that lasted less than 10 minutes are single link failures. Therefore, it seems reasonable to target the one-time configuration of link weights that can handle any link failure.

So, the question is whether there is a scheme that -can give the RO performance but -avoid network instability. Kamrul et al. presented a link weight optimization policy called Preventive Start-time Optimization (PSO) [15] [16] that can answer this question.

2.8. Preventive Start-time Optimization (PSO)

PSO is a scheme that determines, at the start time, a suitable set of link weights that can handle any single link failure scenario preventively. The objective of this scheme is to determine, at the start time, the most appropriate set of link weights that can avoid both unexpected network congestion and network instability, the drawbacks of SO and RO, regardless of which link fails. PSO considers all possible single link failure scenarios at start time in order to determine a suitable set of link weights.

However, the current PSO scheme is usable when the traffic demand of the network is known. This traffic model is called a *pipe model*. But in general, the hose model is proved to be very difficult for network operators to use [17] [18].

2.9. Equal Cost Multipath (ECMP) Routing

Current routing schemes typically focus on discovering a single "*optimal*" path for routing, according to some desired metric. Accordingly, traffic is always routed over a single path, which often results in substantial waste of network resources. Equal Cost MultiPath (ECMP) routing is an alternative approach that distributes the traffic among several "*equal cost*" paths instead of routing all the traffic along a single "*best*" path.

ECMP routing can be fundamentally more efficient than the currently used single path routing protocols. It can significantly reduce congestion in "*hotspots*", by deviating traffic to unused network resources, thus improving network utilization and providing load balancing [19]. Moreover congested links usually result in poor performance and high variance. For such circumstances, ECMP routing can offer steady and smooth data streams [11].

2.9.1. ECMP Routing for PSO. PSO considers single link failure scenarios and optimize the link weights for the worst-case scenario. To show that PSO is able to reduce the worst-case congestion ratio, Figure 2.4 (a) shows a network topology with link weights that were optimized using SO and Figure 2.5 (a) shows the same network with link weights optimized using PSO. To simplify this situation let us consider the traffic demand is given and it is 1 unit from node 1 to node 3 and 1 unit from node 4 to node 3. Figures 2.4 (b) and (c) show the shortest paths for each traffic demand, respectively for no link failure. We consider that this network supports ECMP routing and all link capacities are equal to 1

unit. The links (1,3), (1,2), (2,3) are utilized maximumly, which means that the congestion ratio r is 0.75 when ECMP routing is utilized. But, if this network does not support ECMP routing the congestion ratio will be 1 or higher. This shows that the networks utilizing ECMP routing are able to reduce the network congestion compared to networks that do not utilize ECMP routing. If we consider all the single link failure scenarios (5 for this network topology), Figures 2. 4 (d), (e), (f) show the worst-case scenarios. In case of single link failure, the worst-case congestion ratio r is 1.5 for this network topology. On the other hand, for the network utilizing PSO link weights, the worst-case congestion ratio r is 1, as shown in Figures 2.5 (d), (e), (f), (g) and (h), even though the congestion ratio for no link failure is 0.75. This proves that the set of link weights optimized using PSO policy is able to reduce the worst-case congestion ratio when there is a single link failure in the network.

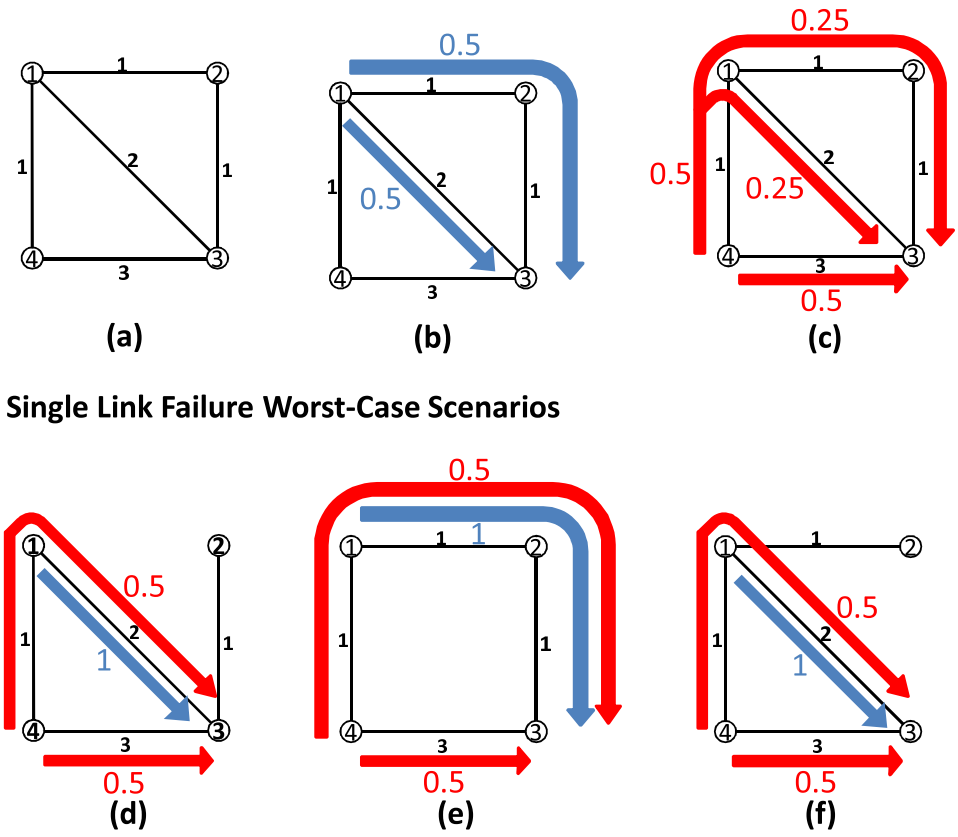

FIGURE 2.4. Routing in a network using SO link weights with ECMP

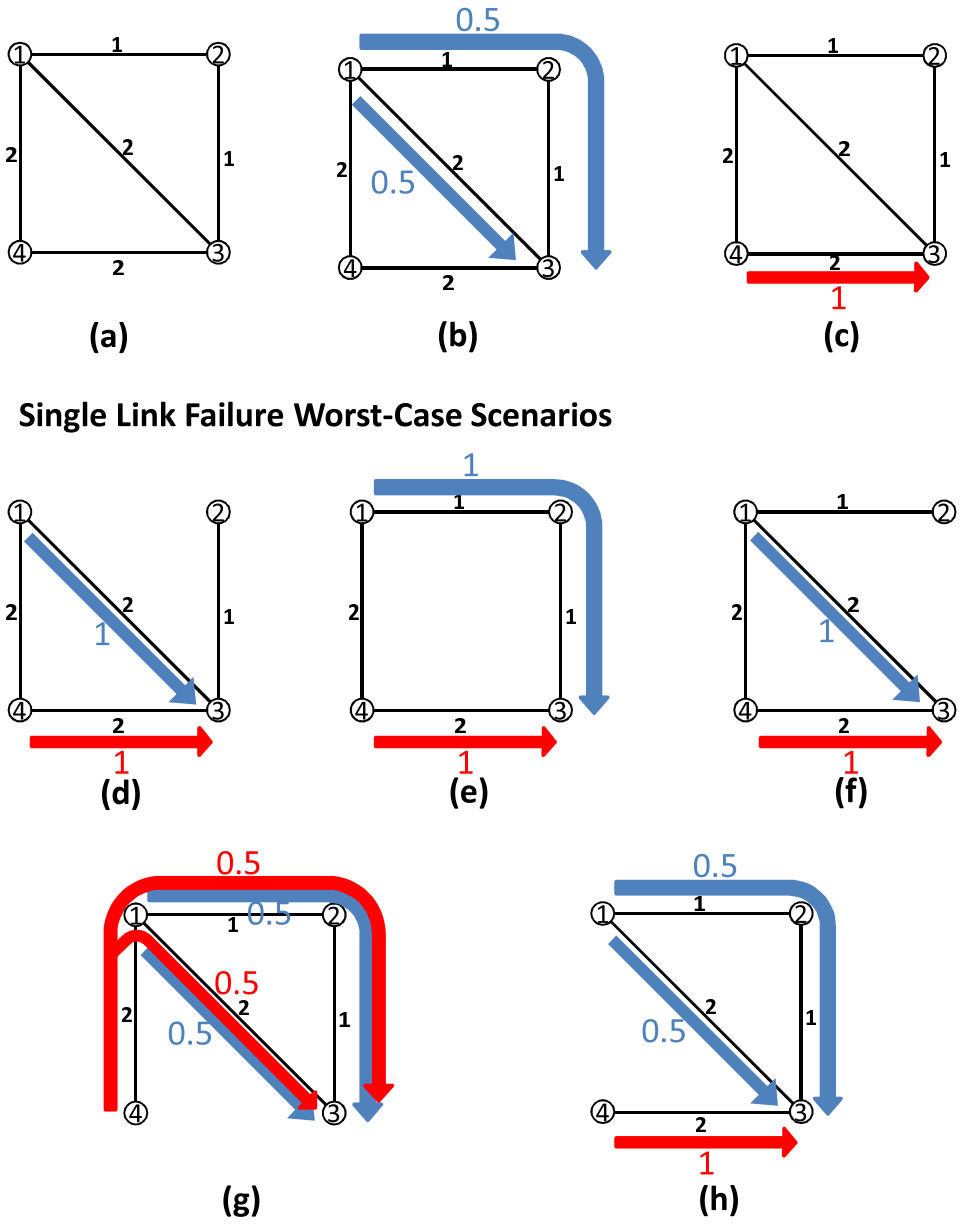

FIGURE 2.5. Routing in a network using PSO link weights with ECMP

CHAPTER 3

Hose Model/Pipe Model

In terms of bandwidth specification, networks can be divided into a *pipe model* and a *hose model*.

3.1. Pipe Model

The pipe model needs to specify the bandwidth requirement between any two nodes. It means that the entire traffic demand (traffic matrix) of the network is given. In the pipe model, customers must have precise predictions in advance about the complete traffic demand of each node pair in a network. Then the network service provider finds a data transmission path for traffic between each node pair. However, customers may be unwilling, or unable, to know the traffic demand of each node pair in a network. This is specially true when the number of nodes per network is large. Figure 3.1 shows an example of the pipe model and its traffic demand.

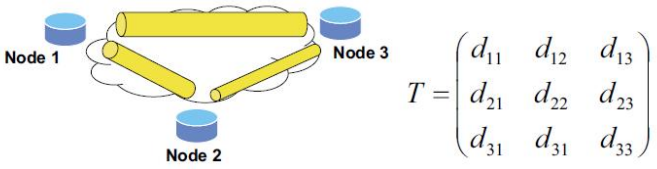

FIGURE 3.1. Pipe model and its traffic demand

The PSO scheme introduced in the previous chapter is based on the pipe model. They calculated the optimal link weights using a given traffic matrix.

3.2. Hose Model

Since the prediction of the actual traffic requirement as required in the pipe model is difficult, it is, therefore, considered to be easier for network operators to specify the traffic as just the total ingress (α_i) and egress traffic (β_i) (i.e., the amount of traffic that can be sent to and received from the backbone network) at each node. The traffic model that has

achieved this specification is called a *hose model*. Chu et al. formulated the general routing problem of the hose model and presented an algorithm for solving the link weight searching problem in [21]. The hose model presents some challenging problems for traffic engineering as the hose model only needs to specify the total ingress bandwidth requirement and the total egress bandwidth requirement at each node. Chu et al. in [21] presented an algorithm to minimize weight changes caused by link failures. As explained before, even one link weight change may cause network instability. An example of the hose model is shown in Figure 3.2. The ingress and egress bandwidth requirement is as Equation 3.1.

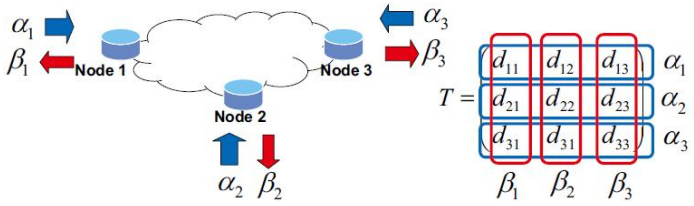

FIGURE 3.2. Hose model and its traffic constraints

$$\sum_j d_{ij} \leq \alpha_i \tag{3.1a}$$

$$\sum_i d_{ij} \leq \beta_j \tag{3.1b}$$

Many hose model provisioning algorithms have been proposed for MPLS-based networks and the goal is to maximize the total amount of traffic admissible to the network. If link weights are given, the shortest paths are determined. Thus the hose model network provisioning problem in an IP-based public network is equivalent to a link weight setting problem. Chu et al. in [21] presented a mixed-integer programming formulation to compute the optimal weights that can maximize the ingress and egress network traffic with bandwidth guarantees admissible to a hop-by-hop routing network.

3.3. Target of this Research

As explained in earlier chapters, network failures, uncertain traffic demand and backbone power consumption are major challenges network operators face today. The future routing mechanisms required to be resilient for traffic demand, network failure and use

minimum energy. This research studies how to reduce the energy consumption in backbone networks under link failure and uncertain traffic demand by optimizing link weights. This study is divided into two parts.

- Resilient routing: Preventive start-time link weight optimization to counter link failures for hose model.
- Energy saving routing with resilient routing: Reduce the energy consumption of backbone network by evenly splitting the data between the links in the network while considering link failure for hose model.

In the following sections PSO policy for hose model is called as PSO-H.

CHAPTER 4

PSO for Hose Model

4.1. Definitions

The network is described as a directed graph $G(V, E)$, where V is the set of nodes and E is the set of links. $v \in V$, where $v = 1, 2, \ldots, N$, indicates an individual node and e, where $e = 1, 2, \ldots, L$, indicates a bidirectional bundled link. c_e is the capacity of bundled link $e \in E$. The traffic volume on link e is denoted as u_e. n_{ij} is the number of routers used for routing data from node i to node j. As described in [9], the probability of concurrent multiple link failures is much less than that of single link failures. Therefore, we consider only single link failure in the network. F is the set of link failure indices l, where $l = 0, 1, 2, \ldots, L$ and $F = E \cup \{0\}$. The number of elements in F is $|F| = L + 1$. $l = 0$ indicates no link failure and $l(\neq 0)$ indicates the failure of bundled link $e = l(\neq 0)$ because of link failure. G_l denotes G that has no bundled link $e = l(\neq 0)$ because of link failure. $G_0 = G$ as $l = 0$ indicates no link failure. $W = \{w_e\}$ is the link weight matrix of network G, where w_e is the weight of bundled link e. Let $\{1, \ldots, w_{max}\}$ be the set of possible link weights. x_{ij}^e is the portion of traffic from node $i \in V$ to $j \in V$ routed through bundled link e.

Note that routing and x_{ij}^e are determined if link weights are known since an OSPF-based backbone uses the shortest path routing. $x_{ij}^e(W)$ is used to represent the load distribution variables under link weight set W. Let a_i and b_i represent the maximum amount of ingress and egress traffic allowed to enter and leave the network at node i, respectively. Given the ingress and egress traffic constraints specified by $H = [(a_1, b_1), \ldots, (a_n, b_n)]$, there are many traffic matrices that satisfy the constraints imposed by H. A traffic matrix $T = \{d_{ij}\}$, where d_{ij} represents the traffic rate from node i to node j, is called a *valid* traffic matrix if it does not violate the constraints imposed by H. Let $\tilde{D}$ be the set of all valid Ts.

The network congestion ratio r refers to the maximum value of all link utilization ratios in the network. r is defined by,

$$r = \max_{e \in E} \frac{u_e}{c_e}, \tag{4.1}$$

where $0 \leq r \leq 1$. Under the condition that routing is not changed, traffic volume $\frac{1-r}{r} d_{ij}$ is the highest traffic volume that can be added to the existing traffic volume of d_{ij} for any pair of source node i and destination node j such that the traffic volume passing through any link e does not exceed c_e. After adding $\frac{1-r}{r} d_{ij}$ to d_{ij}, the total traffic volume becomes $\frac{1}{r} d_{ij}$. Therefore, the updated network congestion ratio becomes 1, which is the upper limit. Maximizing the additional traffic volume of $\frac{1-r}{r} d_{ij}$ is equivalent to minimizing r [5].

In this work, each edge router can only accept a limited amount of traffic in both the ingress and the egress directions. They are called the ingress and egress traffic constraints of an edge router. We call a network $non - blocking$ if none of its internal links will ever experience congestion as long as the ingress and egress routers of the network have capacity to admit the flow. If we use a non-blocking IP-based network, we only need to check the bandwidth availability at the ingress and egress points of the network.

4.2. PSO-H for Resilient Routing

PSO-H for resilient routing is described in this section. The target of this research is to find the most appropriate set of link weights, W_{min}, for network G that minimizes the worst-case network congestion ratio over link failure index $l \in F$ and traffic matrices $T \in \tilde{D}$. W_{min} is defined by,

$$W_{min} = \arg \min_{W \in w} \max_{G_l \in \tilde{G}} \max_{T \in \tilde{D}} r(G_l, T, W). \tag{4.2}$$

The traffic matrix $T \in \tilde{D}$ that maximizes the congestion ratio against all the single link failure scenarios of $G_l \in \tilde{G}$ is searched followed by the finding of the link weight set that minimizes the worst-case congestion ratio.

4.2.1. Procedure. The proposed scheme is divided into three stages.

- Stage 1: Generating traffic matrices.

- Stage 2: Searching for an optimal link weight set that reduces the worst-case congestion ratio against all the possible single link failure scenarios.
- Stage 3: Choosing a link weight set that provides the minimum congestion ratio.

At Stage 1, we generate the traffic matrices that lead to the maximum load appeared on each link $e \in E$ in the allowable traffic bound (a_i, b_i).

At Stage 2, we calculate the congestion ratios for all the traffic matrices against single link failure and find which link failure topology gives the maximum congestion ratio. As shown in Figure 4.1, for the first traffic matrix T^1, G_1 maximizes the congestion ratio; for the second traffic matrix T^2, G_0 maximizes the congestion ratio, etc. Within all of the traffic matrices against single link failure topologies the traffic matrix that maximizes the congestion ratio is chosen. In the example shown in Figure 4.1, the second traffic matrix T^2 is chosen. Then we try to reduce that congestion ratio by changing the link weight of the most congested link (w_3 in the example). The same topology after one link weight change (link weight set W_2) is shown in Figure 4.2.

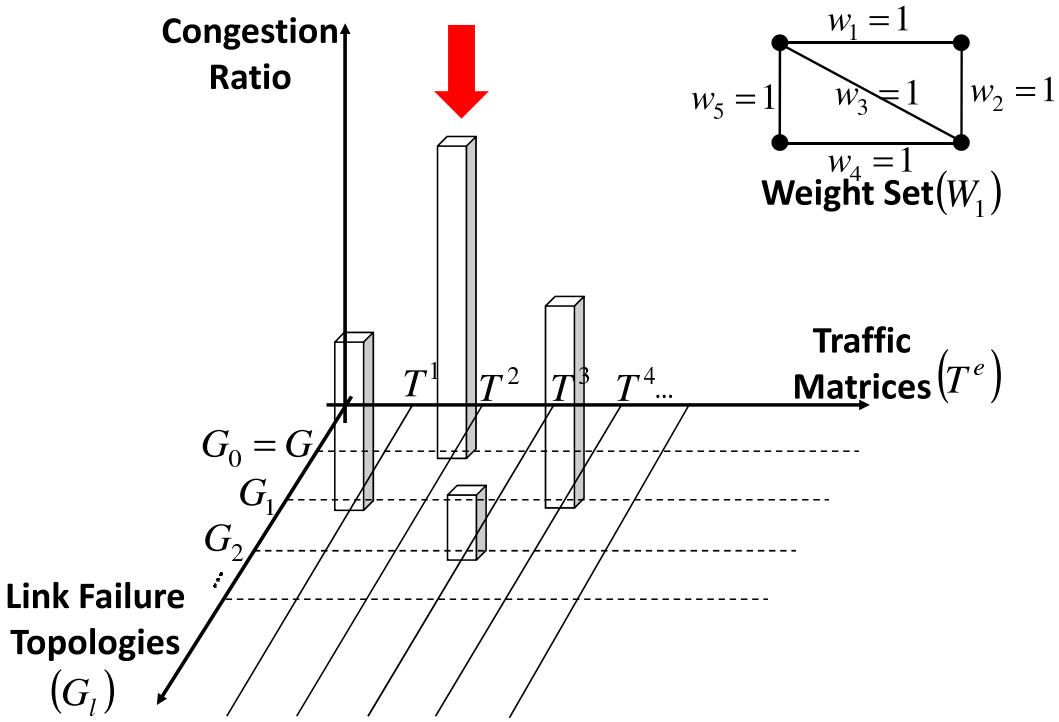

FIGURE 4.1. An example of the proposed scheme

At Stage 3, the improvement of the new link weight set is evaluated. If the link weight set is accepted, the algorithm terminates. If not, it returns to Stage 1.

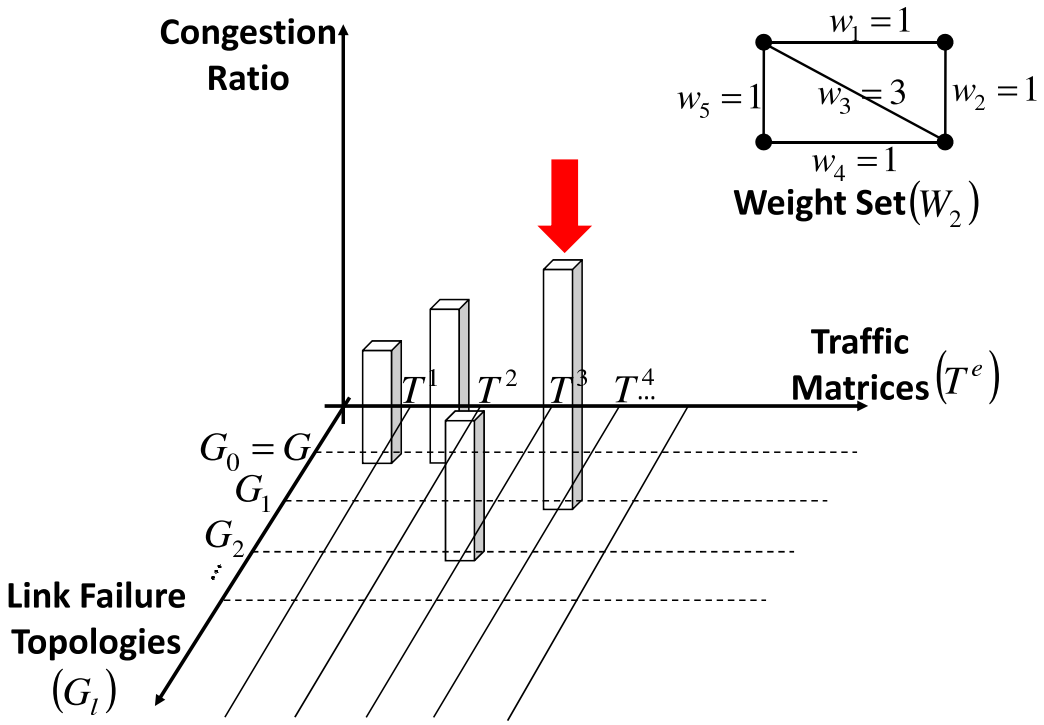

FIGURE 4.2. An example of the proposed scheme after one link weight change

4.2.2. PSO-H for Resilient Routing Algorithm. The description of PSO-H, which is a TS-based link weight optimization scheme, is as follows.

Stage 1: Generating traffic matrices

- Step 1: *Set initial link weights*

  At first, the link weights are generated randomly. Once link weights are known, the shortest paths and routing $x_{ij}^e(W)$ are determined.

- Step 2: *Generate traffic matrices*

  For each link e, the following linear programming formulation is used to find the worst-case traffic matrix T^e that leads to the maximum load appeared on link e.

$$\max \sum_{ij} x_{ij}^e(W) d_{ij} \tag{4.3a}$$

$$s.t \sum_{j \in V} d_{ij} \leq a_i \quad i \in V \tag{4.3b}$$

$$\sum_{I \in V} d_{ij} \leq b_j \quad j \in V \tag{4.3c}$$

$$d_{ij} \geq 0 \quad i, j \in V \tag{4.3d}$$

The traffic matrix T^e that achieves the maximum link utilization for each link e will be added to the set $\tilde{D}$ if it is not in $\tilde{D}$ already.

The updated set $\tilde{D}$ produced at Stage 1 is used to search for new link weights that reduce an objective function. The objective function considerably affects the efficiency of the algorithm. Let $\tilde{r}$ denote the congestion ratio for set $\tilde{D}$. Let r_T be the maximum link utilization ratio for a specific traffic matrix T. Therefore, $\tilde{r} = \max_{T \in \tilde{D}}\{r_T\}$. Although our goal is to minimize $\tilde{r}$, we find that $\tilde{r}$ is not a suitable objective function in the optimization process because changing a link weight reduces one r_T but also often increases a different $r_{T'}$. This means that the improvement of $\tilde{r}$ cannot be done in any iteration.

A better objective function, as used in [21], should include r_T for all traffic matrices in $\tilde{D}$. The sum of individual cost function $\phi(r_T)$ of r_T is chosen as the objective function $F(\tilde{D})$ of the proposed scheme. $F(\tilde{D})$ is defined as,

$$F(\tilde{D}, G_l)_{|\text{for given weights}} = \max_{G_l \in \tilde{G}} \sum_{T \in \tilde{D}} \phi(r_T), \tag{4.4}$$

where $\phi(r_T)$ increases with r_T. Inspired by [21], we adopted the following convex piecewise linear cost function for $\phi(r_T)$.

$$\phi(r_T) = \begin{cases} r_T, & 0 \le r_T < \frac{1}{3} \\ 3r_T - \frac{2}{3}, & \frac{1}{3} \le r_T < \frac{2}{3} \\ 10r_T - \frac{16}{3}, & \frac{2}{3} \le r_T < \frac{9}{10} \\ 70r_T - \frac{178}{3}, & \frac{9}{10} \le r_T < 1 \\ 500r_T - \frac{1468}{3}, & 1 \le r_T < \frac{11}{10} \\ 5000r_T - \frac{16318}{3}, & \frac{11}{10} \le r_T < \infty. \end{cases} \tag{4.5}$$

In [21], it is stated that they have tried different convex objective functions and they all have similar performances in terms of network congestion ratio minimization. Thus, the proposed scheme also uses the same convex objective function.

Stage 2: Searching for an optimal link weight set

- Step 1: *Initialize*

  Variable F_{min}, which is used to store the value of the objective function, is set

to infinite. The repetition counter I_c, used to stop the oscillation of the objective function, is also set to zero.

- Step 2: *Choose a traffic matrix*

  At first the repetition counter I_c is checked. If it is greater than the allowed repetition number, go to Step 1 of Stage 3. If not, the traffic matrix T_{max} that maximizes the cost function defined in Eq. 4.5 against all the single link failure instances is selected.

- Step 3: *Find the most congested link*

  By using the traffic matrix T_{max}, which was selected in Step 2 of Stage 2, the most congested link, e_{cong}, in the network against single link failures is selected.

- Step 4: *Update the link weight*

  The link weight of the most congested link, selected in the previous step is increased by the minimum value that changes at least one route passing through the link for all single link failure scenarios. Therefore, the congestion of the most congested link is decreased. The updated link weight set is inserted into the tabu list. If the updated link weight exceeds the upper limit of the feasible link weight, go to Step 1 of Stage 3.

- Step 5: *Evaluate the objective function*

  For the updated traffic distribution obtained in Step 4 of Stage 2, the objective function of Eq. 4.4 is calculated and compared with that of the old weight set. If the value of Eq. 4.4 for the new weight set is greater than that of the old weight set, repetition counter I_c is reset to zero and the new weight set is set as W_{min} and go to Step 2 of Stage 2. Otherwise, repetition counter I_c is increased by one and go to Step 2 of Stage 2.

Stage 3: Choosing an optimal link weight set

- Step 1: The congestion ratio r for W_{min} is calculated and, if r differs from $\tilde{r}$ by a predefined ϵ, the algorithm terminates. If not, go to Step 2 of Stage 1 and start from the calculation of traffic matrices. W_{min} is an optimal link weight set for the given network against single link failure scenarios.

Since the traffic matrices play an important role in the effectiveness of the proposed method, we randomly use a significantly large number of independent initial link weight

sets. The link weight set that gives the minimum congestion ratio against single link failure is selected as an optimal link weight set.

4.3. PSO-H for Energy Saving Routing with Resilient Routing

This section describes the PSO-H policy for energy saving routing with resilient routing. The PSO-H for resilient routing policy is aimed at reducing the worst-case network congestion ratio; which is the most utilized link in the network. Energy efficiency in networks can be achieved by evenly distributing the traffic among the links. The energy consumed by a network can be expressed by the resources it uses. Lesser the network resources are used less the energy consumed by the network. This is similar to a computer, where a computer consumes a lot of energy if it is doing heavy calculations. Routers in a network also act like a computer, routing the data packets according to destination address. If the traffic directed by a router is lesser the energy consumed by the router is decreased. Not only link utilization but also the number of routers used to route data from source to destination effects the energy consumption in the network. In this section, we describe how to find the most appropriate set of link weights, W_{min}, for network G that minimizes the worst-case total resource usage over link failure index $l \in F$ and traffic matrices $T \in \tilde{D}$. W_{min} is defined by,

$$W_{min} = \arg \min_{W \in w} \max_{G_l \in \tilde{G}} \max_{T \in \tilde{D}} R(G_l, T, W), \tag{4.6}$$

where $R(G, T, W)$ is the total resources used for given G, T, l. The procedure of PSO-H for Energy saving is same as PSO-H for resilient routing. Next, the algorithm of PSO-H for energy saving routing with resilient routing is explained.

4.3.1. PSO for Energy Saving Routing with Resilient Routing Algorithm.

The description of the proposed scheme is as follows.

Stage 1: Generating traffic matrices

- Step 1: *Set initial link weights*

  At first, the link weights are generated randomly. Once link weights are known, the shortest paths and routing $x_{ij}^{e}(W)$ are determined.

- Step 2: *Generate traffic matrices*

  For each link e, the linear programming formulation stated in Eq. 4.3 is used to find the worst-case traffic matrix T^e that leads to the maximum load appeared on each link e.

The main objective of [18] is to minimize the worst-case congestion ratio. Therefore, the objective function used in [18] is sum of the congestion ratios for all the traffic matrices. But, when considering the total network energy consumption, sum of link utilization ratios and the number of routers used for routing is a better objective function.

As the hose model accommodates multiple traffic matrices, the objective function, as used in [18], should include u_e for all traffic matrices in $\tilde{D}$. The sum of individual cost function $\phi(u_e)$ of u_e is chosen as the objective function $F(\tilde{D})$ of the proposed scheme. $F(\tilde{D})$ is defined as,

$$F(\tilde{D}, G_l)_{|\text{for given weights}} = \max_{G_l \in \tilde{G}} \sum_{T \in \tilde{D}} \{ \sum_{e \in E} \phi(u_e) \times P1 + \sum_{ij} n_{ij} \times P2 \}, \tag{4.7}$$

where $\phi(u_e)$ increases with u_e. As mentioned in [21], we adopted the following convex piecewise linear cost function for $\phi(u_e)$.

$$\phi(u_e) = \begin{cases} \frac{u_e}{c_e}, & 0 \leq \frac{u_e}{c_e} < \frac{1}{3} \\ 3\frac{u_e}{c_e} - \frac{2}{3}, & \frac{1}{3} \leq \frac{u_e}{c_e} < \frac{2}{3} \\ 10\frac{u_e}{c_e} - \frac{16}{3}, & \frac{2}{3} \leq \frac{u_e}{c_e} < \frac{9}{10} \\ 70\frac{u_e}{c_e} - \frac{178}{3}, & \frac{9}{10} \leq \frac{u_e}{c_e} < 1 \\ 500\frac{u_e}{c_e} - \frac{1468}{3}, & 1 \leq \frac{u_e}{c_e} < \frac{11}{10} \\ 5000\frac{u_e}{c_e} - \frac{16318}{3}, & \frac{11}{10} \leq \frac{u_e}{c_e} < \infty. \end{cases} \tag{4.8}$$

In [21], it is stated that they have tried different convex cost functions and they all have similar performances in terms of network congestion ratio minimization. Thus, the proposed scheme also uses the same convex cost function.

Stage 2: Searching for an optimal link weight set

- Step 2: *Initialize*

  Variable F_{min}, which is used to store the value of the objective function, is set

to infinite. The repetition counter I_c, used to stop the oscillation of the objective function, is also set to zero.

- Step 2: *Choose a traffic matrix*

  At first the repetition counter I_c is checked. If it is greater than the allowed repetition number, go to Step 1 of Stage 3. If not, the traffic matrix T_{max} that maximizes the cost function defined in Eq. 4.8 against all the single link failure instances is selected.

- Step 3: *Update a link weight randomly*

  A link is selected randomly and its weight is changed (increase or decrease randomly) by the minimum value that changes at least one route passing through the link for all single link failure scenarios. This local search is based on simulated annealing [13]. If the updated link weight exceeds the upper limit of the feasible link weight, go to Step 1 of Stage 3.

- Step 4: *Evaluate the objective function*

  For the updated traffic distribution obtained in Step 3 of Stage 2, the objective function of Eq. 4.7 is calculated and compared with that of the old weight set. If the value of Eq. 4.7 for the new weight set is greater than that of the old weight set, repetition counter I_c is reset to zero and the new weight set is set as W_{min} and go to Step 2 of Stage 2. Otherwise, repetition counter I_c is increased by one and go to Step 2 of Stage 2.

Stage 3: Choosing an optimal link weight set

- Step 1: The total resource usages R for W_{min} and R' for W are calculated and, if the difference of R and R' is smaller than a predefined ϵ, the algorithm terminates. If not, go to Step 2 of Stage 1 and start from the calculation of traffic matrices. W_{min} is an optimal link weight set for the given network against single link failure scenarios.

CHAPTER 5

Performance Evaluation

5.1. Simulation Model

The simulation models that we used are described as follows. To determine the basic characteristics of this, six sample networks are used as shown in Figure 5.1. Networks 1-3 mirror the typical backbone networks used to evaluate the routing performance in both [15] and [21]. Network 4 is the Abilene network [22] and Network 5 is the National Science Foundation network [23]. Network 6 is a random network generated using the BRITE topology generator [24] and the *Waxman's Probability model* was used to create it. Table 5.1 summaries the basic characteristics of the sample networks used. The link capacities of the sample networks were randomly generated with uniform distribution in the range of ($10U_c$, $100U_c$), where U_c[Gbit/s] is given a constant integer value. The maximum link weight, w_{max}, is set at 100. We confirmed that $w_{max} > 100$ provides the same results as $w_{max} = 100$ in our examined networks. C_{max} is set to 1000, as it needs to exceed the maximum number of links in our examined networks [15]. The simulation program is coded in C language on a Linux computer with 4GB of RAM and an AMD Phenom II X6 processor. The linear programming problem in Eq. 4.3 is solved using the IBM ILOG CPLEX Optimization Studio 12.4.

TABLE 5.1. Characteristics of the sample networks

Network	No. of nodes	Average node degree	No. of links (bidirectional)
1	6	3.67	11
2	12	4.00	24
3	15	3.73	28
4	11	2.54	14
5	14	3.00	21
6	20	3.70	37

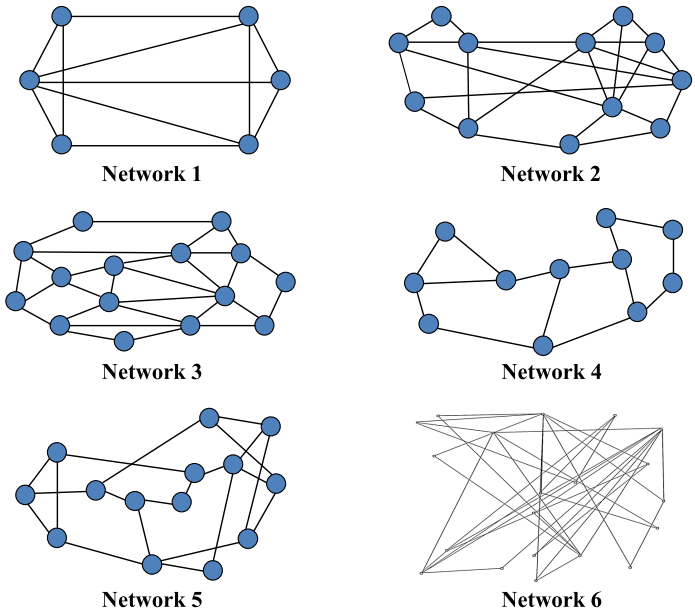

FIGURE 5.1. Sample networks

5.2. Evaluation of PSO-H for Resilient Routing

This section compares the performance of PSO-H with that of SO and RO via simulations. Network congestion ratio r is the performance measure of the evaluation.

Let $r(l)$ be denoted as the network congestion ratio for link failure index $l \in F$. To normalize the calculated network congestion ratios of the sample networks, the congestion ratio of SO without any link failure is used. The normalized network congestion ratio of SO is denoted as $r_{SO}(l)$, the normalized congestion ratio of RO is denoted as $r_{RO}(l)$, and the normalized congestion ratio of PSO-H is denoted as $r_{PSO-H}(l)$.

The worst-case network congestion ratios, $\max_{l \in F} r_{SO}(l)$, $\max_{l \in F} r_{RO}(l)$, and $\max_{l \in F} r_{PSO-H}(l)$ for the sample networks presented in Figure 5.1 for all single link failure scenarios are calculated as shown in Table 5.2. For the worst-case network congestion ratio for single link failure, the following relationship is observed,

$$\max_{l \in F} r_{RO}(l) \leq \max_{l \in F} r_{PSO-H}(l) \leq \max_{l \in F} r_{SO}(l). \tag{5.1}$$

This indicates that the proposed scheme, PSO-H is able to reduce the worst-case network congestion ratio as compared with SO. It also avoids the run-time link weight changes, which would cause network instability. As expected RO gives the optimal performance when a link failure occurs even though RO may lead to network instability. The achieved

reduction rate of the worst-case congestion ratio, α, is defined as,

$$\alpha = \frac{\max\limits_{l \in F} r_{SO}(l) - \max\limits_{l \in F} r_{PSO-H}(l)}{\max\limits_{l \in F} r_{SO}(l)}. \tag{5.2}$$

α is also shown in Table 5.2.

TABLE 5.2. Comparison of worst-case network congestion ratios for single link failure scenarios

Network	$\max\limits_{l \in F} r_{SO}(l)$	$\max\limits_{l \in F} r_{RO}(l)$	$\max\limits_{l \in F} r_{PSO-H}(l)$	α
1	1.80	1.10	1.10	0.39
2	1.70	1.30	1.60	0.06
3	2.50	1.40	1.50	0.40
4	1.90	1.10	1.20	0.37
5	1.45	1.00	1.00	0.31
6	3.00	1.90	2.00	0.33

The normalized congestion ratios for no link failure scenario are shown in Table 5.3. For the case of no link failure,

$$r_{RO}(0) = r_{SO}(0) \leq r_{PSO-H}(0) \tag{5.3}$$

is observed. When there is no link failure, the congestion ratio of the link weight set obtained using PSO-H may be higher than that of SO or RO. This is because the objective of PSO-H is to reduce the worst-case network congestion ratio when link failure occurs. β is the deviation between $r_{PSO-H}(0)$ and $r_{SO}(0)$. β is defined as,

$$\beta = \frac{r_{PSO-H}(0) - r_{SO}(0)}{r_{SO}(0)}. \tag{5.4}$$

β is also shown in Table 5.3. When there is no link failure, β is the "penalty" PSO-H has to "pay" to reduce the worst-case network congestion.

TABLE 5.3. Comparison of network congestion ratios with no link failure

Network	$r_{SO}(0)(= r_{RO}(0))$	$r_{PSO-H}(0)$	β
1	1.00	1.00	0.00
2	1.00	1.20	0.20
3	1.00	1.13	0.13
4	1.00	1.48	0.48
5	1.00	1.03	0.03
6	1.00	1.50	0.50

In order to understand the relationship between the worst-case congestion ratio and network topology, several random network topologies are used. These random networks are generated using the BRITE [24] Internet topology generator by changing the number of nodes N and the number of adjacency nodes m of the network. The Waxman's probability model is used for interconnecting the nodes of the topology, which is given by,

$$P(u, v) = A \ \exp(-\frac{d}{BL}), \tag{5.5}$$

where, $0 < A,\ B \leq 1$, d is the Euclidean distance from node u to node v, and L is the maximum distance between any two nodes. A and B are set to 0.15 and 0.2, respectively. The number of nodes N is set to $8, 10, 15$ and the number of adjacency nodes m is set to $3, 4, 5, 6$. The characteristics of the generated random network topologies are shown in Table 5.4.

TABLE 5.4. Characteristics of the random sample networks

N	m	Average node degree	No. of links (bidirectional)
8	3	4.50	18
8	4	4.75	19
8	5	3.50	14
8	6	3.00	12
10	3	5.60	28
10	4	6.00	30
10	5	6.40	32
10	6	4.80	24
15	3	6.00	45
15	4	7.20	54
15	5	8.53	64
15	6	8.93	67

The dependency of α on N and m is shown in Figure 5.2. This result shows that α is increasing with N when m is higher ($m = 5, 6$). It means the difference between the worst-case congestion ratios of SO and PSO-H is increasing. This may relate to the fact that routing flexibility is increased as N and m become higher.

Figure 5.3 shows the dependency of β against N and m. Figure 5.3 indicates that PSO-H is able to achieve the same result as SO for no link failure when the network becomes larger. This may occur because there is a wider number of routes to chose from when the network is larger.

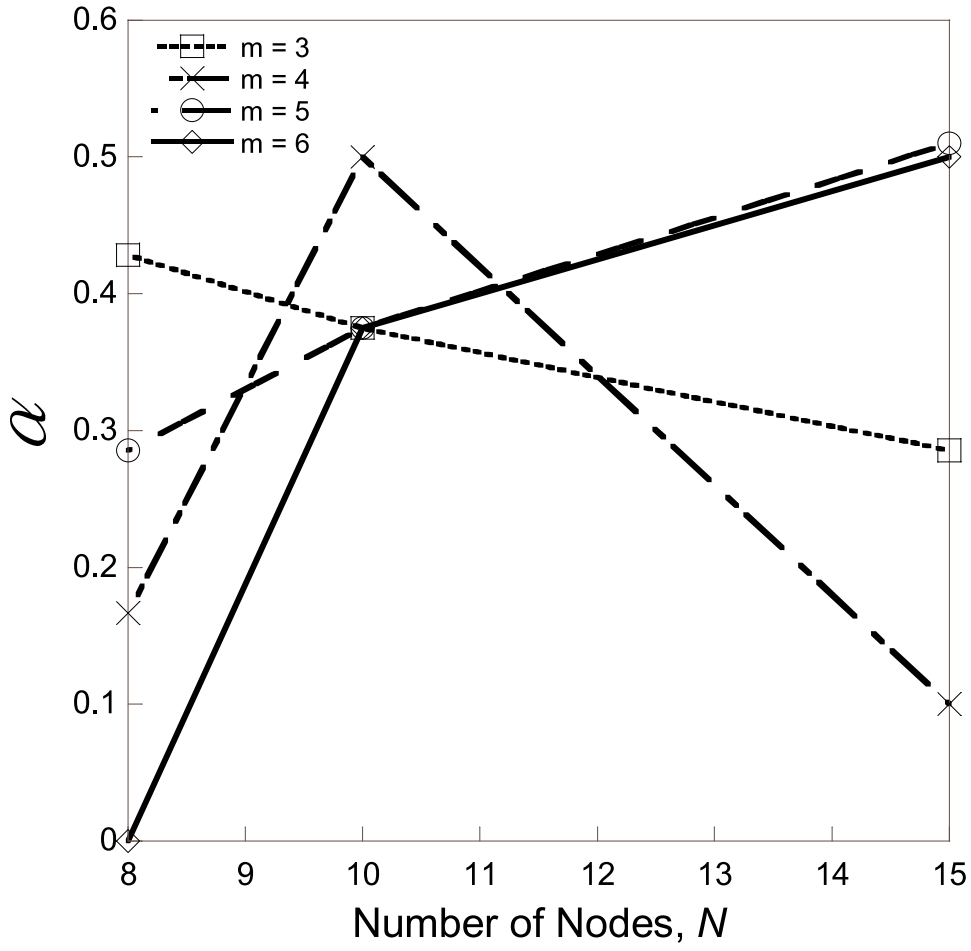

FIGURE 5.2. α's dependency on number of nodes and adjacency nodes

In order to show the effectiveness of the proposed scheme, the worst-case congestion ratios of the proposed scheme are compared with those of the two following link weight setting schemes. One is a scheme in which a link weight is inversely proportional to its capacity [25]. We call it the IPC scheme. The other scheme is the one in which all the link weights are set to one. As a result, minimum-hop routing is achieved. We call it the min-hop scheme. Table 5.5 shows the worst-case congestion ratios of the three schemes, which are normalized by that of the proposed scheme. Table 5.5 indicates that the proposed scheme reduces the worst-case congestion ratio, compared to the IPC scheme and the min-hop scheme. This is because the proposed scheme determines link weights considering any single link failure so as to minimize the worst-case congestion ratio.

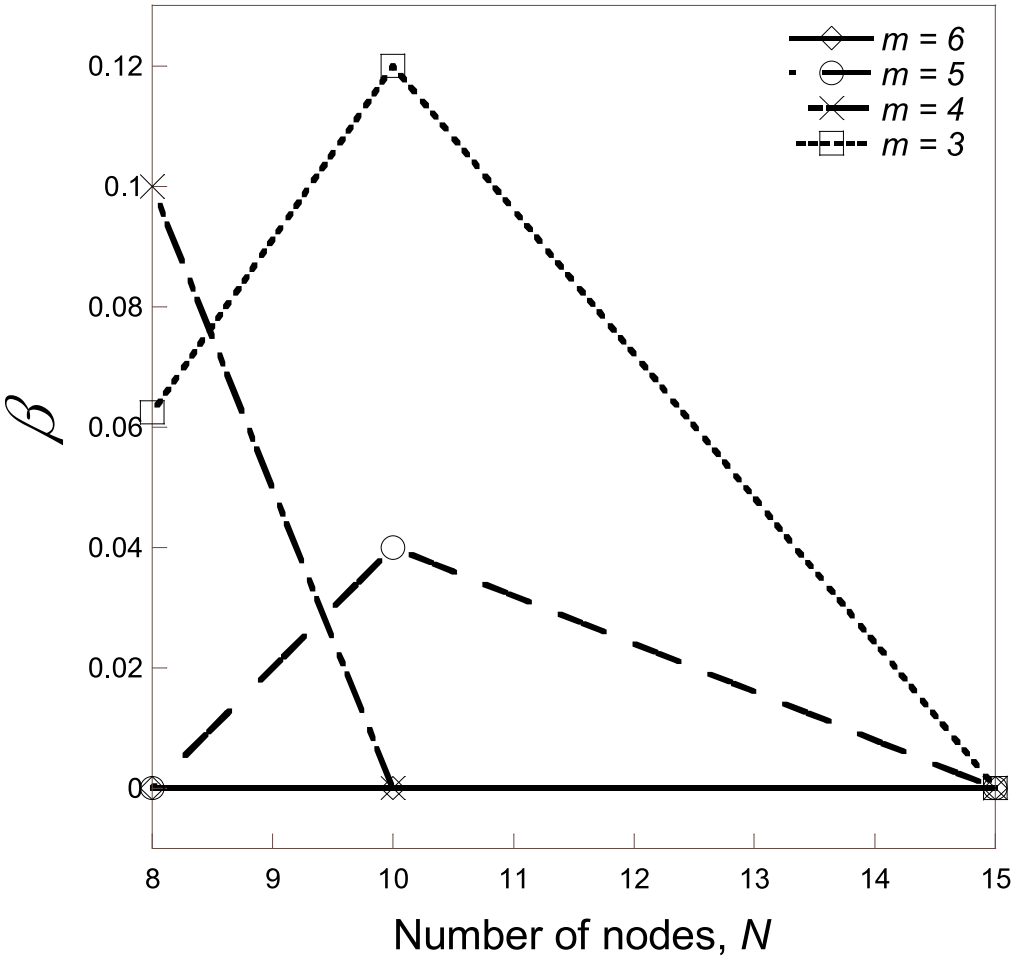

FIGURE 5.3. β's dependency on number of nodes and adjacency nodes

TABLE 5.5. Comparison of worst-case congestion ratios with different link weight setting schemes

Network	Proposed scheme	IPC scheme	min-hop scheme
1	1.00	1.32	2.00
2	1.00	1.91	1.62
3	1.00	2.43	3.50
4	1.00	3.23	1.12
5	1.00	2.03	2.40
6	1.00	2.61	2.75

The numerical results via simulation show that PSO-H for resilient routing finds a suitable link weight set to reduce the worst-case network congestion in most cases. It has to pay a penalty of β for no link failure scenario, however if the network administrators want

to reduce the worst-case network congestion ratio for single link failure, PSO-H is a better choice. It is observed that PSO-H outperforms SO for larger networks.

5.3. Evaluation of PSO-H for Energy Saving Routing with Resilient Routing

This study compares the performance of the proposed scheme with that of SO and minimum-hop (min-hop) routing via simulations. The min-hop routing is achieved by setting all the link weights to one. The total resource used, R for routing the data is the performance measure of the evaluation.

Let $R(l)$ be denoted as the total resource usage for link failure index $l \in F$. For the no link failure scenario the simulation results are normalized with the total resource usage of SO and for the worst-case single-link failure scenario the simulation results are normalized with the total resource usage of PSO. The normalized total resource usage of SO is denoted as $R_{SO}(l)$, the resource usage of min-hop is denoted as $R_{min-hop}(l)$, and the total resource usage of PSO is denoted as $R_{PSO}(l)$.

The worst-case total resource usages, $\max_{l \in F} R_{PSO}(l)$, $\max_{l \in F} R_{SO}(l)$, and $\max_{l \in F} R_{min-hop}(l)$ for the sample networks presented in Figure 5.1 for all single link failure scenarios when $P1 = 1$ $P2 = 1$ are calculated as shown in Table 5.6. The data in Table 5.6 are normalized with $\max_{l \in F} R_{PSO}(l)$. For the worst-case network resource usage for single link failure, the following relationship is observed,

$$\max_{l \in F} R_{PSO}(l) < \max_{l \in F} R_{SO}(l). \tag{5.6}$$

This indicates that the proposed scheme is able to reduce the worst-case network resources used as compared with SO and min-hop. The achieved reduction rate of the worst-case resource usage, α, is defined as,

$$\alpha = \frac{\max_{l \in F} R_{SO}(l) - \max_{l \in F} R_{PSO}(l)}{\max_{l \in F} R_{SO}(l)}. \tag{5.7}$$

α is also shown in Table 5.6.

The total resource usages for no link failure scenario, normalized by $R_{SO}(0)$, when $P1 = 1$ $P2 = 1$ is shown in Table 5.7. For the case of no link failure,

$$R_{SO}(0) \leq R_{PSO}(0) \leq R_{min-hop}(0) \tag{5.8}$$

TABLE 5.6. Comparison of worst-case network resource usage for single link failure scenarios (P1=1, P2=1)

Network	$\max_{l \in F} R_{PSO}(l)$	$\max_{l \in F} R_{SO}(l)$	$\max_{l \in F} R_{min-hop}(l)$	α
1	1.00	1.55	1.35	0.36
2	1.00	1.24	1.23	0.19
3	1.00	1.11	1.11	0.09
4	1.00	1.24	2.00	0.19
5	1.00	1.45	2.00	0.31
6	1.00	1.60	1.36	0.38

is observed. When there is no link failure, the total resource usage of the link weight set obtained using PSO may be higher than that of SO. This is because the objective of PSO is to reduce the worst-case network resource usage when link failure occurs. β is the deviation between $r_{PSO}(0)$ and $r_{SO}(0)$. β is defined as,

$$\beta = \frac{R_{PSO}(0) - R_{SO}(0)}{r_{SO}(0)}. \tag{5.9}$$

β is also shown in Table 5.7. When there is no link failure, β is the "penalty" PSO has to "pay" to reduce the worst-case network resource usage.

TABLE 5.7. Comparison of network resource usage for no link failure scenario (P1=1, P2=1)

Network	$R_{SO}(0)$	$R_{PSO}(0)$	$R_{min-hop}(0)$	β
1	1.00	1.15	1.65	0.15
2	1.00	1.17	1.60	0.17
3	1.00	1.01	2.22	0.01
4	1.00	1.11	1.11	0.11
5	1.00	1.00	2.21	0.00
6	1.00	1.05	1.75	0.05

The total resource usage when $P1 = 1$ $P2 = 0$ for worst-case scenario and no link failure scenario are shown in Table 5.8, 5.9 respectively. The above observations in Eq. 5.6, 5.8 are also true when $P1 = 1$ $P2 = 0$.

The total resource usage when $P1 = 0$ $P2 = 1$ for worst-case scenario and no link failure scenario are shown in Table 5.10, 5.11 respectively. The above observations in Eq. 5.6, 5.8 are also true when $P1 = 0$ $P2 = 1$.

It is clear that when $P1 = 0$, $P2 = 1$ the total resource usage of PSO, SO, and min-hop are almost the same. This means that when only the number of routers in the path is

TABLE 5.8. Comparison of worst-case network resource usage for single link failure scenarios (P1=1, P2=0)

Network	$\max_{l \in F} R_{PSO}(l)$	$\max_{l \in F} R_{SO}(l)$	$\max_{l \in F} R_{min-hop}(l)$	α
1	1.00	1.13	1.18	0.12
2	1.00	1.09	1.12	0.08
3	1.00	1.23	1.22	0.19
4	1.00	1.18	1.12	0.15
5	1.00	1.22	1.05	0.18
6	1.00	1.19	1.81	0.16

TABLE 5.9. Comparison of network resource usage for no link failure scenario (P1=1, P2=0)

Network	$R_{SO}(0)$	$R_{PSO}(0)$	$R_{min-hop}(0)$	β
1	1.00	1.03	1.08	0.03
2	1.00	1.01	1.03	0.01
3	1.00	1.05	1.03	0.05
4	1.00	1.14	1.26	0.14
5	1.00	1.02	1.03	0.02
6	1.00	1.04	1.06	0.04

TABLE 5.10. Comparison of worst-case network resource usage for single link failure scenarios (P1=0, P2=1)

Network	$\max_{l \in F} R_{PSO}(l)$	$\max_{l \in F} R_{SO}(l)$	$\max_{l \in F} R_{min-hop}(l)$	α
1	1.00	1.011	1.007	0.011
2	1.00	1.003	1.021	0.003
3	1.00	1.016	1.002	0.016
4	1.00	1.007	1.003	0.006
5	1.00	1.019	1.009	0.019
6	1.00	1.004	1.009	0.004

TABLE 5.11. Comparison of network resource usage for no link failure scenario (P1=0, P2=1)

Network	$R_{SO}(0)$	$R_{PSO}(0)$	$R_{min-hop}(0)$	β
1	1.00	1.000	1.000	0.000
2	1.00	1.007	1.019	0.007
3	1.00	1.006	1.008	0.006
4	1.00	1.072	1.039	0.072
5	1.00	1.060	1.041	0.060
6	1.00	1.003	1.017	0.003

considered, the routing performance is almost identical for no link failure scenario and the worst-case scenario.

The numerical results via simulation show that PSO-H for energy saving routing with resilient routing finds a suitable link weight set to reduce the worst-case network resources used in all cases. It has to pay a penalty of β for no link failure scenario, however if the network administrators want to reduce the worst-case network resource usage for single link failure, PSO is a better choice. It is observed that PSO outperforms SO for larger networks.

CHAPTER 6

Conclusion

The objective of this research is to optimize the OSPF link weights to achieve energy aware, resilient routing in backbone networks. This study was done in two parts. First, we consider the resilient routing and proposed a link weight optimization scheme, based on PSO, to minimize the worst-case congestion ratio against single link failures. Then, we extended it to reduce the network energy consumption by evenly splitting the traffic through the network. This study employs the hose model, which can cope with traffic fluctuations because it only considers the ingress and egress traffic at each node.

The proposed scheme employs a heuristic algorithm to determine a suitable set of link weights to reduce the worst-case congestion, total resource usage for any single link failure respectively. It efficiently selects the worst-case performance traffic matrix and reduces the objective function as compared to a brute-force scheme that is computationally expensive when searching the link weight space against all the possible traffic matrices and topologies created by single link failures. The effectiveness of the proposals were demonstrated through comparison with major link weight optimization policies, SO, RO, and min-hop.

This work only considers single link failure since the probability of multiple link failures is much less than that of single link failure [13]. The proposed scheme can be applied easily to multiple link failures including shared risk link group (SRLG) failures with extra computation time.

Publications

Curriculum Vitae

Ravindra Sandaruwan Ranaweera is a master's student at the University of Electro-Communications. He came to Japan in 2006 as a Monbukagakusho (MEXT) scholar. He received his Bachelor of Engineering from the Department of information and communication, the University of Electro-Communications, Tokyo, Japan in 2012. Then he will receive his Master of Engineering majoring in Information and Communications Systems in March 2013. His research focuses on network optimization, network failure, network energy consumption, and routing.

Journal Publications

- R. S. Ranaweera, I. M. Kamrul, and E. Oki, "Preventive start-time optimization of OSPF link weights for hose model," *IET Networks*, available online, Sep. 2013

Conference Proceeding Publications

- R. S. Ranaweera, I. M. Kamrul, and E. Oki, "Preventive start-time optimization of OSPF link weights against link failure for hose model," *18th Asia-Pacific Conference on Communications (APCC 2012)*, Oct. 2012.
- R. S. Ranaweera, I. M. Kamrul, and E. Oki, "Performance evaluation of preventive start-time routing optimization for hose model," *IEICE Technical report (PN2013-4)*, Vol. 113, No.91, Page:19-24, June 2013.
- R. S. Ranaweera, I. A. Ouedraogo, and E. Oki, "Performance evaluation of preventive start-time optimization for energy saving of hose model against link failure," *IEICE Technical report (NS)*, Mar. 2014.

APPENDIX A

Simulation Program Source Code

I_{max} refers to the allowable number of iterations to reduce the total network resource usage. network size, number of link in the network, allowable computation time, and the quality of solution desired are the factors we take into consideration when we decided a value for I_{max}. The larger I_{max} allows us to get a more appropriate solution closer to the optimum solution, while increasing the computation time. Our careful observation on the effect of I_{max} suggests that, after reaching an enough high value, performance of the PSO-H for energy saving algorithm remains the same, where the total network resource usage does not decreased.

Table A.1 shows the effect of I_{max} for network 1, as shown in Figure 5.1 when $P1 = 1$ $P2 = 1$. This observation indicates that the performance of PSO-H for energy saving does not change with the increment of I_{max} after 800. Therefore, in our simulation we set the value of I_{max} to 1000.

TABLE A.1. Comparison of algorithm performance related to I_{max}

Value of I_{max}	Worst-case total network resource usage
100	49.17
200	48.83
300	48.83
400	48.67
500	48.67
600	48.67
700	48.33
800	47.92
900	47.92
1000	47.92
1100	47.92
1200	47.92
1300	47.92
1400	47.92
1500	47.92

Next, we explain the importance of deciding on the value of w_{max}. In OSPF networks, link cost can take an integer value between 1-65535. If 65535 is assigned to a link between nodes a and b it means that there is no link between a and b. It is ideal to make w_{max} 65535, but increases the computation time without improving the result. Our careful observation on w_{max} suggests that, after reaching an enough high value performance of the PSO-H for energy saving algorithm remains the same, where the total network resource usage does not decreased.

Table A.2 shows the effect of w_{max} for network 1, as shown in Figure 5.1 when $P1 = 1$ $P2 = 1$. This observation indicates that the performance of PSO-H for energy saving does not change with the increment of w_{max} after 90. Therefore, in this study we set the value of w_{max} to 100 in order to reduce the computation time without affecting the results.

TABLE A.2. Comparison of algorithm performance related to w_{max}

Value of w_{max}	Worst-case total network resource usage
10	53.17
20	51.17
40	48.83
50	48.33
80	48.00
90	47.92
100	47.92
120	47.92

All of the above observations are also true for other sample networks shown in Figure 5.1.

Bibliography

[1] A. Farrel, "The Internet and Its Protocols," ELSEVIER, 2004.

[2] J. T. Moy, "OSPF Version 2. Network Working Group," 2004.

[3] CISCO, "Configuring OSPF," 1997.

[4] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, Clifford (2001), "Section 24.3: Dijkstra's algorithm", "Introduction to Algorithms (Second Edition)", MIT Press and McGraw-Hill, Page:595-601.

[5] E. Oki, and A. Iwaki, "F-TPR: Fine two-phase IP routing scheme over shortest paths for hose model, IEEE Commun. Letters, Apr. 2009, Vol.13, No.4, Page: 277-279.

[6] J. G. Koomey, "Estimating total power consumption by servers in the U.S. and the world", Staff Scientist, Lawrence Berkeley National Laboratory and Consulting Professor, Stanford University. Feb. 2007.

[7] T. Asami and S. Namiki, "Energy consumption targets for network systems," Sept. 2008, in Proc. 34th ECOC, 2008, pp. 1-4.

[8] K. W. Roth and A. D. Little., "Energy consumption by office and telecommunications equipment in commercial buildings volume I: energy consumption baseline,?h National Technical Information Service (NTIS), U.S. Department of Commerce, Springfield, VA 22161, NTIS Number: PB2002-101438.

[9] C. Lange, "Energy-related aspects in backbone networks,?h in Proc. ECOC 2009.

[10] B. Fortz and M. Thorup, "Optimizing OSPF/IS-IS Weights in a Changing World," May 2002, IEEE Journal on Selected Areas in Communications, Vol.20, No.4, Page:756-767.

[11] F. Glover, "Tabu Search - Part 1 and Part 2," 1989, ORSA Journal on Computing 1 and 2.

[12] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing:an experimental evaluation," Oper. Res., Vol.39, No.1, May-Jun. 1991.

[13] G. Iannaccone, C. Chuah, R. Mortier, S. Bhattacharyya and C. Diot, "Analysis of Link Failures in a Large IP Backbone," Nov. 2002, Proc. 2nd ACM SIGCOM Internet Measurement Workshop.

[14] Fortz, B., and Thorup, M.: "Optimizing OSPF/IS-IS weights in a changing world", IEEE Journal on Selected Areas in Communications, May 2002, 20, (4), pp: 756-767.

[15] I. M. Kamrul and E. Oki, "PSO: Preventive Start-time Optimization of OSPF link weights to counter network failure," IEEE Commun. Letters, June 2010, 14, (6), pp: 581-583.

[16] I. M. Kamrul and E. Oki, "Optimization of OSPF link weights to counter network failure," IEICE Trans. on Commun., July 2011, E94B, (7), pp: 1964-1972.

[17] R. S. Ranaweera, I. M. Kamrul, and E. Oki, "Preventive start-time optimization of OSPF link weights against link failure for hose model," 18th Asia-Pacific Conference on Communications (APCC 2012), Oct. 2012.

[18] R. S. Ranaweera, I. M. Kamrul, and E. Oki, "Preventive start-time optimization of OSPF link weights for hose model," IET Networks, available online, Sep. 2013

[19] S. Iyer, S. Bhattacharyya, N. Taft, N. McKeoen and C. Diot, "A measurement based study of load balancing in an IP backbone," May 2002, Sprint ATL technical report, TR02-ATL-051027.

[20] D. Bertsekas and R. Gallager, "Data networks", 1992, Prentice-Hall.

[21] Jian Chu and Chin-Tau Lea, "Optimal Link Weights for IP-Based Networks Supporting Hose-Model VPNs," June 2009, IEEE/ACM TRANSACTIONS ON NETWORKING Vol.17, No.3, Page:778-788.

[22] "The Internet2 Network," online, http://www.internet2.edu/network/, Nov. 2013.

[23] Ramaswami, R., and Sivarajan, K.N.: "Design of logical topologies for wavelength-routed optical networks," IEEE J. Selected Areas in Communications, June 1996, Vol.14, No.5, pp: 840-851.

[24] http://www.cs.bu.edu/brite/, Dec. 2014.

[25] Cisco: "Configuring OSPF," online, http://www.cisco.com, Jan. 2014.