Adriana de Andrade Oliveira

Um Arcabouço para Engenharia de Tráfego em Redes MPLS

Tese apresentada ao Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Universidade Federal de Minas Gerais
Belo Horizonte, Outubro de 2005

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais e irmãos que sempre estiveram ao meu lado e se fizeram presentes durante o desenvolvimento deste trabalho. Obrigada pela confiança e e pelo apoio incondicional.

Ao professor e orientador Geraldo Robson Mateus pelos valiosos ensinamentos e dedicação.

Aos membros da banca, professores José Augusto Suruagy Monteiro, Celso Ribeiro, Antônio Alfredo Loureiro e José Marcos Silva Nogueira, pelas críticas e sugestões feitas para a melhoria desse trabalho.

Aos demais professores do DCC que contribuíram para minha formação intelectual e pessoal. Em especial, gostaria de agradecer ao professor Virgílio pelos conselhos e amizade.

As funcionárias da secretaria, Renata e Sheila, pela extrema eficiência com que me ajudaram a resolver todos os problemas burocráticos envolvidos com a tese, mesmo à distância.

Aos amigos Rainer e Rodrigo Carceroni pela confiança dispensada e pelas defesas das minhas questões nas reuniões de colegiado.

Aos amigos e colegas do LaPO, especialmente Gurvan, pelo seu trabalho de administração tão eficiente, Fred e Djavan pela *Fazendeira* e pelos momentos de descontração no ICB, Kátia, pelo apoio no desenvolvimento do *meu* genético e Habib pelo apoio na utilização do NS.

Aos amigos Alla, Arthur, Barra, Paulo, Tassni, Flávia, Vitorino, Fábio, Guilherme, Kíssia, Geraldo pelo apoio, pela terapia de graça, pela amizade, pelas idas ao cinema, pelos almoços e pelas risadas no corredor. Às amigas Lena, Silvana, Letícia, Fernanda pela companhia nas festas e finais de semana. Aos demais amigos do DCC, em especial ao pessoal da *dvd-session*, pelo apoio e pela convivência tão alegre.

Ao meu marido, Thomas Stahl, pelo amor e incentivo que eu precisava para terminar esse trabalho.

Resumo

Introdução

O crescimento do uso da Internet, juntamente com a diversificação de suas aplicações, trouxe novos desafios à área de projeto e planejamento de redes. Um desses desafios é utilizar eficientemente a infra-estrutura existente, oferecendo serviços com diferenciados níveis de qualidade. Em redes orientadas a conexão, a transferência de informação entre usuários finais, com determinados níveis de qualidade, é conseguida pelas próprias funções de rede que selecionam e alocam recursos da rede. Já em redes IP (Internet Protocol), esta é uma necessidade nova, já que não existe o conceito de reserva de recursos.

Em redes IP, freqüentemente se percebe a concentração do fluxo em algumas partes da rede enquanto outras partes permanecem sub-utilizadas. A má utilização de recursos existentes é devida à má distribuição do tráfego feita pelos algoritmos de roteamento, que buscam geralmente o menor caminho da origem até o destino e não levam em consideração o tráfego já existente nesse caminho. Portanto, para tornar realidade o provisionamento de qualidade de serviço (QoS) em redes IP, torna-se necessário o desenvolvimento de ferramentas de Engenharia de Tráfego.

As ferramentas de Engenharia de Tráfego são responsáveis por várias funções de gerenciamento do tráfego. Entre elas podem ser citadas o estabelecimento de políticas para controle de admissão de requisições, controle e determinação de rotas físicas para os fluxos, assim como a reserva de recursos e controle da rede em diferentes escalas de tempo. A combinação de tecnologias de comutação de rótulos e mecanismos de roteamento que levam em consideração métricas de desempenho da rede e, ao mesmo tempo, permitem aos administradores influenciar as decisões de roteamento baseados em suas preferências parece ser uma solução para se fazer Engenharia de Tráfego na Internet de maneira eficiente e com custo razoável.

O MPLS (Multi-Protocol Label Switching) é um mecanismo de encaminhamento de pacotes, que reúne técnicas de transmissão orientadas a conexão e protocolos de roteamento, para simplificar o processamento de pacotes nos roteadores que ficam no caminho entre a origem e o destino do fluxo [Awduche, 1999]. Os pacotes são rotulados ao entrar em um domínio com suporte a MPLS e a classificação e o posterior encaminhamento por uma determinada rota são definidos baseados nestes rótulos. O MPLS surge como aliado da Engenharia de Tráfego uma vez que apresenta a funcionalidade de criação de rotas explícitas e, através desse mecanismo, possibilita a otimização da utilização dos recursos da rede.

Como o MPLS desacopla roteamento de pacotes de encaminhamento de pacotes, ele é capaz de suportar diferentes políticas de roteamento; o que é difícil ou quase impossível de se fazer apenas com o encaminhamento convencional da camada de rede. Esse trabalho trata do problema de seleção das rotas físicas dos LSPs (Label Switched Paths) em redes MPLS (Multi-Protocol Label Switching), que consiste na definição de rotas para os LSPs que minimizem o número total de saltos utilizados no roteamento e o número total de requisições rejeitadas.

Definição do Problema e Contribuições

O objetivo deste trabalho é o projeto e a validação através de simulação de um arcabouço para engenharia de tráfego em redes MPLS. Dentre os problemas de engenharia de tráfego existentes, foi dada prioridade ao problema de definição de rotas físicas para LSPs.

Inicialmente propomos um modelo linear inteiro para solucionar o problema de maneira exata. Esse modelo tenta balancear a carga na rede enquanto minimiza a taxa de rejeição. O objetivo é otimizar o desempenho global da rede roteando as requisições através de enlaces sub-utilizados.

O problema de roteamento em redes MPLS é NP-completo e o tempo necessário para se encontrar a solução exata para o mesmo é inviável na prática. Assim, heurísticas foram propostas para resolver o modelo e foi feita uma comparação das soluções obtidas com as heurísticas com a solução ótima obtida através do CPLEX [ILOG, 2002].

Nossa heurística é baseada em algoritmos genéticos, onde utilizamos a combinação de políticas de roteamento com movimentos adaptativos de rotas. As três políticas de roteamento utilizadas são baseadas em menor caminho, maior balanceamento ou limitação de tráfego. Os

movimentos adaptativos são utilizados para ajuste do roteamento, fornecendo maior flexibilidade às políticas. Quando há possibilidade de rejeição de uma requisição, reroteamentos são realizados para atender a essa requisição sem um grande prejuízo das requisições já estabelecidas.

Para analisar o desempenho do algoritmo genético, foram realizadas várias simulações. Os resultados mostram que o algoritmo genético executa muito mais rápido que o CPLEX. Os resultados também mostram que o uso do algoritmo genético diminui consideravelmente a quantidade de requisições rejeitadas na rede.

Foram executados vários experimentos e resultados satistatórios foram encontrados, mostrando que o algoritmo genético é também apropriado para o roteamento *on-line*. Deve-se ressaltar que há um aumento no custo do roteamento *on-line* devido ao reroteamento necessário.

Além do caso clássico de chegada de novas requisições, esse trabalho também trata do problema de ajuste do roteamento em diversos outros casos de mudanças na rede. Chamamos de *grooming* o processo de ajuste *on-line* do roteamento da rede. Ele ocorre tanto quando uma nova requisição surge quanto um novo elemento de rede é configurado, LSPs são terminados ou elementos de rede se recuperam de falhas. O *grooming* da rede é necessário quando decisões de roteamento tomadas no passado não são mais eficientes no presente.

Um mecanismo de detecção de mudanças de tráfego é proposto para engatilhar o processo de *grooming*. Esse mecanismo é baseado em cartas de controle. Resultados de simulação indicam que um melhor desempenho é alcançado através do uso dos mecanismos propostos.

Trabalhos Relacionados

Alguns conceitos básicos referentes à QoS que são tratados neste trabalho são apresentados em [Guerin et al., 1997], [Crawley et al., 1998], [Xiao et al., 1999], entre outros. Em [Awduche, 1999], a aplicação de MPLS para Engenharia de Tráfego em redes IP é discutida.

Um dos esquemas de roteamento dinâmico mais citados, Minimum Interference Routing Algorithm (MIRA) [Kar et al., 2000], é baseado em uma heurística para seleção de rotas *on-line*. A idéia principal da heurística é explorar o conhecimento dos pares de entrada e saída da rede, buscando evitar rotear as requisições em enlaces que poderão interferir em requisições

futuras.

As principais desvantagens desse esquema são a complexidade computacional necessária para implementá-lo, a falta de balanceamento de carga obtida em algumas topologias, como demonstrado em [Wang et al., 2002] e o fato de que ele não leva em consideração a carga atual da rede na tomada de decisões, como demonstrado em [Boutaba et al., 2002]. Nosso esquema tenta manter seu custo computacional similar ao custo de algoritmos de roteamento tradicionais e tenta conciliar simplicidade, acurácia e custo computacional.

Em [Yilmaz and Matta, 2002] um estudo do algoritmo MIRA e de um algoritmo *Profile-Based* é apresentado mostrando topologias onde o desempenho do algoritmo MIRA é melhor que o do algoritmo *Profile-Based*. Em [Figueiredo et al., 2004] um novo algoritmo de mínima interferência é apresentado. A vantagem desse algortimo é que o mesmo não computa o fluxo máximo para todos os casos, como no caso do MIRA. São comparados os resultados desse novo algoritmo, com os resultados do MIRA e de um algoritmo baseado no menor caminho com maior banda (WSP). Os autores não consideram entretanto reroteamento e apresentam uma função objetivo diferente da apresentada nesse trabalho.

Em [Salvadori and Battiti, 2003] um esquema de balanceamento de carga para redes MPLS é apresentado. A idéia é minimizar o congestionamento da rede através de alterações locais em algumas rotas. A principal diferença entre o esquema proposto nessa tese e o esquema proposto por eles é que eles buscam o melhor caminho alternativo para rerotear e a sua função objetivo é maximizar a capacidade disponível na rede. Em nossa proposta, o algoritmo pretende minimizar o número de rejeições e, em caso de reroteamento, ele busca por uma solução alternativa que satisfaça a demanda e não exceda o número de saltos da solução ótima por uma grande margem. Essa solução alternativa não é necessariamente ótima, já que a busca por uma solução alternativa não é feita como em [Salvadori and Battiti, 2003].

O esquema de Salvadori apresenta um aspecto preventivo uma vez que ele inicia o procedimento de reroteamento quando a configuração de um LSP causa a detecção de uma situação de congestionamento. Nosso esquema não apresenta o aspecto preventivo e tenta rerotear algum LSP apenas quando uma nova requisição chega e não há rota disponível com um número de saltos aceitável.

Em [Nobre et al., 2005] um esquema de balanceamento de carga baseado no trabalho de

[Salvadori and Battiti, 2003] é apresentado. A principal diferença é a função objetivo utilizada que trata o problema de atraso causado pelo uso de rotas longas no reroteamento. Um algoritmo *on-line* muito interessante também foi apresentado em [Boutaba et al., 2002], embora eles não considerem reroteamento em seu esquema.

Alguns modelos lineares inteiros foram propostos para solucionar o problema de roteamento *offline* em redes MPLS, tais como [Girish et al., 2000], [Liu, 2003], [Chou, 2004], [Dias et al., 2003] e [Dias and Camponogara, 2003]. Outros modelos foram apresentados em áreas afins como roteamento de circuitos [Resende and Ribeiro, 2003]. Em [Fortz and Thorup, 2000], o objetivo dos autores foi otimizar o estabelecimento de pesos nos enlaces, para o protocolo OSPF (Open Shortest Path First), baseado nas demandas projetadas.

Todos os modelos tratam de uma função com um único objetivo, com exceção de [Chou, 2004]. Não é apresentado como o modelo poderia ser utilizado em uma versão *on-line* do problema. Não são apresentados resultados para diversas topologias e tráfegos. Nesse trabalho, além de compararmos a solução obtida pela heurística com a solução exata, experimentos foram feitos simulando a rede em operação.

[Dias et al., 2003], [Dias and Camponogara, 2003] e [Resende and Ribeiro, 2003] utilizam diferentes heurísticas para resolver os modelos propostos: relaxação lagrangeana e GRASP, respectivamente. A vantagem da utilização da heurística evolucionária é que várias políticas podem ser usadas para inicialização da população incluíndo políticas utilizadas na solução obtida pelo GRASP, por exemplo.

Em [Capone et al., 2006] alguns dos principais esquemas de roteamento dinâmico são revisados e seu desempenho é comparado. Nenhuma referência é feita a esquemas baseados em algoritmos genéticos.

Roteamento em Redes MPLS

Para solucionar o problema de roteamento em redes MPLS, um modelo linear inteiro de duas fases foi proposto. Sua função objetivo é a minimização do número de rejeições de requisições de LSPs e a minimização do número de saltos utilizado no roteamento.

No primeiro passo, minimizamos a utilização máxima dos enlaces e o problema é formulado como mostrado a seguir:

$$P1 : Min\alpha \tag{1}$$

Subject to:

$$\sum_{k=1}^{K} x_{ij}^k \leq \alpha\mu_{ij} \qquad \forall(i,j) \in E \tag{2}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(j,i)\in E} x_{ji}^k = d_i^k \qquad \forall i \in V, \forall k \in K \tag{3}$$

$$x_{ij}^k \in 0,1 \qquad \forall(i,j) \in E, \forall k \in K \tag{4}$$

Façamos α^* o valor ótimo obtido no primeiro passo da otimização. O segundo passo da otimização é minimizar o número de saltos, respeitando o limite de utilização definido no primeiro passo. A formulação matemática para o segundo passo é mostrada no modelo P2.

$$P2 : Min \sum_{(i,j)\in E} \sum_{k=1}^{K} x_{ij}^k + \sum_{k=1}^{K} M(1 - a^k) \tag{5}$$

Subject to:

$$\sum_{k=1}^{K} x_{ij}^k \leq \alpha^*\mu_{ij} \qquad \forall(i,j) \in E \tag{6}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(j,i)\in E} x_{ji}^k = d_i^k a^k \qquad \forall i \in V, \forall k \in K \tag{7}$$

$$\sum_{k=1}^{K} a^k >= C \tag{8}$$

$$x_{ij}^k \in 0,1 \qquad \forall(i,j) \in E, \forall k \in K \tag{9}$$

$$a^k \in 0,1 \qquad \forall k \in K \tag{10}$$

O modelo de dois passos foi resolvido utilizando-se o CPLEX. Notamos porém que o tempo de execução observado foi muito alto, tornando a utilização do CPLEX para solução do modelo inviável na prática.

Demonstramos todavia nesse capítulo as vantagens do uso de técnicas de otimização na definição das rotas dos LSPs. As simulações da rede em operação, utilizando a ferramenta de simulação NS [McCanne and Floyd, 2003], onde todas as mensagens do protocolo são levadas em conta, demonstram as vantagens da definição das rotas físicas através do modelo linear em contra-posição com a definição das rotas físicas utilizando o menor caminho.

Algoritmos Genéticos

As técnicas para se resolver problemas de otimização podem ser divididas em métodos exatos e métodos heurísticos. Métodos exatos apresentam uma solução ótima no final do processamento, mas podem levar um tempo exponencial para obter essa solução. Por outro lado, métodos heurísticos não garantem a obtenção de soluções ótimas, mas eles tipicamente obtêm uma solução próxima da ótima em um tempo relativamente curto e permitem que informações adicionais sejam usadas na busca pela solução.

A natureza do problema que estamos tratando [Fortz et al., 2002] não permite que utilizemos métodos exatos para resolver instâncias de tamanho moderado, quiçá instâncias reais. Para lidar como essas dificuldades, propomos o uso de heurísticas baseadas em algoritmos evolutivos para a solução do problema de roteamento. Especificamente, implementamos um algoritmo genético e comparamos seu resultado com a solução ótima. Validamos o uso de algoritmos genéticos para resolver o modelo e obter soluções próximas da ótima em um tempo factível de ser usado na vida real.

Usando Algoritmos Genéticos *Online*

Anteriormente descrevemos nossa contribuição na implementação de um algoritmo genético que alcança um desempenho próximo do ótimo, em um tempo razoável. Nosso esquema de roteamento conseguiu reduzir substancialmente o número de requisições rejeitadas em comparação com o esquema básico utilizado atualmente.

Nossa solução utiliza o conhecimento prévio das requisições, ou seja, assumimos que todos os LSPs a serem roteados são conhecidos no momento em que o roteamento é feito. Na prática entretanto é comum que novas requisições aparecam, novos elementos de rede tenham que ser configurados e enlaces e/ou nós falhem. Consequentemente, mecanismos de roteamento devem ser adaptativos usando uma estratégia *on-line*.

Nesse capítulo, demonstramos como o genético, proposto para resolver o problema de maneira *offline*, pode ser utilizado *on-line*. Foram executados vários experimentos e resultados satistatórios foram encontrados.

Monitoração de Tráfego e *Grooming* da Rede

Uma vez que decisões *on-line* podem resultar em soluções não ótimas e já que o tráfego pode mudar com o tempo, as rotas devem ser constantemente monitoradas e em alguns casos alteradas. Propomos um mecanismo para lidar com a natureza dinâmica do tráfego na Internet baseado na monitoração do tráfego utilizando cartas de controle. O objetivo de cartas de controle é detectar se um determinado processo está sob controle, ou seja, o processo apresenta as métricas esperadas.

Uma vez que a notificação de toda mudança em um enlace não é escalável, o esquema proposto tenta determinar se a mudança percebida é uma variação natural ou se é devida a um fato inesperado. Dessa forma, as notificações de mudança do estado de um enlace ocorrerão somente em casos reais de alteração do tráfego, o que reduz o volume de dados na rede. Nesse trabalho, esse mecanismo é usado para definir o momento em que o *grooming* deve acontecer. Chamamos *grooming* o processo de ajuste do roteamento da rede. O *grooming* da rede é necessário quando decisões de roteamento tomadas no passado não são mais eficientes no presente.

Aplicamos cartas de controles computadas de duas maneiras: EWMA e XBar [Montgomery, 1990]. Diversos experimentos foram executados com dados sintéticos para demonstrar as principais vantagens de cada tipo de carta. Cartas XBar devem ser usadas quando mudanças em maior escala devem ser detectadas. Cartas EWMA devem ser usadas quando é necessário responder rapidamente a pequenas mudanças.

Experimentos mostram que é vantajoso o uso de cartas de controle para detectar mudanças no tráfego da rede e permitir que os protocolos de roteamento respondam apropriadamente a elas.

Conclusões

Nessa tese o problema de roteamento offline e *on-line* de LSPs em redes MPLS foi tratado. A qualidade de serviço em redes MPLS pode ser melhorado através do cálculo de rotas explícitas para os LSPs. As principais contribuições dessa tese são:

- Propusemos um modelo linear inteiro para o problema de roteamento *offline* de LSPs

em uma rede MPLS. O modelo tenta balancear a carga da rede, minimizando a sua taxa de rejeição.

- Propusemos um algoritmo genético que resolve o problema de roteamento de LSPs em uma rede MPLS de maneira heurística. Esse algoritmo é baseado em várias políticas, que roteiam os LSPs usando diferentes critérios, e baseado na combinação dessas políticas com os movimentos adaptativos. Os movimentos adaptativos são responsáveis pelo ajuste da rede em casos onde as políticas não são suficientemente flexíveis. A execução do algoritmo genético é muito mais rápida que a execução da ferramenta comercial CPLEX, que resolve o modelo de maneira exata. O uso das rotas calculadas pelo algoritmo genético são melhores que as rotas computadas pelo esquema de roteamento padrão do MPLS.

- Propusemos e verificamos o uso do algoritmo genético para resolver o problema do roteamento dinâmico em redes MPLS.

- Propusemos uma técnica de *grooming* e um mecanismo para detecção de mudanças de tráfego. Demonstramos como a estratégia de *grooming* pode ser aplicada em caso de um aumento dos recursos da rede. Demonstramos também como o mecanismo de detecção de mudanças na utilização dos enlaces pode ser utilizado para iniciar o processo de *goorming*.

O modelo linear inteiro foi resolvido pelo CPLEX. Entretanto, o tempo necessário para solucionar o modelo de maneira exata não é viável na prática. Por isso, o algoritmo genético foi desenvolvido para solucionar o modelo de maneira heurística. O modelo linear inteiro foi muito útil como uma ferramenta formal de definição do problema, mas sua solução de maneira exata consome muito tempo, já que o mesmo pode ser classificado como sendo um problema NP-difícil.

Várias simulações foram feitas para se verificar o desempenho do algoritmo genético em diferentes cenários de tráfego, com diferentes topologias de rede e diferentes políticas de roteamento. Foram simulados cenários sem e com o uso de movimentos adaptativos, de maneira *on-line* e *offline*.

Os resultados das simulações foram comparados com os resultados obtidos pelo CPLEX. Os resultados obtidos pelo algoritmo genético foram bastante satisfatórios, demonstrando que o mesmo é capaz de extrair a melhor solução apresentada pelas diversas políticas.

Estudamos o uso do algoritmo genético *on-line* junto com técnicas de reroteamento, que podem acontecer quando um novo elemento de rede é configurado, LSPs são finalizados ou um elemento de rede se recupera de falhas. Apresentamos a utilização de cartas de controle como mecanismo de detecção de mudanças de utilização dos enlaces. As cartas de controle são o gatilho para o processo de ajuste da rede, definindo o momento em que o reroteamento deve ser feito. Simulações indicam que a utilização desses mecanismos resulta em um melhor desempenho da rede.

Devido à enorme importância de flexibilidade e capacidade de adaptação, acreditamos que algoritmos genéticos são muito convenientes para roteamento *on-line*. O algoritmo genético proposto nesse trabalho pode ser estendido com novas políticas de roteamento.

Trabalhos Futuros

Há várias maneiras de se estender o trabalho apresentado nessa tese. O algoritmo genético proposto pode ser estendido com a inclusão de outras políticas assim como outros operadores genéticos. Uma versão multi-objetivo poderia ser desenvolvida, o que traria maior flexibilidade para os administradores de rede.

Outra importante face do processo de roteamento que pode ser investigada é seu aspecto temporal. A granularidade das decisões de roteamento podem variar em uma escala menor entre *offline* e *on-line*.

Nesse trabalho mostramos que o uso de técnicas de ajuste da rede associadas com cartas de controle podem melhorar o desempenho das redes. Acreditamos que é importante estender a capacidade das ferramentas de monitoração, já que as mesmas são o gatilho do processo de ajuste da rede. Uma importante extensão a esse trabalho seria o desenvolvimento de cartas de controle com regras adicionais.

Ainda em relação ao processo de ajuste de rotas (*grooming* de rotas), um fato que deveria ser investigado é o uso de ferramentas de predição de tráfego. Junto com ferramentas de monitoração, técnicas de predição podem levar o processo de *grooming* a obter melhores

resultados. Nesse trabalho, assumimos que o ajsute da rede ocorrerá após a detecção de uma situação anormal. Com a ajuda de ferramentas de predição de tráfego, o ajuste poderá ser preventivo.

Outra possibilidade é analisar a influência da topologia da rede nas políticas propostas. Notamos que redes com a mesma carga apresentam comportamento diferente devido à sua topologia.

Finalmente, como trabalho futuro, há o desenvolvimento de uma versão usável e portável do simulador, assim como o desenvolvimento de documentação. Isso ajudaria o trabalho dos novos estudantes do laboratório.

Adriana de Andrade Oliveira

A Framework for Traffic Engineering in MPLS Networks

Thesis presented to the Doctoral Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Federal University of Minas Gerais
Belo Horizonte, October, 2005

Abstract

This work addresses the problem of physical route selection for Label Switched Paths (LSPs) in Multi-Protocol Label Switching (MPLS) networks. It consists of defining routes for LSPs trying to minimize the total number of hops and the total number of rejections. The LSPs path selection problem is addressed through two different approaches: on-line and offline.

We propose an ILP (Integer Linear Programming) model to solve the offline problem in an exact manner. This model tries to balance the network load while minimizing the network rejection rate. The goal is to optimize the overall network performance by routing requests through under-utilized links. Issues concerning the execution time necessary to solve the model are noticed and a genetic algorithm (GA) is developed to address theses issues.

The GA is based on the combination of routing policies and adaptive route movements. In order to analyze the genetic algorithm's performance, various simulations are conducted. Results show that the GA executes much faster than the commercial tool used to solve the model in an exact manner. Results also show that the use of the GA considerably decreases the amount of rejected requests in the network. Experiments are conducted and satisfactory results are found showing that the GA is also suitable for on-line routing.

This work also addresses the problem of adjusting the network routing in other cases besides the arrival of new requests. The process of adjusting the network routing is called *Grooming* and it occurs when a new network element is configured, LSPs are torn down or a network element recovers from failure. It is necessary when routing decisions taken in the past are no longer efficient.

A traffic/resource change detection mechanism is proposed to trigger the *Grooming* process. This mechanism is based on control charts and is used in the context of defining the moment the Grooming should take place. Simulation results indicate that better performance is achieved using the proposed mechanisms.

Contents

List of Acronyms

AB Adaptive Load Balanced

ALU Adaptive Limited Utilization

AMH Adaptive Min Hop

ARS Adaptive Routing Scheme

B Load Balanced

EWMA Exponentially Weight Moving Average

GA Genetic Algorithm

ILP Integer Linear Programming

IP Internet Protocol

LSP Label Switched Path

LU Limited Utilization

MH Min Hop

MPLS Multi-Protocol Label Switching

NS Network Simulator

OSPF Open Shortest Path First

QoS Quality of Service

SNMP Simple Network Management Protocol

SPC Statistical Process Control

TE Traffic Engineering

MIRA Minimum Interference Routing Algorithm

WSP Widest Shortest Path

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we present our motivation, the problem definition and our objectives. We also present a summary of our contribution and the organization of this thesis.

1.1 Motivation

In recent years there has been a tremendous growth of the Internet. Various real-time services are being deployed and new applications such as streaming media and voice over IP present new traffic patterns and new demands to the network. Pressures are being placed on Internet protocols to support quality of service.

The Internet is currently based on the best-effort paradigm, which, despite being highly scalable, cannot provide the hard guarantees that is desired by most time-critical bandwidth intensive applications. Normally, IP traffic follows rules established by routing protocols, such as OSPF. Each router computes the shortest paths using weights assigned by the network operator, and creates destination tables used to direct each IP packet to the next router on the path to its final destination. There is no service differentiation in IP networks.

Besides no service differentiation, shortest path destination based routing often leads to unbalanced traffic distribution across the network. Sometimes the traffic flows through the same set of links, which are part of the shortest path, creating the so called *hot spots*, while other parts of the network have very light traffic loads.

This fact has prompted a fairly large research effort to harness the unused resources to run

useful work. There is a need for traffic engineering tools, since controlling how traffic flows through the network is one of the traffic engineering functions.

The ability to control the traffic is precisely what the MPLS [Awduche et al., 1999] technology provides. MPLS is a packet label-based switching technique where packets are assigned a short and fixed-length data header, a label, which identifies the path the packets will follow in the network as well as the treatment the packets will receive in the network. MPLS allows sophisticated routing control capabilities to be introduced into IP networks, such as explicit routing, and can help to build backbone networks that better support QoS traffic. MPLS plays a key role by providing services unsupported by the IP protocol.

Traffic Engineering is an essential ingredient for guaranteeing QoS and for efficient design and operation of IP/MPLS networks. Nevertheless the use of MPLS and its explicit route feature depends on the use of optimization techniques to define the best routes for the LSPs.

This work addresses the problem of traffic engineering in MPLS networks by presenting a new routing scheme that copes with congestion and lack of load balancing. It uses the explicit routing feature of a MPLS network associated with constraint-based routing and rerouting techniques.

Our research goal aims to deliver an adaptive routing scheme that is able to automatically and dynamically manage network traffic within a delimited administrative domain. This scheme would be capable of automatically rerouting traffic, redirecting it from congested paths to less congested routes, thereby improving overall network performance.

At the heart of this scheme lies an optimization algorithm, responsible for granting and refusing admission to new traffic requests and establishing the best routes according to a number of criteria for the granted requests.

1.2 Problem Definition and Objectives

This work addresses the problem of defining the route configuration for LSPs in a MPLS capable network. We are particularly interested in using optimization techniques to find the best routes available.

The problem we are addressing in this thesis can be stated as follows: given a set of requests, how to configure the routes in the network so that the number of requests rejected

and the routing cost are minimized? The routing cost in this work is defined as the total number of hops used to route the LSP requests.

The path selection problem is addressed through three different approaches. We solve the problem statically, based on its integer linear formulation, we solve it statically using a genetic algorithm, and we also solve it dynamically using the same GA.

To our knowledge there is only one work using GA in the context of MPLS networks [Hong et al., 2003] and their work differs in very important aspects such as the genetic representation of the problem and the genetic operators. Their objective function does not take into account the possibility of rejections. Their definition and implementation of the genetic representation is different from ours and much more complex as well as the definition and implementation of the genetic operators. Our work has a simple and clear definition and implementation of the genetic representation as well of the genetic operators and still presents fast and reliable results.

There are other related works such as [Buriol, 2003, Buriol et al., 2005, Ericsson et al., 2002] but they are used in the context of OSPF. The GA is used to set weights to the links and it does not differentiate routes when the request has the same origin-destination pair.

The GA together with the traffic change detection mechanism and the *grooming* techniques proposed in this thesis represent a new approach in the design of routing solutions to MPLS networks.

In order to analyze the heuristic's performance some scenarios are studied and simulations are conducted. Simulation results show that when using the GA the amount of rejected requests in the network decreases. Furthermore, results also indicate that we can approach the best performance in the LSPs routing using *grooming* techniques.

1.3 Contributions

This work presents the design and evaluation of a new adaptive routing scheme for MPLS capable networks with load balancing, and minimal rejection rates and minimal cost goals. To achieve these objectives, we model the path selection mechanism as an integer linear problem and use an evolutionary heuristic to solve it, since the problem is proved to be NP-hard

[Fortz et al., 2002].

Besides being used to solve the offline version of this problem, we also propose the use of the same genetic algorithm on-line. The policies used in the GA together with adaptive route movements are suitable for dynamic routing, since by dynamically changing routes the algorithm accepts more requests and improves the network performance. To evaluate the proposed routing scheme simulations are built and numerical results are presented.

We also propose a traffic change detection mechanism to be used with the adaptive movements. Since some on-line routing decisions can lead to non-optimal solutions and since traffic can change over time, the routes should be constantly monitored and in some cases even *groomed*.

To cope with the traffic change detection problem, control charts are applied. Experiments are run with two types of control charts: EWMA and XBar-Charts. They show that is feasible to use control charts for traffic change detection, since it provides information about the network conditions to the routing algorithms.

In summary, the main contributions of this work are:

- The use of an evolutionary heuristic based on genetic algorithms to solve the problem of routing LSPs in a MPLS network.

- The use of the genetic algorithm on-line.

- The design of a traffic change detection tool that monitors the network.

- The combination of the traffic change detection tool with the path selection module. With this combination, better results can be obtained since we are defining routes for LSPs based on topology configuration and traffic conditions as well.

Another important contribution of this thesis, although it is not a primary contribution, is the implementation of a framework that combines all aspects of the simulations. We have used it to study several scenarios of traffic and topology definition and our intention is to make it available to other research groups.

1.4 Organization of this work

This work is organized as follows: in the next chapter we briefly survey the related work and available technology. In Chapter 3, we present the mathematical model and its exact solution. Chapter 3 also presents some simulations performed using NS [McCanne and Floyd, 2003] to show the effectiveness of the proposed solution in the operational environment. In Chapter 4, we present the genetic algorithm developed and the simulations of various network scenarios. Chapter 5 presents the results for using the GA on-line and discusses some important issues that appear when traffic or topology changes. Chapter 6 describes how control charts can be used as a traffic change detection mechanism. Finally, Chapter 7 brings the conclusion of this thesis and the possibilities for future work.

Chapter 2

Background

In this chapter, we will present some background information on IP routing, traffic engineering and MPLS. We also discuss related work and how our work differs from it.

2.1 Intra-Domain IP Routing

The exponential growth of the Internet has placed heavy burdens on network management and control operations. The Internet is expected to become a carrier for voice, video and data applications. To support the requirements of multimedia applications it is essential to incorporate new technologies into its infrastructure. QoS provisioning and resource usage optimization are two essential attributes that these new technologies should have.

The optimized use of resources is a necessary step in order to avoid traffic congestion and degradation of services. Adding more resources to the network may temporarily relieve congestion conditions, but it is not a cost-effective solution in the long-run. The optimized use of resources is accomplished by traffic engineering that consists of a number of procedures such as traffic measurements, characterization and load balancing [Liu et al., 2000].

The effectiveness of traffic engineering is directly governed by the routing process, which controls the flow of requests. Shortest path routing is sufficient for achieving connectivity but does not always make good use of available network resources and is not satisfactory from a traffic engineering point of view. Besides, in traditional IP routing, all packets with the same destination address have to follow the same path through the network and these paths have

often been computed based on static and single link metrics. These problems can cause traffic concentration and thus degradation in quality of service.

The problems can be illustrated by the famous *fish* example in Figure 2.1. Because of IP's shortest path destination based routing, all traffic from R1 to R6 and R2 to R6 will follow the upper path. Consequently, the sub-path R3-R4-R5 may get over-utilized while the alternate path R3-R7-R8-R5 stays under-utilized.

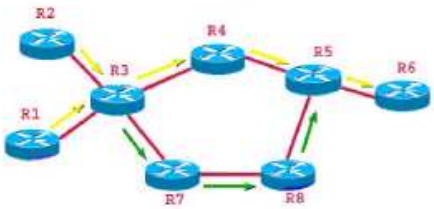


Figure 2.1: The Fish

Nevertheless, it was shown in [Savage et al., 1999] that at any time there are many potential paths through the internet connecting any two hosts. They also showed that there is a great diversity in the end-to-end performance observed on the Internet. In their work, for 30-80% of the paths they examined, there was an alternate path with significantly superior quality.

In this context, solutions were proposed to increase utilization of the network; among them are per-packet dynamic routing and per-flow explicit routing. Although per-packet dynamic routing is effective for load balancing, it is difficult to deploy because of potential routing oscillations [Yilmaz and Matta, 2002]. Consequently, the focus changed to solutions using explicitly routed paths.

2.2 Traffic Engineering

The best effort nature of the current Internet makes it virtually impossible to provide minimal QoS guarantees for the applications. Currently in IP networks, packets are routed at each node based only on the destination address stored at their headers. Packets belonging to distinct applications but with the same source-destination pair may pass through the same path. In an IP network, various links stay underutilized most of the time, while others are congested, creating the known *hot spot* problem.

This presents at least two inconvenient situations. First, the application packets with different QoS requirements will receive the same treatment, which may compromise the offering of guaranteed QoS for those packets demanding the highest QoS level. Secondly, this forwarding approach produces an uneven utilization of the routes going to the same destination.

This is the main motivation for optimizing resource utilization as well as for the adoption of intelligent network load balancing. Traffic Engineering (TE) is the process of controlling how traffic flows through a network so as to optimize resource utilization and network performance.

A number of techniques were proposed to provide TE in IP networks. Among the most popular are Integrated Services (IntServ) [Seaman et al., 2000], Differentiated Services (DiffServ) [Blake et al., 1998], Constraint-Based Routing (CBR) [Apostolopoulos et al., 1999], and, more recently, Multi Protocol Label Switching [Awduche et al., 1999, Boyle et al., 2002].

IntServ and DiffServ were the two first attempts of the Internet community to support QoS in the Internet. The IntServ solution weakness is its lack of scalability, since the resource reservation process is made on a per-flow basis. DiffServ relies on traffic conditioners sitting at the edge of the network to perform traffic engineering functions such as: traffic classification, marking, shaping and policing. Nevertheless, the DiffServ model does not attempt to guarantee a level of service. It rather strives for a relative ordering of aggregations such that one traffic aggregation will receive better or worse treatment relative to other aggregations. Admission control at the boundary does not consider the availability of resources in the DiffServ network region along a specific path.

Constraint-Based Routing (CBR) is a process that is able to find paths that are subject to multiple quantitative as well as qualitative constraints.

Multi-Protocol Label Switching has been widely recognized as an important traffic engineering tool for IP networks. The explicit routing feature of MPLS was introduced to address the shortcomings associated with current IP routing schemes, which are hampered by the requirement to forward packets based only on destination addresses along shortest paths computed using mostly static and non-traffic related link's metrics. With the MPLS explicit route feature, the flow no longer has to follow the shortest path route defined for it.

Several ongoing works are already proposing mechanisms to combine some of the techniques described. CBR, for example, is an important tool to be used in conjunction with MPLS for

arranging how traffic flows through the network and improving its utilization.

Although all proposed techniques allow an increase in the quality of services provided, they do not address the problem of dynamically balancing the traffic on the network to achieve congestion control. Our work addresses this problem. It tries to control the routing scheme of an MPLS network based on a series of policies, adaptive movements of routes and a traffic change detection tool.

2.2.1 MPLS

In recent years there has been active research in the field of MPLS and an increasing number of networks are supporting this technology. MPLS has recently emerged to facilitate the network traffic engineering, which can be achieved by directing packet flows over paths that satisfy multiple requirements.

MPLS is a switching technology to forward packets based on a short, fixed length identifier, a label. Labels are used as indexes of a table that contains the connection path. Using indexing instead of longest prefix matching, MPLS achieves fast forwarding.

An MPLS network consists of label switched paths (LSPs) and edge/core label switch routers (LSRs). The LSRs store the label translation tables. Core LSRs provide transit services in the middle of the network while edge LSRs provide an interface with external networks.

LSPs are virtual unidirectional paths established from the sender to the receiver. Packets with identical labels are forwarded on the same LSP. An extension of the Resource reSerVation Protocol (RSVP) is used to establish and maintain LSPs in the backbone.

In [Awduche, 1999], the applications of MPLS to traffic engineering in IP networks are discussed. Many problems such as the definition of the network topology, LSP dimensioning, LSP setup/tear-down procedures, LSP routing and LSP adaptation for incoming resources requests need to be solved.

Two different approaches can be used for MPLS network design: traffic-driven and topology-driven. In the traffic-driven approach, the LSP is established on demand according to a request for a flow, traffic trunk or bandwidth reservation. The LSP is released when the request becomes inactive.

In the topology-driven approach, the LSP is established in advance according to the routing protocol information, i.e., when a routing entry is generated by the routing protocol. The advantage of the traffic-driven approach is that only the required LSPs are setup, while in the topology-driven approach, the LSPs are established in advance even if no data flow occurs. Our work can be classified as using a traffic-driven approach.

2.3 Related Work

Reactive congestion management policies react to congestion by initiating relevant actions to reduce them, while preventive policies prevent congestion on the basis of estimates of future potential problems. Most of the proposed congestion control schemes in the literature are preventive: they allocate paths in the network in order to prevent congestion, while only a few are reactive, which means they act only when problems start to appear [Salvadori and Battiti, 2003]. Our scheme can be classified as hybrid. It uses adaptive movements to reroute LSPs when there is a possibility that a new LSPs will be discarded, which gives an reactive characteristic to the scheme. But it also has a preventive facet since the policies can control the utilization on each link.

One of the most cited preventive dynamic routing schemes for MPLS networks, Minimum Interference Routing Algorithm (MIRA) [Kar et al., 2000], is based on a heuristic dynamic online path selection algorithm. The key idea is to exploit the knowledge of ingress-egress pairs to avoid routing over links that could *interfere* with potential future path setups. The main weaknesses of this scheme are the computational complexity necessary to implement it, the unbalanced network utilization for some network topologies as shown in [Wang et al., 2002] and the fact that it does not take into account the current traffic load in routing decisions [Boutaba et al., 2002]. In our scheme, we tried to keep the computational cost of routing at a level comparable to that of traditional routing algorithms. Our scheme tries to compromise between simplicity, accuracy and computational cost.

In [Yilmaz and Matta, 2002] a study of MIRA and a Profile-Based algorithm is presented and they show topologies where the MIRA algorithm performs better.

In [Figueiredo et al., 2004] a new minimum interference routing algorithm was presented. The advantage of this algorithm is that it does not compute the maximum flow for critical

edges detection. They compare their results against MIRA and Widest Shortest Path (WSP). They do not consider rerouting in their scheme and their objective function is different than ours.

In [Salvadori and Battiti, 2003] a load-balancing scheme for an MPLS network was presented. The idea was to minimize the congestion of the network by performing local modifications to the routes. The main differences between our scheme and theirs is that they search the best alternative path to reroute and their objective function is to maximize the available capacity on the network. In our scheme, the algorithm wants to minimize the number of rejections and in case of rerouting it searches for *one* alternative path that satisfies the demand and has a number of hops that does not exceed the shortest path by much.

Their scheme also shows a preventive aspect since it triggers the rerouting procedure when the set up of a new LSP causes the detection of network congestion. Our scheme does not present this preventive aspect and tries to reroute a LSP only if a new request arrives and there is no route available or the one that is available is very long. Our scheme uses control charts to detect traffic changes on links that can enhance the actual network configuration and the process of enhancing the network is called *grooming*. It is not a preventive mechanism.

In [Nobre et al., 2005] a load balancing scheme is presented based on the work presented in [Salvadori and Battiti, 2003]. The main difference is the objective function since the authors do not address the delay problem caused by the use of longer routes.

A very interesting on-line algorithm was shown in [Boutaba et al., 2002], although they do not consider LSP rerouting in their scheme.

Some integer linear models were proposed to solve the problem of offline routing in MPLS networks, such as [Girish et al., 2000] and [Liu, 2003], [Chou, 2004], [Dias et al., 2003] and [Dias and Camponogara, 2003]. Other models were presented in similar areas such as circuit routing [Resende and Ribeiro, 2003]. All models deal with a single objective function with the exception of the work in [Chou, 2004]. They do not show how the model could be used in an on-line version of the problem, which is the more realistic case. Also they do not show results for various topologies and traffic scenarios. In our work besides comparing the solution obtained by the heuristic with the exact solution, we also run the experiments in an operational environment.

In [Dias et al., 2003], [Dias and Camponogara, 2003] and [Resende and Ribeiro, 2003] different heuristics are used to solve the model: Lagrangean relaxation and GRASP, respectively. The advantage of using an evolutionary heuristic as we do is that various policies can be used as the initial population including the policy used in the GRASP solution.

In [Capone et al., 2006] some of the major dynamic QoS routing schemes are reviewed and their performance is compared. No reference is made to any scheme based on genetic algorithms showing the novelty of our approach.

Chapter 3

Routing in MPLS Networks

In this chapter we propose a two-step ILP model, which has as its objective function the minimization of the number of rejections and of the number of hops when routing LSPs requests in MPLS networks. We also present some results obtained solving the model using CPLEX.

3.1 Introduction

A MPLS network provides a set of long-term virtual paths between endpoints on a backbone network. There are numerous choices in designing an MPLS-based traffic engineering scheme, but it is known that the effectiveness of this scheme is directly governed by the routing process.

This chapter deals with the problem of establishing routes for a set of LSP requests that take into account total number of hops and rejection rate, where the final objective function is a combination of the two criteria. In our approach, the objective function can be parameterized to change the utilization priority or number of rejections according to administrative rules.

Most of the traffic engineering problems that arise from the task of managing traffic in MPLS networks can be posed as an integer linear programming (ILP) problem. We point out that the single objective traffic engineering formulations proposed in the literature address only one particular aspect of the traffic engineering problem [Dias et al., 2003, Boutaba et al., 2002].

3.2 Mathematical Formulation

Let $G = (V, E)$ be a directed graph representing the MPLS network under study. Let E denote the set of links that connect the backbone nodes, and $V = 1, \cdots, n$ denote the set of backbone nodes, where MPLS routers reside. For each link $(i, j) \in E$, let μ_{ij} denote the link bandwidth (the maximum kbits/sec rate) allowed to be routed on the link or the link capacity.

The set K of LSPs to be routed is represented by a list of origin-destination (O-D) pairs

$$K = \{(o_1, d_1), (o_2, d_2), \cdots, (o_n, d_n)\}$$

where we associate with each pair a bandwidth requirement. Each commodity $k \in K$ is a LSP to be routed, associated with an origin-destination pair and with a bandwidth requirement or demand (d_i^k).

A route for LSP (o, d) is a sequence of adjacent links, where the first link originates in node o and the last link terminates in node d. A set of routing assignments is feasible, if for all links $(i, j) \in E$, the total LSP effective bandwidth requirements routed on (i, j) does not exceed $\alpha\mu_{ij}$, where α represents the maximum utilization allowed in the links.

Network load balancing is achieved by minimizing the load on the most utilized link. Routing assignments with minimum LSP delays may not achieve the best link load balance. Likewise, routing assignments having the best link load balance may not minimize LSP delays. A compromising objective is to route all LSPs in set K such that a desired point in the trade-off curve between LSP delays and link load balancing is achieved. Our problem is modeled using an approach consisting of two steps. In the first step, we minimize the maximum link utilization and the problem is stated as follows:

The ultimate objective is to minimize the load, α, on the most utilized link. The first group of constraints (3.2) imposes limits on traffic over the links, while the second group of constraints (3.3) guarantees flow conservation. The demand d_i^k takes the value 1 if the node i is a LSP ingress node, it takes the value -1 if the node i is a LSP egress node and it takes the value 0 otherwise. The variable x_{ij}^k takes the value 1 if, and only if, the virtual path of the $k - th$ LSP goes through the link (i, j).

Let α^* be the optimal value of α obtained in the first optimization step. The second

$$P1 : Min\alpha \tag{3.1}$$

Subject to:

$$\sum_{k=1}^{K} x_{ij}^k \leq \alpha\mu_{ij} \qquad \forall(i,j) \in E \tag{3.2}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(j,i)\in E} x_{ji}^k = d_i^k \qquad \forall i \in V, \forall k \in K \tag{3.3}$$

$$x_{ij}^k \in 0,1 \qquad \forall(i,j) \in E, \forall k \in K \tag{3.4}$$

optimization step is to minimize the cost subject to the constraint that all link utilization remains under α^*.

The mathematical formulation for the second step is shown in Model P2. The parameter M indicates the penalty given to rejections. In our experiments, M is set to 10. The ultimate objective is to minimize the number of hops and number of rejections while balancing the load.

If α^* is less than 1, it means that no LSP rejections have to happen. If α^* is greater than 1, our solution will allow rejections to happen.

$$P2 : Min \sum_{(i,j)\in E} \sum_{k=1}^{K} x_{ij}^k + \sum_{k=1}^{K} M(1 - a^k) \tag{3.5}$$

Subject to:

$$\sum_{k=1}^{K} x_{ij}^k \leq \alpha^*\mu_{ij} \qquad \forall(i,j) \in E \tag{3.6}$$

$$\sum_{(i,j)\in E} x_{ij}^k - \sum_{(j,i)\in E} x_{ji}^k = d_i^k a^k \qquad \forall i \in V, \forall k \in K \tag{3.7}$$

$$\sum_{k=1}^{K} a^k >= C \tag{3.8}$$

$$x_{ij}^k \in 0,1 \qquad \forall(i,j) \in E, \forall k \in K \tag{3.9}$$

$$a^k \in 0,1 \qquad \forall k \in K \tag{3.10}$$

The constraints in the second optimization step are the same as those in the first step, except for constraint (3.7) and (3.8). The variable a^k takes the value 1 if, and only if, the $k-th$ LSP is accepted into the network, allowing the network to reject requests. The parameter C

indicates the minimum number of LSPs that need to be provided and it is used in the constraint (3.8). Ideally, C should be equal to the number of LSP requests.

The mathematical programming formulation of this problem consists of tasks that are computationally hard and for which no polynomial-time algorithm is known. The proposed optimization problem is NP-hard [Fortz et al., 2002] and involves a large number of variables.

In [Salvadori and Battiti, 2003], it was shown that min max utilization is an appropriate objective function for a heavily loaded network, while end to end delay is a more appropriate optimization objective for lightly loaded conditions, at which most of the backbones are operating. The approach presented is flexible enough to deal with both scenarios.

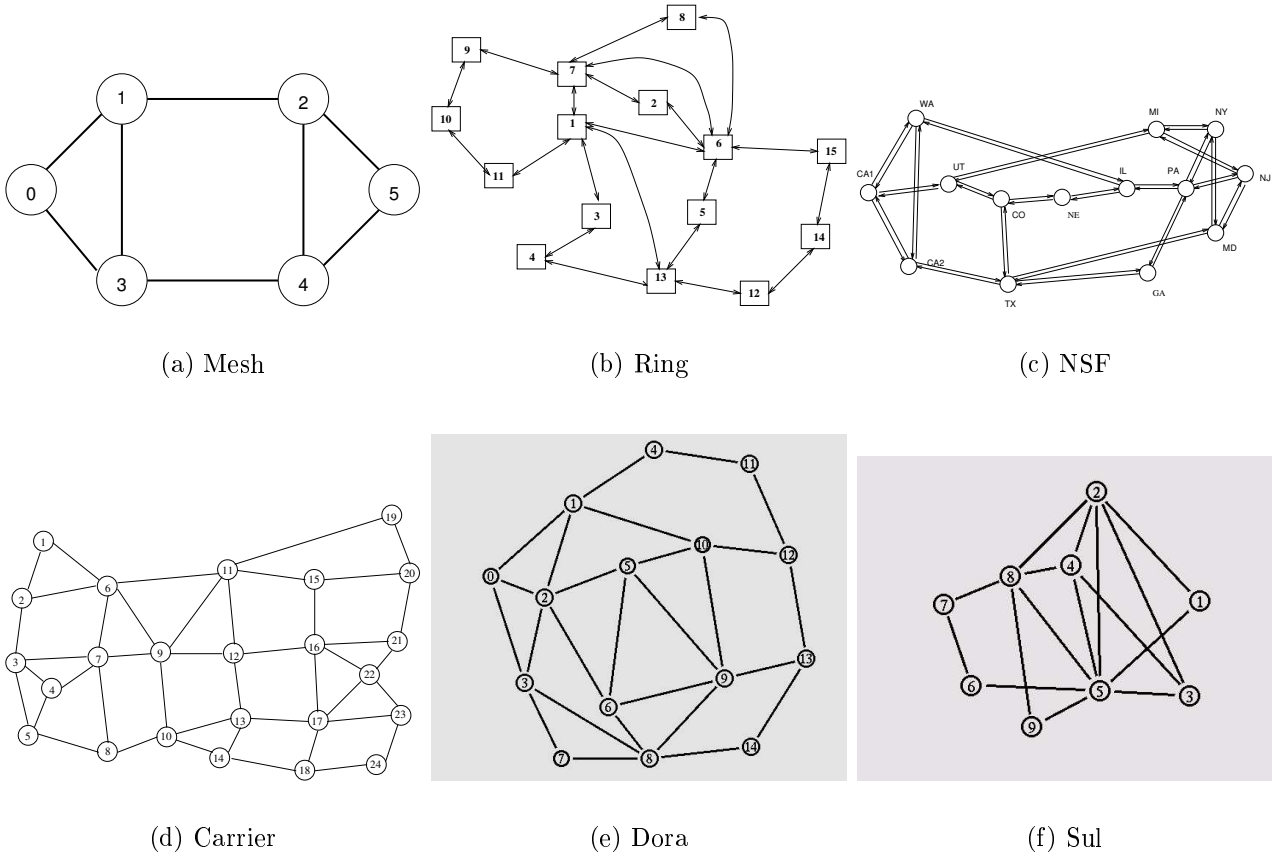
3.3 Experiments

We solved the two-step model using CPLEX [ILOG, 2002] for a series of topologies and traffic scenarios. The model was written using AMPL [Fourer, 2002]. The CPLEX version was 9.0. The topologies and traffic scenarios used in the experiments are described in the next subsections.

3.3.1 Topologies

The network topologies used in this work are shown in Figure 3.1. They were chosen because they represent either real networks (Ring, NSF, Carrier) or they were used in related studies as examples of common topologies (Dora, Sul). Some detailed information about the network topology's characteristics is presented on Table 3.1.

The simple 6-node network has mesh characteristics. It is used for illustrative purposes and to get the numerical examples for large complex problems. The second topology is a telecommunication metropolitan network, composed of four rings. This is a typical topology showing how today's backbone networks are interconnected. The third topology is widely used as an illustrative wide-area backbone network topology. The fourth topology is a modified version of a well connected carrier's IP backbone topology, widely used for simulation experiments. The fifth and sixth topologies were used in related work [Dias et al., 2003, Kar et al., 2000, Boutaba et al., 2002].



(a) Mesh (b) Ring (c) NSF

(d) Carrier (e) Dora (f) Sul

Figure 3.1: Topologies Used in the Experiments

Table 3.1: Topologies Characterization

Topology	Nodes	Bi-directional Links	Avg Nodal Degree	Avg. Num. Hops
Mesh	6	8	2.67	1.53
Ring	15	21	2.8	2.42
NSF	14	19	3.0	2.14
Carrier	24	43	3.58	2.99
Dora	15	26	3.46	2.25
Sul	9	16	3.22	1.72

3.3.2 Traffic Scenarios

The sets of requests used in the experiments range from light to heavy traffic scenarios. For the simulation study, source-destination pairs were chosen by chance to represent the LSPs and are defined in Table 3.2.

Each simulation consists of a set of CBR type data flows, each with transmission rates set

Table 3.2: LSP Definitions

Topology	LSPs
Mesh	(0,4) (2,0) (3,2) (4,1) (5,3)
Ring	(3,1) (8,5) (10,14) (11,6) (12,7)
NSF	(0,5) (1,12) (4,13) (7,3) (11,6)
Carrier	(1,12) (3,5) (3,10) (7,13) (11,18) (14,5) (15,22) (20,21) (20,24)
Dora	(0,12) (3,1) (4,8) (4,14)
Sul	(1,6) (1,8) (1,9) (2,5) (2,6) (5,2) (6,2) (8,1) (8,2) (8,5)

at 200 Kbps. The number of data flows varied from 10 up to 50. The purpose of this workload is to contrast the performance of the proposed algorithm with that of conventional routing.

Based on the LSPs definition presented in Table 3.2 we generated various traffic scenarios by configuring the number of requests to vary from 10 to 50 with a step of 10 requests as described in [Boutaba et al., 2002, Kar et al., 2000, Dias and Camponogara, 2003, Salvadori and Battiti, 2003]. For each traffic scenario and topology, we show the results over 5 run trials and the standard deviation.

3.3.3 CPLEX

Computing the Minimum Load on the Most Utilized Link - Computing α^*

Our first step was to find the solution for our test scenarios using the two-step model proposed. We executed the Model P1 using CPLEX and therefore computed the minimum load on the most utilized link for all traffic scenarios and topologies.

The results are described in Table 3.3. Each element of the Table 3.3 reports the average minimum load on the most utilized link for each number of flows over 5 run trials and its standard deviation (within brackets).

As expected, the load in the most utilized link increases as the number of flows increases. It can be noticed that there are values in the Table 3.3 greater than 1.0. This means that some LSPs will be rejected when the second model is executed.

We noticed from Table 3.4 that the Model P1 can take more that 30 minutes to execute in some cases. No conclusion could be made about the relation of the number of flows and the

Table 3.3: α^* - Minimum Load on the Most Utilized Link (1.0 represents 100%)

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.72 (0.10)	1.32 (0.16)	1.96 (0.20)	2.64 (0.15)	3.04 (0.15)
Ring	0.56 (0.15)	0.84 (0.08)	1.40 (0.13)	1.64 (0.08)	2.08 (0.16)
NSF	0.48 (0.10)	0.76 (0.08)	1.04 (0.08)	1.40 (0.13)	1.84 (0.20)
Carrier	0.36 (0.08)	0.60 (0.00)	0.80 (0.13)	1.0 (0.00)	1.3 (0.10)
Dora	0.33 (0.00)	0.50 (0.00)	0.67 (0.00)	0.86 (0.07)	1.17 (0.00)
Sul	0.88 (0.16)	1.44 (0.20)	1.92 (0.39)	2.98 (0.39)	3.36 (0.54)

Table 3.4: CPLEX Execution Time in Seconds (Max. 30min)

Top.	Number of Flows				
	10	20	30	40	50
Mesh	0.04 (0.07)	0.02 (0.00)	0.02 (0.01)	363.81 (813.44)	0.05 (0.00)
Ring	0.02 (0.01)	362.19 (809.79)	723.78 (990.96)	1088.41 (888.61)	361.71 (723.13)
NSF	0.03 (0.01)	375.43 (801.44)	722.83 (989.65)	1102.77 (964.09)	361.34 (807.50)
Carrier	653.36 (746.87)	362.25 (807.24)	362.14 (809.37)	1809.59 (2.52)	1813.18 (1.38)
Dora	9.76 (4.99)	743.44 (973.78)	1084.52 (989.93)	361.92 (809.02)	1810.43 (0.44)
Sul	0.02 (0.00)	0.03 (0.01)	0.04 (0.01)	0.05 (0.01)	0.07 (0.02)

time necessary to solve the model. It seems that the model can be solved quickly when the combination topology-traffic has a reduced number of options.

This experiment shows that the execution time is heavily dependent on the topology and the source-destination pairs. It also shows that the execution time in CPLEX varies a lot. Therefore, the use of CPLEX to solve the model in the operational environment will not be possible.

The combinatorial nature of the problem makes the execution time very long even for small networks. These experiments showed the problem of performance that we are facing; since this problem is NP-hard [Fortz et al., 2002].

Minimum Number of Hops and Minimum Rejection Rate

After computing α^*, the second step, model P2, which was created to allow LSP rejections was implemented using CPLEX. In this model, one LSP can be accepted or not. If it is accepted, it obeys the capacity constraints and tries to find the shortest possible path, keeping the network load balanced.

As previously explained, when we use the maximum load computed by the first step of the model as a constrain, we obtain a solution where the traffic is balanced. This is true in the cases where the maximum load is less than 1.0. If we set the maximum load to 1.0 in all links, as OSPF, the solution will not be balanced.

The results for this step are presented in Tables 3.5 and 3.6, that represent the OSPF solution and the balanced solution respectively.

Table 3.5: Minimum Number of Hops and Minimum Rejection With a Non Balanced Load

Topology	Number of Flows									
	10		20		30		40		50	
	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops
Mesh	0	20	1	48	11	148	20	240	30	340
Ring	0	29	0	68	2	137	10	231	20	331
NSF	0	30	0	63	0	103	6	184	17	289
Carrier	0	25	0	54	0	77	0	134	6	189
Dora	0	33	0	68	0	98	0	137	1	204
Sul	0	14	3	57	10	131	5	117	14	206

Table 3.6: Minimum Number of Hops and Minimum Rejection With a Balanced Load

Topology	Number of Flows									
	10		20		30		40		50	
	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops
Mesh	0	20	1	48	11	148	20	240	30	340
Ring	0	35	0	81	2	137	10	231	20	331
NSF	0	30	0	70	0	103	6	184	17	289
Carrier	0	28	0	58	0	87	0	134	6	189
Dora	0	35	0	73	0	109	0	146	1	204
Sul	0	16	3	57	10	131	5	117	14	206

It is noticeable that there is a small increase in the number of hops for the balanced solution when the network load is less than 1.0. When the network is overloaded, the solutions are the same. Despite having the same total number of hops or very similar results, the selected routes were sometimes very different, meaning that they were using different routes with equal number of hops.

Although the results look the same or even worse for the balanced solution, we could notice after simulating the networks in operation that using alternative routes help avoiding

congestion in some links and improve average packet delay and average number of drops. The results are presented in the next Section.

It is important to notice that the load computed by the model does not take into account the overhead due to the protocol messages. Therefore, it is important to simulate the solutions in an operational environment as will be shown in the next section.

Another conclusion is that it is worth caring for load balance in cases were changing the load in some links could avoid rejections and this point will be explored in the next chapter.

3.4 NS Simulations

A series of experiments were conducted to demonstrate the following points: (1) the need for optimization techniques to better configure the LSPs in a MPLS network and (2) the need for simulations of the network in operation taking into account protocol messages overhead.

These experiments show how a naive configuration of the LSPs leads to bad network performance when compared to a smarter one, obtained by our model. To simulate the MPLS network in the operational environment, simulations were conducted using the well known Network Simulator [McCanne and Floyd, 2003].

NS is an event-driven network simulator. It has an extensible background engine implemented in C++ that uses OTcl as the command and configuration interface. Thus, a simulation is defined by an OTcl script.

Algorithm 1 OTcl Script Defining the Main Simulation

```
1: set ns [new Simulator]
2: set na [open nsf_MPLS.tr w]
3: set nf [open nsf_MPLS.nam w]
4: Classifier/Addr/MPLS set control_driven 1
5: $ns trace-all $na
6: $ns namtrace-all $nf
7: $ns at 0.0 "record"
8: source topo.tcl
9: source trafego.tcl
10: source start.tcl
11: source rotas.tcl
12: source stop.tcl
13: $ns at 10.1 "finish"
14: $ns run
```

In a OTcl script, we have to define: a network topology and a traffic pattern. We can also include commands to collect statistics and to write simulation results to an output file.

Algorithm 1 shows the main script used in our simulations. This script uses various other scripts to define specific parts of the simulation, such as topology, traffic, events, explicit routes, statistical purposes commands.

Algorithm 2 shows some of the methods used to set up the explict routes computed by our Two-Step Model. The method setup-erlsp is one of the methods include in the MPLS package. The first line on this example shows that there will be an explicit route going from node 0 to node 2, passing through node 1.

Algorithm 2 OTcl Script Defining the Explicit Routes for LSPs

```
1: $ns at 0.0 "$LSRn0 setup-erlsp 2 0_1_2 322"
2: $ns at 0.0 "$LSRn3 setup-erlsp 4 3_4 40"
3: $ns at 0.0 "$LSRn5 setup-erlsp 1 5_2_1 412"
4: $ns at 0.0 "$LSRn1 setup-erlsp 0 1_0 200"
```

A Java program is responsible for generating the OTcl scripts. It uses as input files the network topology, the traffic file and the explicit routes computed by our two-step model.

3.4.1 Default MPLS x MPLS with Two-Step Model

Our hypothesis is that the routes found by the optimization method are much better than the routes found using the default approach. Even when the load in the network is under 1.0, there will be congestion and packets dropped due to delays caused by the protocols messages. The number of packets dropped will decrease substantially and the overall throughput will increase if the proposed strategy is implemented, instead of the default approach.

The experiments are conducted for all topologies and traffic scenarios where the network maximum load as computed by the model P1 is less than 1.0 - see Table 3.3. The metrics used to compare the solutions are: the number of packets dropped and the average packet delay. In theory, no packet should be dropped.

Each element of the Table 3.7 reports the percentage of packet drops for each number of flows in NS using the default routing scheme over 5 run trials and its standard deviation(within brackets).

Table 3.7: Percentage of Packets Dropped when using MPLS Default Scheme

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.01 (0.02)	–	–	–	–
Ring	0.00 (0.00)	0.13 (0.04)	–	–	–
NSF	0.03 (0.05)	0.22 (0.04)	0.36 (0.00)	–	–
Carrier	0.00 (0.00)	0.03 (0.04)	0.08 (0.04)	0.16 (0.10)	–
Dora	0.00 (0.00)	0.04 (0.01)	0.21 (0.04)	0.33 (0.04)	–
Sul	0.01 (0.01)	–	–	–	–

Table 3.8: Percentage of Packets Dropped when using Routes Computed by Two-Step Model

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.0 (0.00)	–	–	–	–
Ring	0.0 (0.00)	0.03 (0.04)	–	–	–
NSF	0.0 (0.00)	0.0 (0.00)	0.05 (0.02)	–	–
Carrier	0.0 (0.00)	0.0 (0.00)	0.02 (0.03)	0.05 (0.01)	–
Dora	0.0 (0.00)	0.0 (0.00)	0.01 (0.01)	0.05 (0.04)	–
Sul	0.0 (0.00)	–	–	–	–

Tables 3.8 and 3.7 shows that our approach outperforms conventional routing. Under normal operation conditions the packet discard rate is almost null for our solution, while sometimes over 10 % for conventional routing. Under heavily loaded traffic conditions, number of flows greater than 30, the packet discard rate is over 30% for conventional routing whereas below 3% if our solution is implemented.

Table 3.9: Average Packet Delay when using Routes Computed by Two-Step Model

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.08 (0.00)	—	—	—	—
Ring	0.10 (0.00)	0.18 (0.07)	—	—	—
NSF	0.09 (0.01)	0.10 (0.01)	0.23 (0.01)	—	—
Carrier	0.09 (0.00)	0.10 (0.00)	0.13 (0.04)	0.20 (0.02)	—
Dora	0.09 (0.01)	0.09 (0.01)	0.14 (0.03)	0.18 (0.03)	—
Sul	0.08 (0.00)	—	—	—	—

Tables 3.9 and 3.10 plot the average packet delay time and the average route size for the two-step solution. Tables 3.11 and 3.12 plot the average packet delay time and the average

Table 3.10: Average Route Size when using Routes Computed by Two-Step Model

Topology	Number of Flows				
	10	20	30	40	50
Mesh	3.94 (0.06)	–	–	–	–
Ring	5.42 (0.23)	4.97 (0.23)	–	–	–
NSF	5.24 (0.12)	5.20 (0.12)	4.95 (0.12)	–	–
Carrier	5.06 (0.28)	5.27 (0.15)	4.84 (0.26)	4.91 (0.01)	–
Dora	5.12 (0.17)	5.42 (0.20)	5.24 (0.15)	5.04 (0.26)	–
Sul	3.70 (0.08)	–	–	–	–

route size for the MPLS default scheme solution.

Table 3.11: Average Packet Delay when using Routes Computed by MPLS Default Scheme

Topology	Number of Flows				
	10	20	30	40	50
Mesh	0.09 (0.06)	—	—	—	—
Ring	0.07 (0.00)	0.17 (0.05)	—	—	—
NSF	0.12 (0.06)	0.24 (0.04)	0.23 (0.00)	—	—
Carrier	0.05 (0.00)	0.07 (0.03)	0.10 (0.02)	0.15 (0.01)	—
Dora	0.06 (0.00)	0.13 (0.00)	0.16 (0.02)	0.15 (0.02)	—
Sul	0.13 (0.07)	—	—	—	—

Table 3.12: Average Route Size when using Routes Computed by MPLS Default Scheme

Topology	Number of Flows				
	10	20	30	40	50
Mesh	3.38 (0.12)	–	–	–	–
Ring	3.65 (0.13)	3.18 (0.24)	–	–	–
NSF	3.53 (0.26)	2.94 (0.24)	2.23 (0.02)	–	–
Carrier	2.90 (0.08)	2.86 (0.09)	2.77 (0.20)	2.41 (0.38)	–
Dora	3.68 (0.17)	3.83 (0.08)	3.09 (0.26)	2.53 (0.24)	–
Sul	2.89 (0.18)	–	–	–	–

When we compare the average packet delay in the two cases, Tables 3.9 and 3.11, we can notice that the average delay increases. We could notice that the routes where congested when using the default MPLS scheme cause the average packet drop is much higher, varying from 10% up to 40% when the load increases, results shown in Table 3.7. The increase in average delay is probably due to the increase on the number of packets in the MPLS network, since

the rejection rate is smaller. Therefore, the increase in delay is compensated by the decrease in the number of packets dropped.

The degradation experienced by the MPLS default scheme is slightly higher than the degradation experienced by the two-step model scheme as showed in Table 3.13. The degradation was computed dividing the maximum average packet delay by the minimum average packet delay for all traffic scenarios.

Table 3.13: Degradation of Average Packet Delay for the Schemes

Topology	Default Scheme	Two-Step Scheme
Mesh	–	–
Ring	2.43	1.8
NSF	2.00	2.56
Carrier	3.00	2.22
Dora	2.66	2.00
Sul	–	–

The experiments described in this section show the importance of using alternative paths to balance the traffic load in MPLS networks. The packet discard rate is considerably reduced by the use of the routes defined by the two-step model.

3.5 Concluding Remarks

In this chapter, we presented a formal model to solve the problem of LSP routing in MPLS networks. The model was executed using CPLEX and it was noticed that another method to solve the model is necessary since CPLEX takes too long to find the exact answer.

This chapter also demonstrates the effectiveness of the application of optimization techniques to traffic management in computer networks. It validates the use of optimization techniques for improving performance of IP networks over MPLS. It is a contribution to achieve QoS in MPLS networks.

Another importance of this chapter is that it presents the simulations run in an operational environment, where signaling and resource reservation messages where taken into account.

Chapter 4

Genetic Algorithms

As discussed in Chapter 3, the mathematical programming formulation for the routing problem consists of tasks that are computationally hard and for which no polynomial-time algorithm is known. The proposed optimization problem is NP-hard [Fortz et al., 2002] and involves a large number of variables.

The techniques for solving IP problems can be broadly divided into exact and heuristic methods. Exact methods produce an optimal solution at termination, but can take exponential time to reach one. Heuristic methods on the other hand do not have guarantees but they typically yield near-optimal solutions quickly and allow human insight to be embedded into the solution procedure.

In this chapter, we propose the use of heuristics as a means to yield near-optimal solutions, but in a reasonably short time period. Our heuristic is derived from evolutionary algorithms.

4.1 Background

A genetic algorithm (GA) is a search technique used to find approximate solutions to combinatorial optimization problems [Reeves, 1993]. Genetic algorithms are a particular class of evolutionary algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, natural selection, and recombination (or crossover) [Goldberg, 1989]. A genetic algorithm pseudo code is shown in 3.

Genetic algorithms are typically implemented as a computer simulation in which a popu-

Algorithm 3 Genetic Algorithm Pseudo Code

```
1: Choose initial population of size M
2: repeat
3:    Evaluate the fitnesses of the population
4:    Sort the population based on fitness
5:    Select (C) pairs to be parent's candidates
6:    Mate pairs at random
7:    Apply crossover operator
8:    Keep only one child
9:    Select (M-C) best-ranking individuals
10:   to form the elite
11: until terminating condition
```

lation of abstract representations of candidate solutions to an optimization problem evolves toward better solutions. The abstract representations of candidate solutions are called chromosomes, individuals or organisms.

Initially several chromosomes are generated to form an initial pool of possible solutions: the first generation pool. This may be totally random, or we may seed the gene pool with *hints* to form an initial pool of possible solutions. In our approach, we used policies to create the initial pool and this will be better explained in Section 4.3.

The evolution starts from the first generation pool and happens in generations. During each successive generation, each chromosome is evaluated, and a value of fitness is returned. The fitness function represent the condition of being suitable. In genetic algorithms, fitness is used to guide the search by deciding which chromosomes will be used as future points to look for better solutions. The pool is sorted based on their fitness values, with those having better fitness ranked at the top, the fitness function will represent our objective function.

The next step is to generate a second generation pool of chromosomes, which is done using any or all of the genetic operators: selection, crossover and mutation. A pair of chromosomes are selected for breeding. In Holland's original GA [Holland, 1973], one parent is selected on a fitness basis while the other parent is chosen randomly. Various possibilities exist for the selection of parents.

Following selection, the crossover operation is performed upon the selected chromosomes. They are then *mated* by choosing a crossover point X at random, with the offspring consisting of the pre-X section of one parent followed by the post-X section of the other parent. One

of the existing population is chosen at random and replaced by one of the offspring. Most genetic algorithms will have a probability of crossover, typically between 0.6 and 1.0.

In another version of this procedure, the whole population is changed *en bloc* after a number of M trials, rather than incrementally. Many other variations are possible: both parents can be selected on a fitness basis, mutation can be applied, more than one crossover point can be used, and so on. However, in many cases, the simple genetic algorithm involving just crossover and mutation has proved to be powerful enough. This process is repeated until a chromosome is produced which gives a solution that is *good enough*.

The three most important aspects of using genetic algorithms are: (1) definition and implementation of the genetic representation, (2) definition and implementation of the genetic operators, and (3) definition of the fitness or objective function . Once these three have been defined, the generic genetic algorithm should work fairly well.

In the following subsections, we will describe in detail our approach. The genetic algorithm was implemented using Java.

4.2 Genetic Representation

In our approach, the chromosome represents a solution for the problem, i.e, the set of paths that can be used by all LSPs. Each gene in the chromosome represents a possible path for a specific LSP. Each chromosome has the number of genes equal to the number of LSPs in the problem. The locus of a gene is the LSP identification $k = 1, \cdots, K$. Thus, the value of the $i-th$ gene of the chromosome represents a possible path for the $i-th$ LSP.

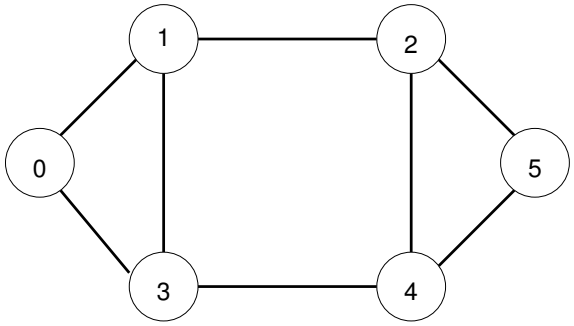


Figure 4.1: Example - Mesh Topology

Table 4.1: LSP Requests

Topology	LSPs
Mesh	(0,4) (3,2) (5,3)

Let's use as an example the network shown in Figure 4.1. Given the LSP's requests shown in Table 4.1, we will show how the genetic representation would work.

The gene alleles represent the gene possible values. Here the alleles represent the possible routes for each LSP. In our solution, we limited the number of possible routes to a parameter K. We work with the K first shortest paths. If $K = 2$ we would have a configuration as shown in Table 4.2.

Table 4.2: Possible Gene Values

LSP	Allele 1	Allele 2
1	0-3-4	0-1-2-4
2	3-1-2	3-4-2
3	5-4-3	5-2-4-3

When $K = 2$ and the number of chromosomes in our population is three, we can have the genes and chromosomes configuration shown in Table 4.3 at some time in our simulation. This configuration can represent the first generation pool or some generation obtained due to some genetic operator.

Table 4.3: Possible Chromosome Values

Chromosome	Gene 1	Gene 2	Gene 3
1	0-3-4	3-1-2	5-4-3
2	0-1-2-4	3-4-2	5-4-3
3	0-3-4	3-4-2	5-2-4-3

4.3 Population Initialization

Our first attempt was to define the first generation pool using a random function. Therefore, in a pre-processing phase of the algorithm, the first K shortest paths available for each LSP

were calculated, using Djikstra's algorithm. Then, for each LSP, one of the K LSPs routes was chosen by chance, defining the genes for the chromosomes.

Besides the number of alternative routes, K, the population size or number of chromosomes is also a parameter in our implementation.

4.3.1 Using Policies

After some experiments, we noticed that the initial population could be improved leading to a faster convergence of the algorithm to a better solution. Hence, three policies were defined to address the problem of initial population definition. Instead of picking a route by chance, we started using different criteria to choose the routes to compose the chromosomes. The polices implemented are: min-hop (MH), limited utilization (LU) and load balance (B) and will be described in detail.

MH - Min Hop

This algorithm tries to grant to each LSP request the shortest route possible for that LSP. The final solution would have the minimum number of hops possible.

This algorithm works as following: for each LSP request, it is verified if the shortest route residual bandwidth is sufficient to satisfy the LSP demand. If yes, the LSP is allocated to this route, otherwise the next route is checked and this process is repeated till all possible routes are verified.

In other words, in the min-hop algorithm, the path from the ingress to the egress node with the least number of *feasible* links is chosen. This algorithm is the most commonly used algorithm for routing LSPs.

It is worth noticing that in this algorithm the LSPs are routed based on their arrival order.

LU - Limited Utilization

This algorithm tries to balance the network load imposing a limit on link utilization to avoid congestion. It is an extension of the MH algorithm but besides being feasible the link utilization must be below a certain limit.

This algorithm works as following: for each LSP request, it is verified if the shortest route residual bandwidth is sufficient to satisfy the LSP demand and if it is below a certain limit. If yes, the LSP is allocated to this route, otherwise the next route is checked and this process is repeated till all possible routes are verified.

It is worth noticing that in this algorithm the LSPs are routed based on their arrival order.

B - Load Balance

This algorithm tries to balance the load among the routes. It looks for the route with the best residual/capacity ratio to allocate the LSP. It is also an extension of the MH algorithm.

We define the residual bandwidth along a link to be the difference between the bandwidth of the link and the sum of the LSP demands that are already routed on that link.

This algorithm works as following: for each LSP request, it is verified which route has the maximum value for the residual/capacity ratio and has sufficient bandwidth to satisfy the LSP demand. This route is allocated to this LSP.

It is worth noticing that in this algorithm the LSPs are routed based on their arrival order.

4.3.2 Adaptive Movements - Reducing Rejection

The policies described previously routed the LSP requests based on their arrival order. After running some experiments, we noticed that changing the routes of previously accepted requests could reduce the number of LSPs rejection. Hence an adaptive scheme was proposed to solve the problem of accommodating requests that would normally be rejected.

The adaptive movements idea came from [Salvadori and Battiti, 2003]. But in their work they *reroute* or *move* some LSPs to obtain the best solution. In our work [Oliveira et al., 2005, Oliveira and Mateus, 2005b], we will *reroute* or *move* a LSP only to allow another LSP to be accepted in the network. We will not test all possibilities, looking for the best solution. This small modification will improve the performance of our algorithm.

In summary, the adaptive movements or adaptive routing scheme (ARS) is a rerouting technique that will be combined with the policies previously described to decrease the network rejection rate.

The algorithm presented in 4 is the pseudo-code for the adaptive routing scheme. This

algorithm works as following: for each LSP request, a route is calculated, based on the chosen policy: MH, LU or B. If the policy can not find a route or if the route found is too long, the ARS tries to change previously accepted LSPs routes to accept the new LSP or to install it in a better way. A route is considered too long if it is at a distance D or greater than D of the shortest path.

The adaptive scheme then searches for the congested link in the first shortest path route. For this link, it checks the LSPs that were already allocated to that link - line 5. If there is a LSP that uses more bandwidth than that requested by the new request and can be moved to another route with a length increase of maximum D hops, lines 6-8, the LSP routes are changed, otherwise nothing is done. In [Salvadori and Battiti, 2003], this search will be exhaustive, compromising performance at some extent.

Algorithm 4 Adaptive Movements Pseudo Code

```
 1: route = alocateOneRoute(new LSP, demand);
 2: if route does not exist or route > minRoute+D then
 3:    find the congested link in one of the available routes;
 4:    repeat
 5:       for all  each LSP that crosses this link do
 6:          if (LSP.demand > newLSP.demand) then
 7:             find a newroute such as newroute ≤ LSP.route +D;
 8:          end if
 9:       end for
10:    until found or LSPs are finished;
11:    if found() then
12:       changeRoutes;
13:    end if
14: end if
```

In summary, the ARS will reroute a LSP only when a new LSP will be otherwise rejected or when the change of number of hops for the new one is considerable, which means that the overall delay of the network would be reduced as shows the condition in line 2 of the algorithm.

The policies previously defined are combined with the adaptive movements generated three new possible ways to establish the first generation pool: adaptive min-hop (AMH), adaptive limited utilization (ALU) and adaptive load balance (AB). Table 4.4 summarizes all strategies that will be used in the population initialization phase.

Table 4.4: Population Initialization Strategies

Policy	Description
MH	Uses feasible routes with minimal number of hops.
AMH	Uses feasible routes with minimal number of hops. Reroutes when necessary
LU	Uses feasible routes with minimal number of hops and with utilization below a certain limit.
ALU	Uses feasible routes with minimal number of hops and with utilization below a certain limit. Reroutes when necessary.
B	Uses feasible routes with minimal number of hops and with better residual/capacity ratio.
AB	Uses feasible routes with minimal number of hops and with better residual/capacity ratio. Reroutes when necessary.

4.4 Fitness Function

In genetic algorithms, fitness is used to guide the search by deciding which chromosomes will be used as future points to look for better solutions. The fitness function represent the condition of being suitable.

In our work, the fitness value of a chromosome will be the number of requests that are rejected due to lack of bandwidth in a path and the total number of links (hops) used in the solution, as shown in equation 4.1. M is a parameter that can be used by the network operator to give more priority to solutions with a lower number of hops or to solutions with a lower number of rejections.

$$Fitness(x) = total\ number\ of\ hops + M.total\ number\ of\ rejections \tag{4.1}$$

Our fitness function is like the objective function of the second step of our ILP model described in Chapter 3. Therefore, the best solutions will have the lowest fitness values.

The load balanced solutions obtained when using the first part of the ILP model will be obtained by the genetic algorithm through the use of the policies: AMH, B, AB, LU and ALU.

In our implementation, the fitness function will be used for the elite definition and algorithm termination.

4.5 Elite Definition

When creating a new population by crossover and mutation, we have a big chance to loose the best chromosome. Elitism is the name of the method that first copies the best chromosome (or few best chromosomes) to the new population. Elitism can rapidly increase the performance of GA, because it prevents the loss of the best solution found so far.

In our approach, some of the better chromosomes from one generation are carried over to the second, unaltered. To define the better chromosomes, the fitness function is used.

4.6 Selection Methods

The selection method determines how individuals are chosen for mating. If you use a selection method that picks only the best individual, then the population will quickly converge to that individual. So the selector should be biased toward better individuals, but should also pick some that are not quite as good, but hopefully have some good genetic material in them.

In our implementation, a number of individuals are picked using random selection as parents candidates. Then, for each mating iteration, two chromosomes are selected from the parent's candidates group. The fitness function is not used at this moment, since the bias toward the better solutions is already achieved by the use of elitism.

4.7 Heuristic Crossover

In our approach, the crossover occurs in the following way: for each mating pair, the parent's genes are compared. Since each gene is a route to a LSP, the metric used to compare the genes is their number of hops. The best gene (route with less number of hops) is copied to the offspring.

It is worth noticing that only one offspring is generated. With the genes selected, we compute the number of rejections and number of hops to calculate the offspring fitness value.

4.8 Experiments

The experiments will show the performance yielded by the heuristic against that obtained solving the integer linear formulation using CPLEX. Our hypothesis is that the genetic algorithm will be able to find a good solution much faster than using CPLEX. To compare the results, we used the same topologies and traffic scenarios described in Chapter 3.

Since the GA can obtain different results depending on the parameters configuration, we first determined the best GA solution and then compared this solution to the solution obtained by CPLEX.

4.8.1 Input Parameters Description

In our implementation, the genetic algorithm receives four parameters as the input:

- number of possible routes (K),
- population initialization strategy,
- population size,
- number of iterations.

The number of possible routes in the experiments varied from 2 to 5. This interval was chosen due to the observation of the average number of alternative routes available for the topologies studied. The average ranged from 2 to 5.

The way the population was initialized was also a parameter:

- 0 - when initialized by chance
- 1 - when initialized using the policies

If the initialization strategy is 0, all the chromosomes in the population are chosen randomly. If the initialization strategy is set to 1, the first 6 chromosomes were initialized using the policies and the rest was chosen randomly.

The population size varied from 5 to 25 with a step of 5. This interval was chosen due to the observation that the policies have a huge influence on the algorithm performance. Since the

number of policies is 6, varying the population size from 5 to 25 would allow some randomized solutions to be also present.

The number of iterations varied from 5 to 25 with a step of 5. Since the best solutions for the GA using policies were found in the first 5 iterations of the algorithm, we defined the interval to vary from 5 to 25 to allow the confirmation of the solution and not to pick a local optimal as solution.

4.8.2 Input Parameters Analysis

The 200 combinations of parameters - number of routes (4) x population initialization strategy (2) x population size (5) x number of iterations (5) - were run for all traffic scenarios and all topologies.

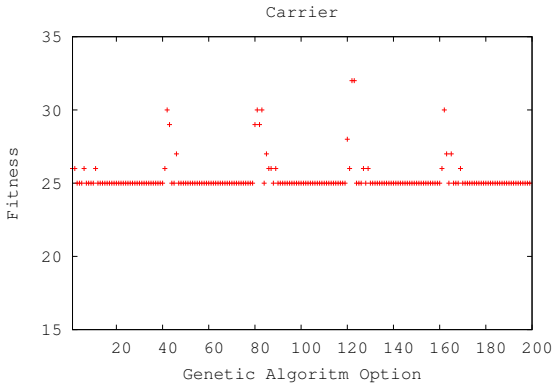
To analyze the performance of the genetic algorithm as well as the parameter influence on the GA solution, we decided to focus on the *Carrier* topology. This topology was chosen because it is a modified version of a well connected carrier's IP backbone topology, widely used for simulations experiments. It also presents traffic scenarios that range from lightly to heavily loaded, as can be seen in Table 3.3.

In Figure 4.2, we show the different results obtained for the different options of parameters. The results for other topologies can be seen in the Appendix. The graphs in Figure 4.2 show the fitness obtained with each possible parameter combination (200 points in the abscissa) with traffic varying from 10 to 50 LSP requests with a step of 10.

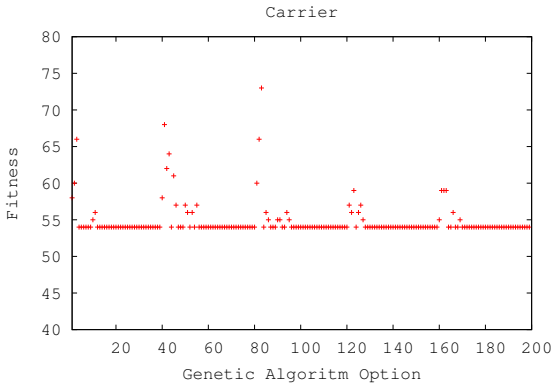
Since analyzing each combination individually would be very difficult, we decided to plot all 200 combinations to check if some of them would immediately show a better result.

It can be noticed that all graphics present clusters of regions where the fitness of the solutions are very similar, this regions are the horizontal lines in the graphs.

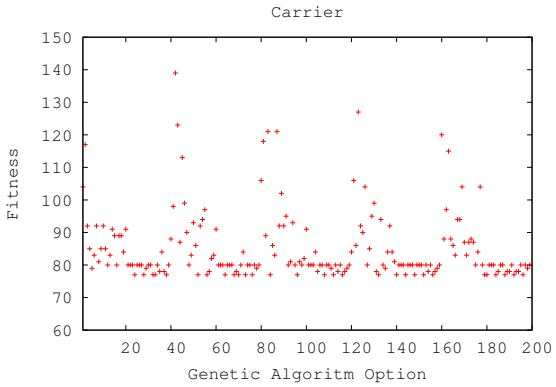
When traffic load is low (10 and 20) requests, the results for all combinations are practically the same. When traffic load increases, the parameters start to make a difference but even so it can be noticed that all graphs present horizontal lines, representing clusters of good solutions. We analyzed these clusters to check which parameters were influencing more the solutions.



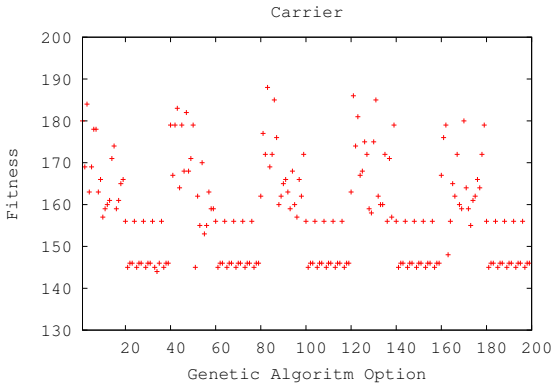
(a) Traffic Scenario - 10



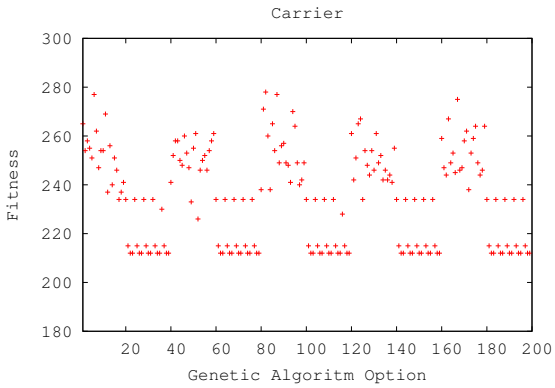
(b) Traffic Scenario - 20



(c) Traffic Scenario - 30



(d) Traffic Scenario - 40



(e) Traffic Scenario - 50

Figure 4.2: Various Results For Carrier Network Using Genetic Algorithms

Population Type

In Figure 4.3, it can be seen that the results obtained by initializing the population by chance, option of population initialization strategy set to 0, is inferior to the results obtained by initializing the population using the policies, option of population initialization strategy set to 1.

The results were shown when traffic is equal to 10, 30 and 50 requests. The number of points in each graph is now 100 since the population initialization strategy results are shown in two different graphs.

The best values (lower values) represent the solutions obtained by running the GA with the option of population initialization strategy set to 1. The solutions obtained initializing the population by chance are worse and are not as concentrated around some values as the ones obtained when initializing the population using policies. This result clearly shows that the use of the policies help the genetic algorithm to convergence faster to better solutions.

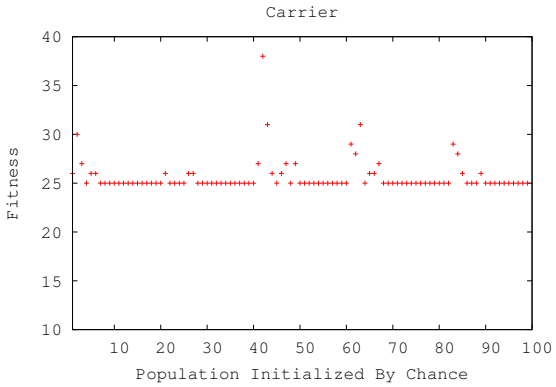
Number of Alternative Routes

It can be seen in Figure 4.3, for the graphs showing the results when the population was initialized using policies, that the number of alternative routes is not important when the traffic load is low. When the traffic load is **10** requests, the results were almost identical.

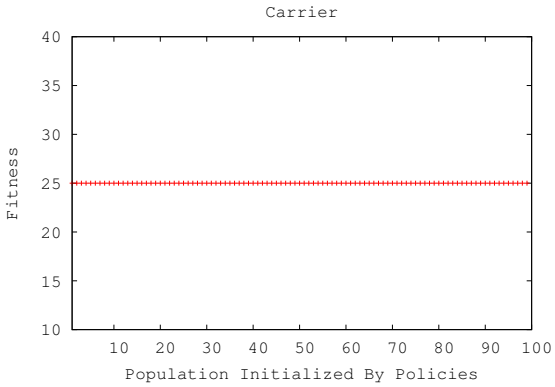
When the traffic load increases, it is noticeable that 2 or 3 horizontal lines appear in the graph. The best results when the traffic load is **30** requests are obtained when the number of alternative routes is 3. This can be explained by the fact that using 2 alternative routes was not enough to balance the traffic and using 4 or 5 alternative routes lead to longer routes.

When the traffic load is **50** requests, 3 lines are noticed. To certify the influence of the number of alternative routes, we analyzed in detail the results when the traffic load is 50 requests and the results are shown in Figure 4.4. The number of points in the graph for each line is now 25 since the results for different number of alternative routes are shown in four different lines.

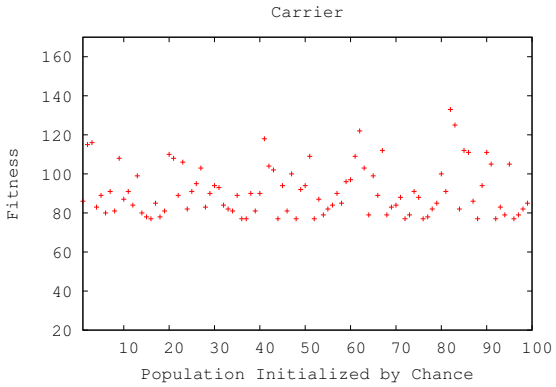
The worst solution is obtained when the number of alternative routes is set to 2. The solutions get better as we increase the number of alternative routes. The intermediate solution is obtained when the number of alternative routes is set to 3 and the best solution is obtained



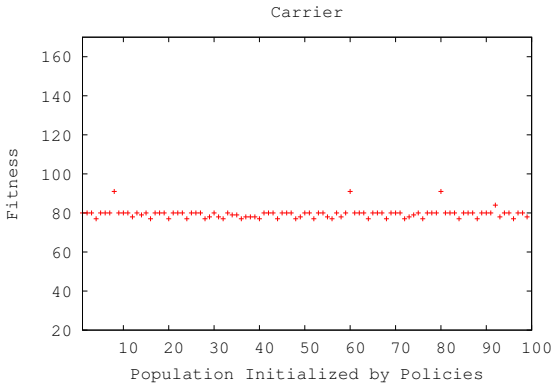
(a) Traffic 10 - Initialization By Chance



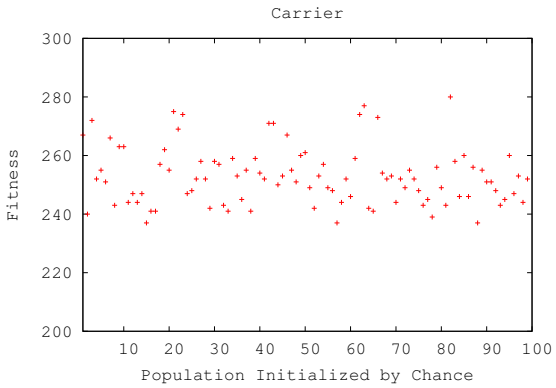
(b) Traffic 10 - Initialization By Policies



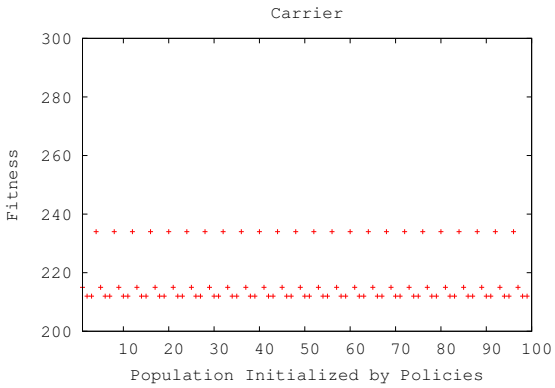
(c) Traffic 30 - Initialization By Chance



(d) Traffic 30 - Initialization By Policies



(e) Traffic 50 - Initialization By Chance



(f) Traffic 50 - Initialization By Policies

Figure 4.3: Population Initialization Strategy Influence

with the number of alternative routes set to 4 and 5. It is noticeable that the results are the same when 4 or 5 alternative routes are used.

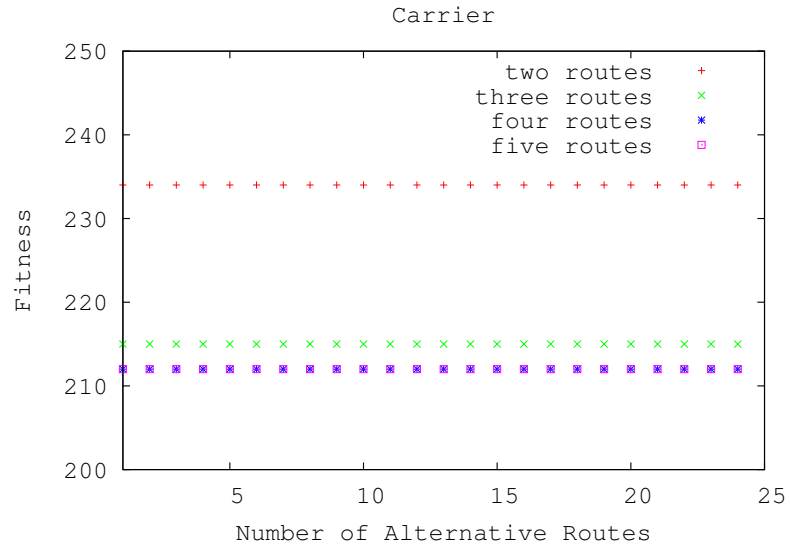


Figure 4.4: Number of Alternative Routes Influence

Hence, we conclude that when the network load increases, the use of alternative routes is crucial to obtain good solutions. The number of alternative routes have to be defined experimentally cause it depends on traffic and network topology.

Population Size

The population size does not influence the GA when it is using the policies, since as long as it has the 6 best fitness values, it can converge to a good solution.

The population size does influence the GA when its population is initialized randomly. This is expected since the more different solutions the GA has, the easier it is to find good solutions. The results for the tests executed fixing the scenario to **30** LSP requests and **10** iterations are shown in Table 4.5.

It can be noticed that the population size and the number of iterations are related. When the population size increases, the number of different solutions also increases and it is necessary to have a number of iterations sufficient to search the pool of options, otherwise the gains of increasing the population size are not obvious, as can be shown for the population size equal

Table 4.5: GA Results as a Function of Population Size and Number of Routes

Population Size	Number of Routes			
	2	3	4	5
5	83	89	80	91
10	82	91	95	103
15	77	94	81	100
20	79	99	89	112

to 10 and number of routes equal to 4. There are more options to search than the 10 allowed iterations.

Number of Iterations

The number of iterations is less that 5 when the GA uses the policies. When the population is initialized randomly, the results show that increasing the number of iterations improves the result. But the number of iterations necessary to give a good solution must be defined experimentally.

The results for the tests executed, fixing the scenario to **30** LSP requests and population of size **10** are shown in Table 4.6. In this example, the optimum was reached with 25 iterations and number of routes equal to 2 or 3. When the number of routes increases, the number of iterations must increase too, since the space of solutions grown.

Table 4.6: GA Results as a Function of Number of Iterations and Number of Routes

Number of Iterations	Number of Routes			
	2	3	4	5
5	110.0	108.0	89.0	106.0
10	82.0	91.0	95.0	103.0
15	83.0	90.0	94.0	93.0
20	84.0	82.0	81.0	89.0
25	77.0	77.0	90.0	81.0

4.8.3 Comparison with the CPLEX Results

The solutions found by the best execution of the genetic algorithm are shown in Table 4.7.

Table 4.7: Number of Rejections and Number of Hops Using the Genetic Algorithm

Topology	Number of Flows									
	10		20		30		40		50	
	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops	Rej.	Hops
Mesh	0	20	1	48	11	148	20	240	30	340
Ring	0	29	0	68	4	139	15	248	24	340
NSF	0	30	0	65	2	118	11	211	20	307
Carrier	0	25	0	54	0	77	1	144	7	212
Dora	0	33	0	71	3	118	8	193	22	314
Sul	0	14	1	43	8	120	8	136	14	210

We notice that the genetic algorithm is able to find the best solution for the vast majority of cases and is very close to the optimal solution in the other cases. The execution time is on the magnitude of milliseconds for all instances executed. Each element of the Table 3.3 reports the average execution time of the algorithm over 5 run trials and its standard deviation (within brackets). The number of iterations necessary to reach the best solution of the GA is generally less than 5.

Table 4.8: GA Average Execution Time for Different Traffic and Topologies

Topology	Number of Flows				
	10	20	30	40	50
Mesh	334.81 (139.44)	401.89(137.5)	429.8 (124.14)	466.88 (125.17)	495.9 (135.02)
Ring	325.6 (80.37)	400.08 (88.4)	454.25 (87.86)	484.01 (94.27)	509.31 (98.64)
NSF	305.41 (77.01)	383.62 (86.35)	440.64 (85.09)	475.03 (90.66)	502.63 (95.99)
Carrier	366.11 (87.01)	445.41 (92.54)	491.35 (100.44)	552.88 (114.97)	589.91 (130.62)
Dora	321.46 (82.86)	392.1 (89.17)	446.22 (88.83)	482.69 (97.71)	522.97 (102.13)
Ring	317.06 (85.76)	381.9 (90.48)	431.95 (83.6)	472.77 (89.12)	508.61 (92.25)

The solutions found using CPLEX were compared with the solutions found by the best execution of the genetic algorithm and are shown in Figure 4.5. The graph shows CPLEX results for the different models described in Chapter 3:

- MinRej - model that takes into account the number of rejections but does not try to balance the load. It is a lower bound for our two-step model.
- MinRejBal - model that takes into account the number of rejections and tries to balance

the load. Gives the results for our two-step model.

- Genetic - heuristic that takes into account the number of rejections and tries to balance the load using policies.

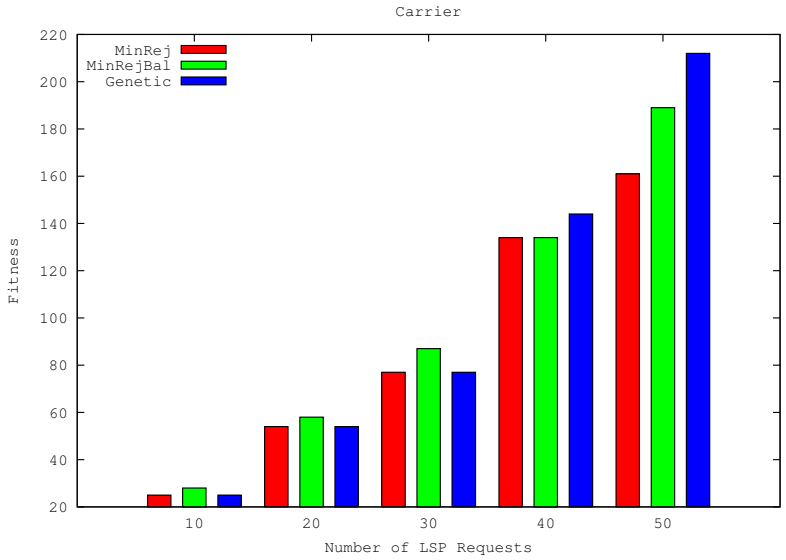


Figure 4.5: CPLEX and GA Results Comparison for the Carrier Network

For the scenarios with light traffic, the GA results are similar to the lower bound, where there is no need for load balancing. When the traffic load starts to increase, the GA solution is between the ones found when the traffic is balanced and the solutions found when it is not balanced.

When the network is overloaded, the GA results found are on average 20% worse than the best solution, but this is not very problematic, since our goal is to obtain good solutions when the load is within the network capacity and the network is still congested.

4.9 Concluding Remarks

The nature of the problem we are addressing prevents the use of exact methods to solve instances of moderate size, let alone instances of typical size in real-time. To cope with these difficulties, we propose the use of heuristics to solve the problem. Specifically, a genetic algorithm was implemented and compared to the optimal solution.

This Chapter validates the use of use of heuristics to obtain the solutions in a time period that is practical in real life. The execution of the two-step model proposed in Chapter 3 using CPLEX takes longer and it is more complicated since it has to occur in phases.

In this chapter, we present a genetic algorithm for LSP routing, which main goal is minimizing the network rejection rate. It is possible, in the population initialization phase, to reroute a LSP to avoid congested links even choosing longer paths, in order to balance the overall link loads and to allow a better use of the networks resources.

The fact that traffic changes, new LSP requests arrive, and node/link failures happen show however that over time the routing decisions taken now will cause some inefficiency in the future and occasional rerouting may be needed.

This fact encouraged the study of the *on-line* version of the problem of alternative path selection in MPLS networks. Another interesting problem to investigate is the detection of the moment when a rerouting is needed. Hence, the next steps in our work are: on-line routing and traffic change/rerouting needs detection.

Chapter 5

Using the Genetic Algorithm On-Line

There are two issues that must be addressed when dealing with the design of a routing mechanism in an IP network: (1) lack of load balancing and (2) the highly dynamic nature of traffic demands. Chapters 3 and 4 discussed offline solutions for the lack of load balance in IP networks. On that moment, our solutions profited from the prior knowledge of the LSP requests. We assumed that all LSPs to be routed and their resource requests were known at the time the routing was done and the determination of the optimal paths for the LSPs required the solution of an optimization problem. One of our contributions was the implementation of a genetic algorithm that achieves near optimum performance in a reasonable time period.

In practice however it is likely that new requests need to be setup, new network elements need to be configured or links and/or nodes face failures and/or recoveries. Consequently, routing mechanisms should try to be very adaptive using an on-line approach.

In this chapter, we address the problem of on-line LSP routing; the main goal still being the minimization of the network rejection rate and number of hops. We will show how the genetic algorithm proposed for offline routing can be used on-line.

5.1 On-line Routing Scheme

Based on the previously conducted experiments, we noticed that our evolutionary routing scheme, the GA with the adaptive movements and policies described in Chapter 4 (Adaptive Routing Scheme), could be used to compute an on-line solution for the LSP routing problem.

The main difference between the use of the GA offline and its use on-line is the fact that while the adaptive movements in the offline approach do not have any operational consequence, they can hurt the network performance when executed on-line due to the intrinsic cost of LSPs rerouting.

Besides most part of the on-line algorithms use one policy only (OSPF, WSP) and our GA will make a crossover of various policies. In a first step, the solutions will be computed by each policy simulating the population inititalization phase online. The second step will be the crossover of the solutions.

The proposed on-line path selection scheme works as follows: when a request arrives at the network, it tries to find a route to allocate to it. When trying to find a route for a request, the policies described in Chapter 4 (MH, AMH, LU, ALU, B and AB) will be used.

Then one of three situations can occur for each of the policies individually:

1. a route with maximum D hops difference from the shortest route is found,

2. a very long route is found,

3. there is no route available.

If no route is found, meaning that the request will be rejected, or if a very long route is found, meaning that the delay for that request will be big, the algorithm tries to reallocate some LSP. It will try to find a LSP that can be rerouted to leave space for the new request that just arrived.

To choose a LSP to be rerouted, all policies use the adaptive movements algorithm that searches for the most utilized link in the set of alternative routes of the request that just arrived. Then it checks for each LSP that crosses this link to see if this LSP has a demand greater than the demand of the new request and if there is an alternative route for this LSP.

Our approach tries to reroute just one LSP, since rerouting of a large number of LSPs is operationally undesirable. The algorithm stops when it finds one LSP that satisfies the constraints or when no LSP is found.

This strategy is similar to the adaptive movements described in Chapter 4. It is worth mentioning that the adaptive movements before had no cost since we were working offline. Now, the trade-off between rerouting and/or rejecting some LSPs has to be considered.

Each policy will therefore propose one configuration which will be used by the genetic algorithm for crossover. Each policy will have its cost based on number of rejections and number of hops and also the number of reroutes necessary to be applied. The GA rerouting cost will be the rerouting cost of the chosen option for that LSP.

5.2 Experiments

Our first experiments were created to check the influence of each policy on the GA solution. We wanted to know if only using one of the policies on-line would be better than using the GA.

The second set of experiments were created to check the performance of the GA when LSP setup needed to be more dynamic - short-lived LSPs and in case of link failures. These experiments were used in [Boutaba et al., 2002] and [Kar et al., 2000] to prove the advantages of their algorithm and is reproduced here with the same motivation.

The topologies used in the experiments are the same topologies used throughout this work, see Figure 5.1.

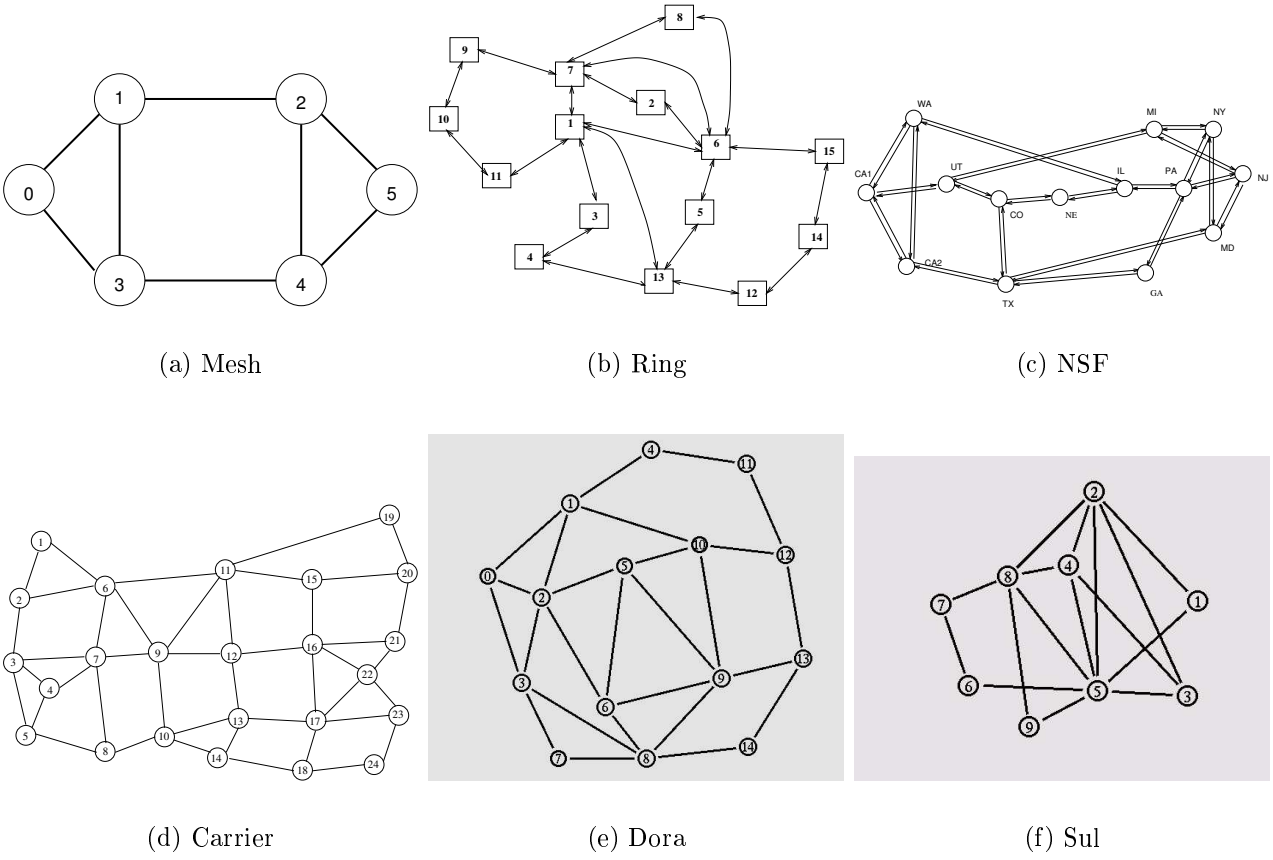
5.2.1 Policies Configuration - Number of Alternative Routes

In this experiment, we show the influence of the number of alternative routes on the final cost obtained by each policy, i.e., each policy is executed with a different number of alternative routes. The results for *Carrier* topology are shown in Figure 5.2. The results for the other topologies can be seen in the Appendix.

A total of 50 static path setup requests are sent to the network. Static paths resemble long-lived tunnels in MPLS networks and once established stay in the network until the end of the simulation.

The optimal result obtained using the ILP model described in Chapter 3 is compared to the result obtained using the policies described for the initialization phase of the population for the GA described in Chapter 4.

The two best number of alternative routes for most of the policies is four or five. The only exception is for Adaptive Balance policy that has the best performance when the number of



(a) Mesh (b) Ring (c) NSF

(d) Carrier (e) Dora (f) Sul

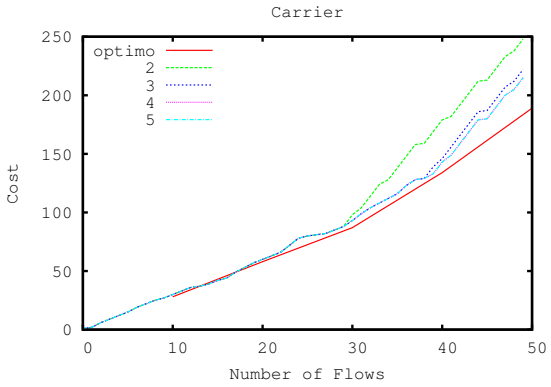
Figure 5.1: Topologies Used in the Experiments

alternative routes is two for the topologies Dora, NSF and Carrier. This can be explained by the fact that these topologies are less loaded than the other ones and the traffic is already balanced, without *hot spots*. Therefore they do not need to used the alternative routes so much.

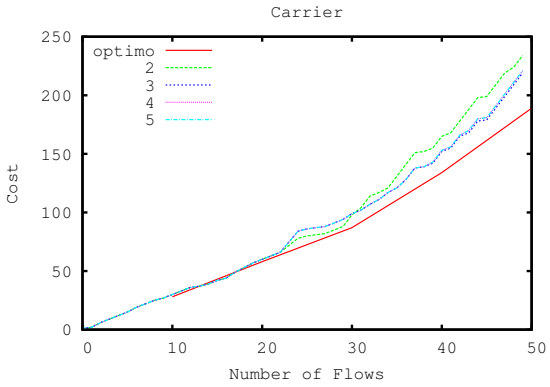
5.2.2 Policies Comparison - Cost

Once we find the best configuration for each policy, we compare the policies considering the total cost, which is equal to the objective function used for the ILP model and equal to the fitness function used for the GA. The results are shown in Figure 5.3.

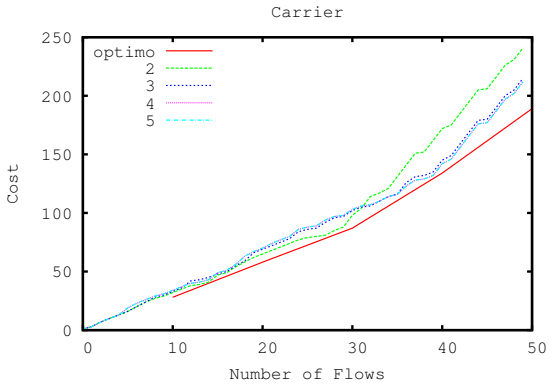
It is noticeable that the policy LU obtains the worst results. It is not clear from the results that one policy is much better than the others. The best polices are B, MH, AMH and ALU with very similar results when the network is lightly loaded. The policy AB performance



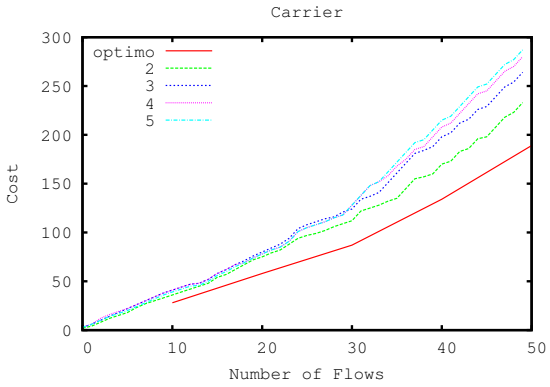
(a) Min-Hop Policy



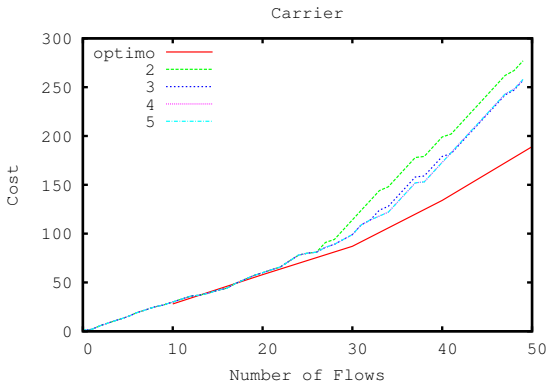
(b) Adaptive Min-Hop Policy



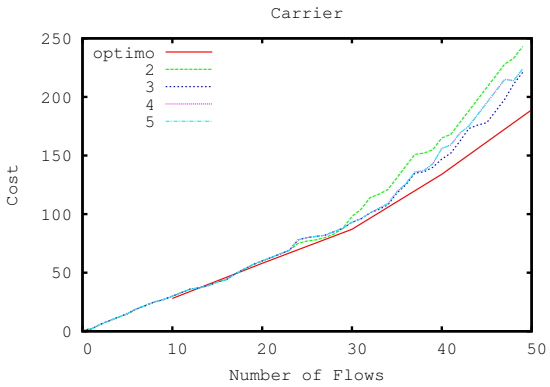
(c) Balance Policy



(d) Adaptive Balance Policy



(e) Limited Utilization Policy



(f) Adaptive Limited Utilization Policy

Figure 5.2: Influence of Number of Alternative Routes on the Policies Cost

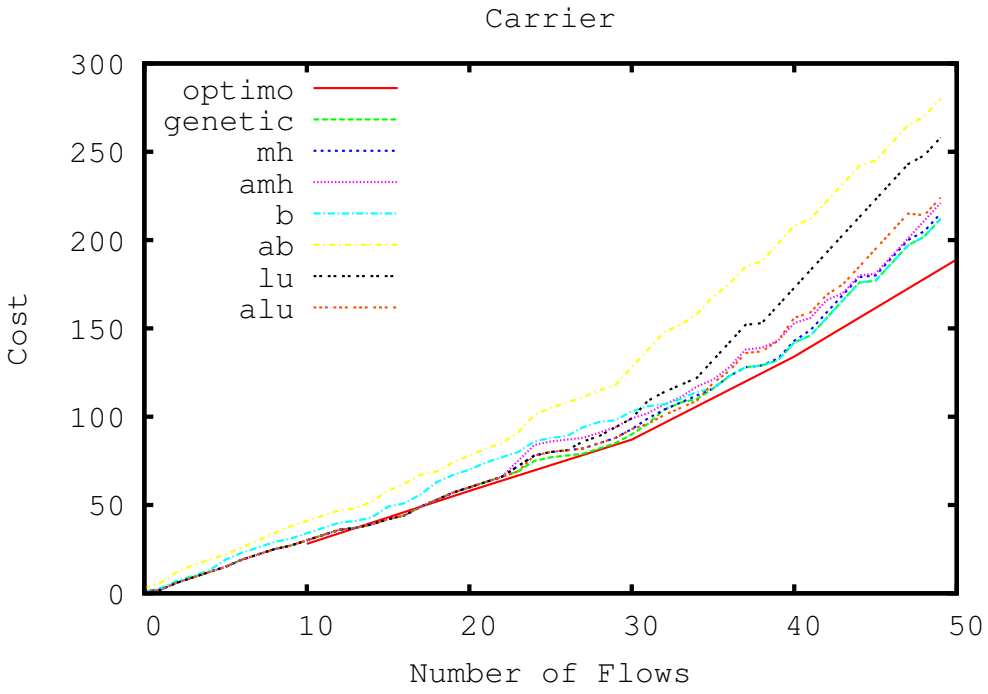


Figure 5.3: Cost for Each Policy as a Function of the Number of Flows For Carrier Network

varies a lot and is the best one for some topologies and the worst for other topologies.

This experiment showed that we could not use only one policy in our on-line solution. The combination of the policies solutions due to the crossover executed by the genetic algorithm gave the best results in overall. The genetic solution is generally equal to the best solution found by any of the policies.

5.2.3 Policies Comparison - Rerouting

Figure 5.4 presents the number of adaptive movements executed by each policy as the number of flows increases. The policies MH, B and LU do not execute adaptive movements. The policies AMH, AB and ALU use the adaptive movements to reroute some LSPs trying to minimize the rejection rate.

For lightly loaded scenarios, like Carrier and Dora, the AB policy executes more movements, since it tries to keep the traffic balanced at all times. The policies AMH and ALU start the movements when the network starts facing congestion. This is a trade-off to keep in mind, since the performance of ALU is not far from the best solution and it does fewer reroutings

for lightly loaded networks.

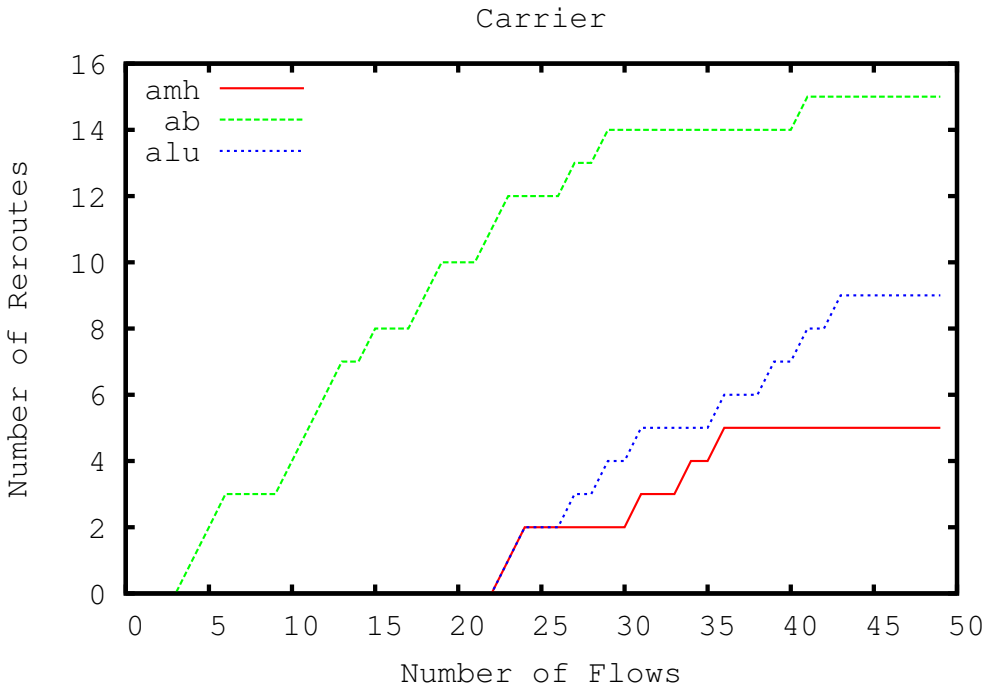


Figure 5.4: Reroutes over Number of Flows For Carrier Network

5.2.4 Setting Dynamic Paths

In this experiment a total of 1000 setup requests are sent to the network. The experiment's goal is to verify the influence of the LSP's holding time on the policies behavior. The holding time of the LSPs ranged from 25 time-units to 300 time-units. The cost as a function of rejections and hops was the metric used to compare the policies and the genetic algorithm. This experiment is similar to the experiments described in [Boutaba et al., 2002] as well as the experiment setup: number of request generated and holding time range.

It is noticeable from Figure 5.5 that the genetic algorithm has the performance very similar to the best policy. Also, the policy that gets the best performance varies from one scenario to the other.

The AB policy has very good results in the most part of the experiments. The other policies have similar results when the network is less congested: in the Dora and Carrier topologies for example.

This experiment shows the advantage of using the genetic algorithm instead of one single policy. The GA is flexible and has the result of the best policy in most part of the time.

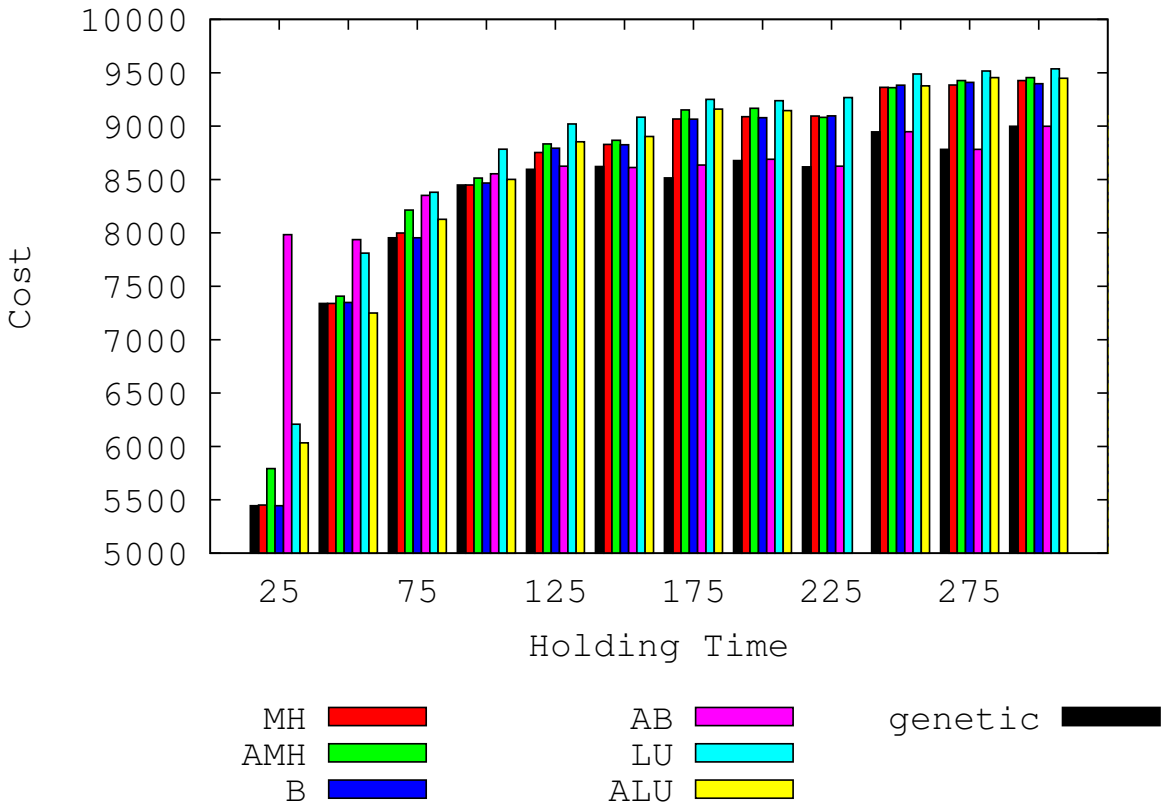


Figure 5.5: Cost as a Function of LSP Holding Time For Carrier Network

5.2.5 Link Failure

It is expected that routing decisions taken on-line cause inefficiency in the network over time and, for this reason, occasional network grooming will be necessary. In this context, network grooming means LSP rerouting in an attempt to keep network resources optimally allocated.

Dynamic traffic changes or topology changes demand a reevaluation or grooming of the previous routing solution. *Grooming* of LSPs can happen:

1. when a link or a node recovers from a failure,

2. when a new network element enter in the network,

3. when a LSP is torn down.

To check the performance of the policies in case of a link failure, the following experiment was designed: in each trial of this experiment, one link is intentionally taken down from the network and the number of LSPs requiring rerouting and the percentage of successful reroutes are recorded. The experiment is repeated for all links.

This experiment is similar to the experiments described in [Kar et al., 2000] and [Boutaba et al., 2002]. Several points of link failure are defined for the topologies studied. The results are shown in Figures 5.6 and 5.7.

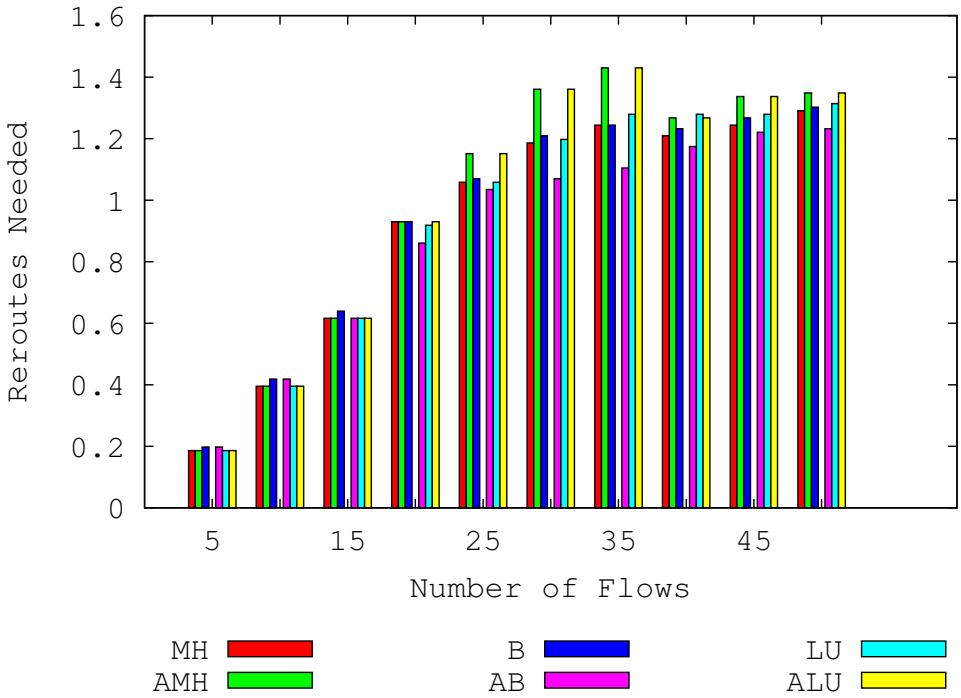


Figure 5.6: Number of Reroutes Needed for Each Policy in Case of Link Failure For Carrier Network

It is noticeable from Figure 5.6 that the policy AB is the best for less loaded networks, but it increases the number of reroutings rapidly when the load increases. When the network becomes congested, the policy LU is the policy that requires the less number of reroutings. For the most part of topologies, the number of reroutings needed is less than 2. It can be noticed that the difference among the policies is small when the network is not congested.

We can notice that the policy LU is very successful in finding new routes for the LSPs since it does not have much to reroute. Figure 5.7 shows that for all policies the success rate in finding new routes for LSPs after a link failure is more than 50%. The policy LU is the

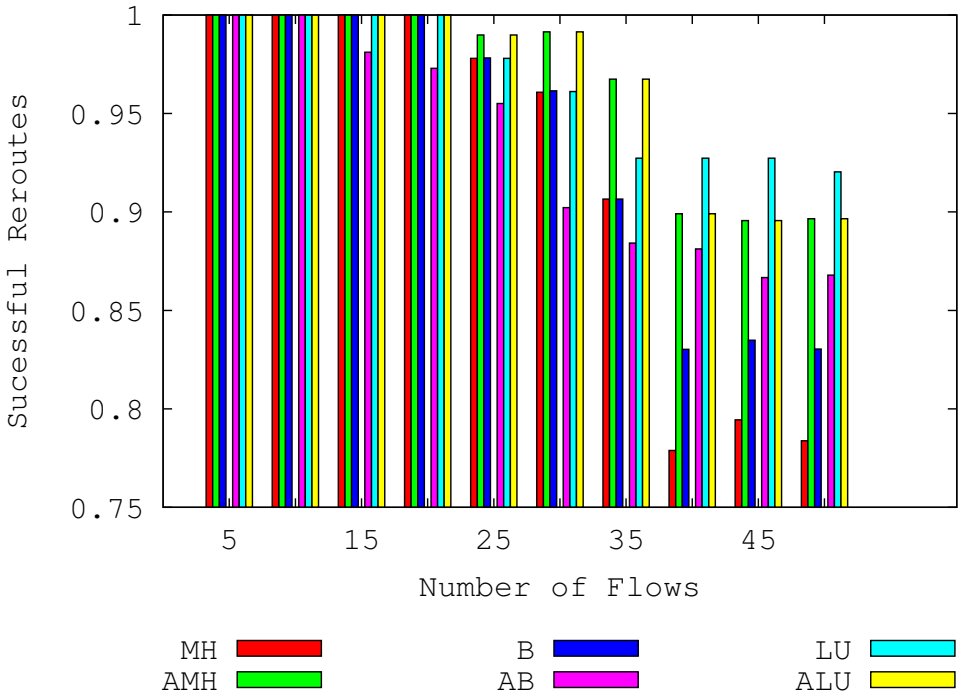


Figure 5.7: % of Successful Rerouting in Case of Link Failure For Carrier Network

best since it leaves more space for future requests. The policy AB has already balanced the traffic and has fewer options in case of link failures.

5.3 Concluding Remarks

Routing mechanisms for MPLS networks carry out critical functions for the performance of these networks. They take on the responsibility for resource management and load balancing. This chapter addresses the problem of adaptive routing in MPLS networks.

It was observed that with the proposed scheme we have obtained substantial reduction in the number of rejected requests in comparison with the basic strategy used nowadays, the default minimum number of hops scheme.

Although the results are encouraging, there are several aspects of the proposal for which additional work is needed. Link failures and LSP request arrivals are straight forward to detect. Traffic changes due to new network elements becoming available or ending of a request are more complex.

A trigger mechanism for example will be necessary to define the moment when grooming needs to occur. This trigger mechanism should detect traffic changes and start the grooming process. In the next chapter, the combination of rerouting mechanisms and monitoring tools will be studied.

Chapter 6

Traffic Monitoring and Network Grooming

Since some on-line routing decisions can lead to non-optimal solutions and since traffic can change over time, the routes should be constantly monitored and in some cases even *groomed*. In this chapter we propose a traffic change detection mechanism to be used with the grooming scheme.

To cope with the traffic change detection problem, control charts are applied. Experiments are run with two types of control charts: EWMA and XBar-charts and they show that is feasible to use control charts for traffic change detection.

6.1 Motivation

One of the modern trends of Internet design is that lower level transport protocols should be network conscious and adaptive. They should monitor network conditions and answer appropriately to changes in these conditions, which requires using network information from the recent past to guide future behavior. There is much interest in using network measurements for both modeling and operational purposes.

In this chapter we proposed a mechanism to deal with the dynamic nature of Internet traffic based on traffic monitoring and control charts. Since it is not scalable to distribute every change in the link states the proposed scheme will try to figure out *if* and *why* the

traffic changed: if the change in traffic is due to common variation or if it is due to unexpected facts, such as a link or a router failure. With this approach, a link state update will be triggered only when a subtle traffic change occurs. This approach reduces the volume of updates of traffic information on the network.

This traffic change detector will be used to trigger the load balancing mechanisms. It will check traffic tendency based on XBar-charts or EWMA-charts [Montgomery, 1990]. XBar-charts and EWMA-charts are control charts and their goal is to detect if a process is under control, which means that the process is presenting the expected metrics. In our work, we will monitor the utilization of a link and check if it is viewing the expected load.

6.2 Traffic Monitoring

A network can be monitored either in an active or passive manner. The first manner gives a measure of the performance of the network whereas the latter gives a measure of the workload on the network. Both have their merits and should be regarded as complimentary.

The active approach relies on the capability to inject packets into the network and then measure the services obtained from it. It introduces extra traffic into the network, but it has the advantage of measuring the desired quantities at the desired time.

Passive measurements are carried out by observing normal network traffic, without the extra load. The passive approach measures the real traffic, but the amount of data accumulated can be substantial because the network will be polled often for information.

There are various quantities of interest that can be insightful about the state of the network. Available bandwidth together with latency, loss, etc., can predict the performance of the network. Based on the available bandwidth , the operator can obtain information about the congestion, decide the admission control, perform routing, etc. For MPLS networks, the available bandwidth information can be used to decide LSP setups, routing and preemption. Therefore, obtaining an accurate measurement of the available bandwidth can be crucial to effective deployment of QoS services in a network.

In this work, the passive approach is used. The link's available bandwidth is monitored to check for rerouting needs.

6.3 SPC - Statistical Process Control

6.3.1 XBar-charts

Shewhart's statistical process control [Hansen, 1963] is a methodology for charting processes in XBar-charts and quickly determining when the process is out of control. The XBar will actually the mean of the process being measured ($\overline{X}$).

A series of rules exist to detect conditions in which the process is behaving abnormally to the extent that an out of control condition is declared. In this work, we used the extreme point condition test. Therefore, the process is out of control where a point is either above the upper limit or below the lower limit of the chart.

In our work, if the bandwidth is above the upper limit or below the lower limit it means that a link is broken or a router is not working.

The XBar-chart is computed from M data points, which are ordered in time. These points are in fact the mean of N data points measured, $\overline{X}_1, \cdots, \overline{X}_M$. Our M data points will be the mean of N measurements of a link bandwidth; what can be easily obtained via SNMP.

In order to compute the chart's control limits it is necessary to compute the population mean, $\overline{\overline{X}}$, and the mean range, $\overline{R}$.

The range of an interval, for instance R_1, is the difference between the largest and the smallest value at this interval.

$$\overline{\overline{X}} = \frac{\overline{X}_1 + \overline{X}_2 + ... + \overline{X}_M}{M} \tag{6.1}$$

$$\overline{R} = \frac{R_1 + R_2 + ... + R_M}{M} \tag{6.2}$$

The chart limits are calculated as follows:

$$Lower Limit(LL) = \overline{\overline{X}} - A_2(n)\overline{R} \tag{6.3}$$

$$Upper Limit(UL) = \overline{\overline{X}} + A_2(n)\overline{R} \tag{6.4}$$

where A_2 values depend on N, as shown in Table 6.1.

Table 6.1: Examples of some A_2 values

N	2	3	4	5	6	7	8	9
A2	1.880	1.023	0.729	0.577	0.483	0.419	0.373	0.337

These limits actually represent a distance bellow or above the mean by 3 times the standard deviation. In this work, the control charts, XBar-chart and EWMA-chart, will be used to check the average bandwidth utilization for a link.

6.3.2 EWMA-charts

An Exponentially Weighted Moving Average chart, EWMA-chart, is used when it is desirable to detect out-of-control situations very quickly [Montgomery, 1990].

It has a built in mechanism for incorporating information from all previous data and weighting more the current information through λ, as can be noticed in Equation 6.6. The equations 6.5 - 6.10 are used to compute the EWMA-chart.

$$\hat{X}(0) = \frac{\overline{X}_1 + \overline{X}_2 + ... + \overline{X}_M}{M} \tag{6.5}$$

$$\hat{X}(i) = \lambda \overline{X}_i + (1 - \lambda)\hat{X}(i-1) \tag{6.6}$$

The upper (UL_i) and lower (LL_i) limits are calculated as follows :

$$Sigma = \frac{\overline{R}}{D2(N)} \tag{6.7}$$

$$F_i = \sqrt{\frac{(\lambda}{1-\lambda)}[1 - (1-\lambda)^{2i}]} \tag{6.8}$$

$$LowerLimit(LL_i) = \hat{X}(0) - \frac{3 * Sigma}{\sqrt{N}} F_i \tag{6.9}$$

$$UpperLimit(UL_i) = \hat{X}(0) + \frac{3 * Sigma}{\sqrt{N}} F_i \tag{6.10}$$

where D_2 values depend on N, as shown in Table 6.2.

Table 6.2: Some D_2 values

N	2	3	4	5	6	7	8	9
D2	1.128	1.693	2.059	2.326	2.534	2.704	2.847	2.970

A tutorial in control charts can be found in [Sytsma and Manley, 1999] as well as a more detailed information about the formulas and tables.

6.4 Control Charts Experimental Results

In this work, to test the use of control charts for traffic change detection, synthetic data is generated, based on the exponential distribution function. Most studies in network simulation use a Poisson distribution to model the arrival of request and an exponential distribution to model the request holding time [Boutaba et al., 2002, Dias et al., 2003, Salvadori and Battiti, 2003].

Two traffic scenarios are synthetically generated. The list bellow describes the scenarios and the purpose of the test.

- First Scenario: Increasing Traffic - in the beginning the traffic follows a exponential distribution with mean 10 Mbps. Then four spaced in time peaks are generated increasing the mean in 15%.
- Second Scenario: Decreasing Traffic - in the beginning the traffic follows a distribution with mean 10 Mbps. Then four spaced in time peaks are generated decreasing the mean in 15%.

Our first test will try to verify what is the best value of the N and λ for the data generated. It is worth noticing that the test scenarios were generated synthetically and the best configuration will be the one that can detect the 4 peaks generated.

6.4.1 Detecting Utilization Mean Changes Through XBar-charts

It can be seen in Figure 6.1 the influence of the sample size, N, for the XBar-chart. We use 2000 points, M is set to 20 and N is set to: 3, 6 and 9. The number of points in the X axis

in each graph can be calculated as $2000/20 * N$.

All charts detect an increase in the mean but the chart with $N = 3$ detect precisely the four peaks. The other charts present the four peaks within the limits.

6.4.2 Detecting Utilization Mean Changes Through EWMA-charts

Several experiments are run using different values for λ in EWMA-charts: 0.1, 0.2, 0.3, 0.5 and 0.7. Bigger λ value means that more importance is given to the current data.

It can be seen in Figure 6.2 that as λ increases the limits of the EWMA-chart are more distant. For small λ values, the traffic changes are detected, since the EWMA-chart is expected to be more conservative.

6.4.3 XBar-charts x EWMA-charts

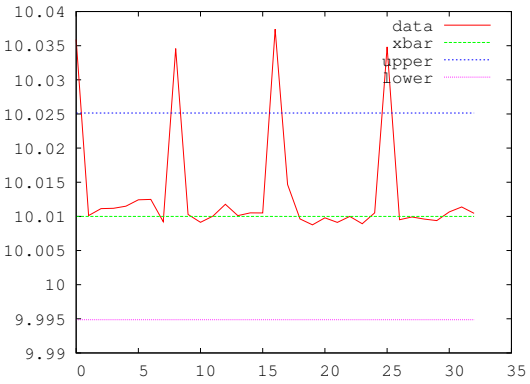
If we compare XBar-charts and EWMA-charts, it is noticeable in Figure 6.3 that the EWMA-chart reacts faster, adjusting its limits according to the data.

The experiments are run with M set to 20, N set to 3 for the XBar-chart and λ set to 0.3 for the EWMA-chart. Figure 6.3 shows the best results of the two charts for the two scenarios: increasing and decreasing traffic.

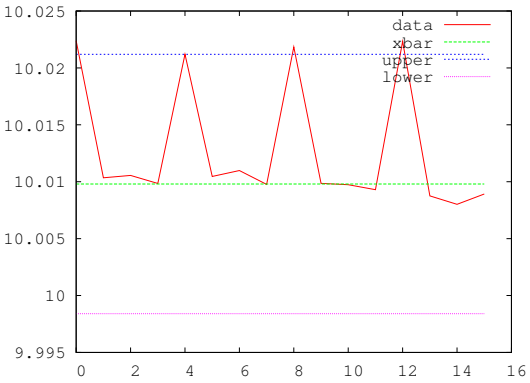
The results for both control charts were very similar. It is interesting to keep in mind some advantages and/or disadvantages of each one. The main advantages of EWMA are: (1) they detect out-of-control conditions more quickly, (2) detection can be done by using only one rule - the point has to be inside the 3 standard deviation limits to be considered in control. The EWMA worst disadvantage is that it is more difficult to construct. Its value at time t is computed recursively from the past points.

Despite being easier to calculate, XBar-charts have more rules to test in order to detect abnormal situations than EWMA-charts [Hansen, 1963].

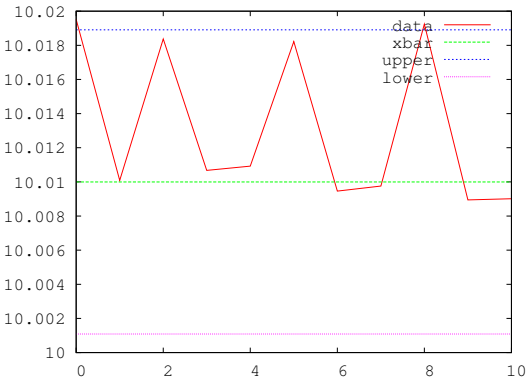
In this work, we used XBar-charts cause they are easier to compute and the rule used in the experiments to detect out-of-control situations was the simple rule: extreme point condition.



(a) n=3

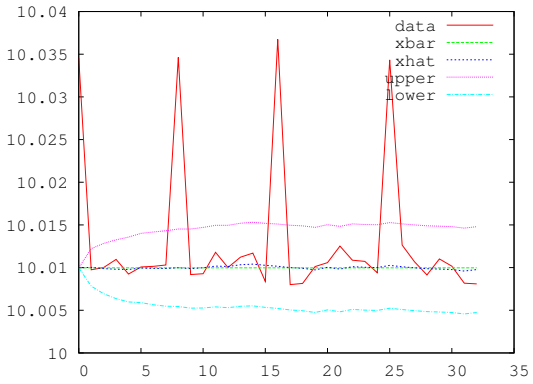


(b) n=6

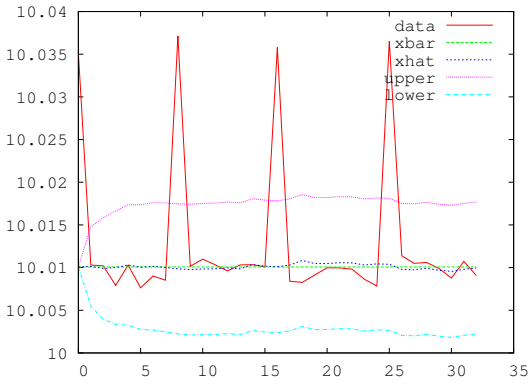


(c) n=9

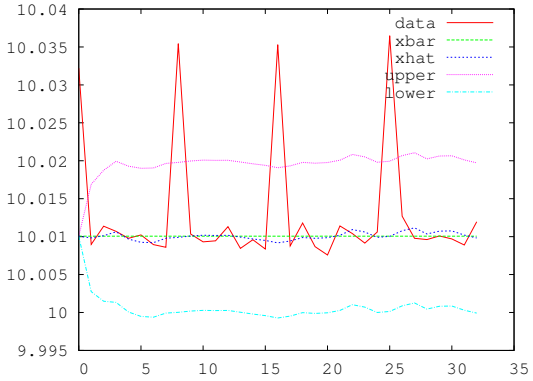
Figure 6.1: Influence of Sample Size in XBar-charts



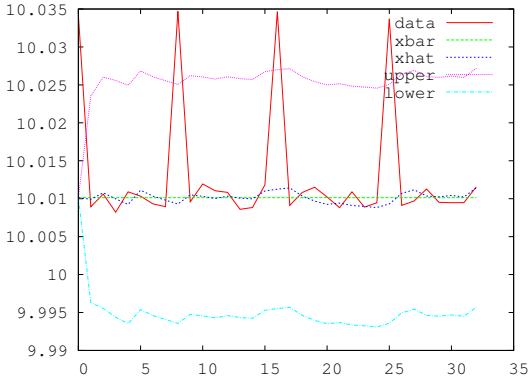
(a) λ=0.1



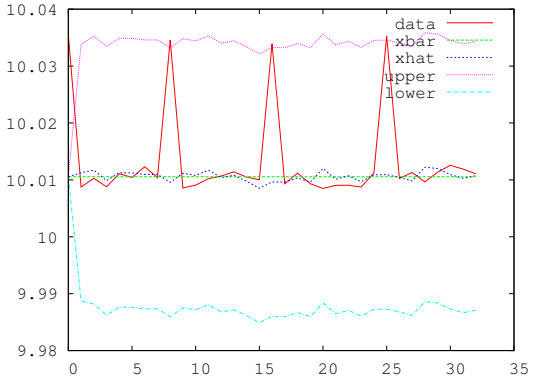
(b) λ=0.2



(c) λ=0.3

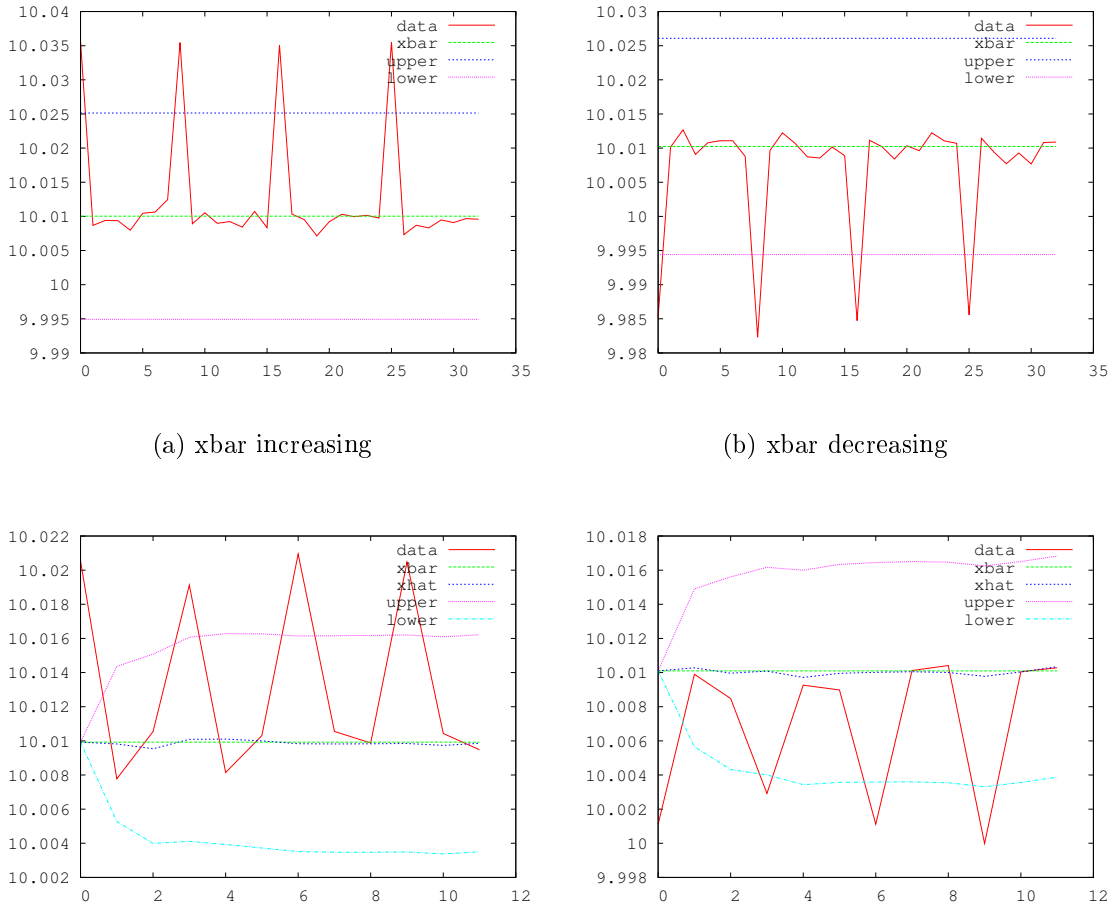


(d) λ=0.5



(e) λ=0.7

Figure 6.2: Influence of λ in EWMA-charts



(a) xbar increasing

(b) xbar decreasing

(c) ewma increasing

(d) ewma decreasing

Figure 6.3: XBar and EWMA Results for Both Test Scenarios

6.5 Traffic Changes Detection Scheme

After discussing how control charts work, we propose a grooming scheme for MPLS networks based on a traffic change detection mechanism.

Our previously described adaptive routing scheme or adaptive movements scheme will be combined with the control charts in the following way: each ingress node in the network monitors its links verifying if the link utilization is under control.

If a link/node failure occurs or if a LSP is torn down, probably the link's utilization will be affected. Thus, if an *out-of-control* situation is detected, a rerouting offline mechanism is triggered.

Algorithm 5 gives the proposed scheme pseudo-code.

Algorithm 5 Grooming/Monitoring Pseudo Code

```
1: calculateOptimalRouting();
2: for all Links do
3:    calculateControlChart();
4: end for
5: loop
6:    if monitorLink detects traffic has changed then
7:       updateControlChart();
8:       getUpdatedTopology();
9:       recomputeRouting();
10:   end if
11: end loop
```

The first step of the proposed scheme is the optimal routing computation. If nothing abnormal happens, the optimal routing will be used in the network without changes. The control charts are also computed and they will be used to monitor each ingress router. If a traffic change is detected, the control chart is updated and the grooming process is trigged. The grooming process consists of rerouting of LSPs in order to increase the network performance.

6.6 Experimental Results

An experiment is conducted showing how the grooming process can help the achievement of better network performance. The experiment works in the following way: in the first step a series of requests are optimally routed. Then some requests are finished, i.e. the bandwidth reserved for these requests becomes available again, opening space for reconfiguration of the already installed LSPs.

In the experiments, we defined the LSPs to be finished using an uniform distribution. The goal of the experiment is to check if the bandwidth decrease in the links will be detected and if some rerouting will happen improving the network delay time (less hops used to route some LSPs). In the first set of experiments, after the decrease in utilization is detected nothing is done. In the second set of experiments, after the decrease in utilization is detected, the grooming will happen.

The metric used to compare the benefits of grooming are residual bandwidth and number

of hops (cost) and the results are presented for the policies that are adaptive in Figures 6.4, 6.5 and 6.6 for the Carrier network. The network residual bandwidth is computing by adding the residual bandwidth of all links.

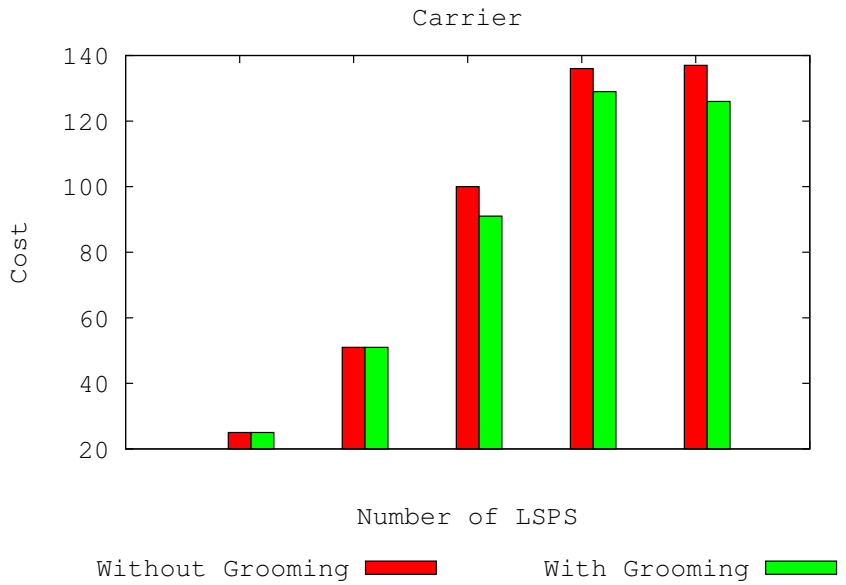
When we vary the policy, we vary the network status in terms of bandwith utilization in each link. For all policies and topologies, the cost decreased and the residual bandwidth increased when the grooming process was used in comparison with the case when no grooming was done, specially in the cases where the network was more congested, i.e. number of flows greater than 30.

6.7 Concluding Remarks

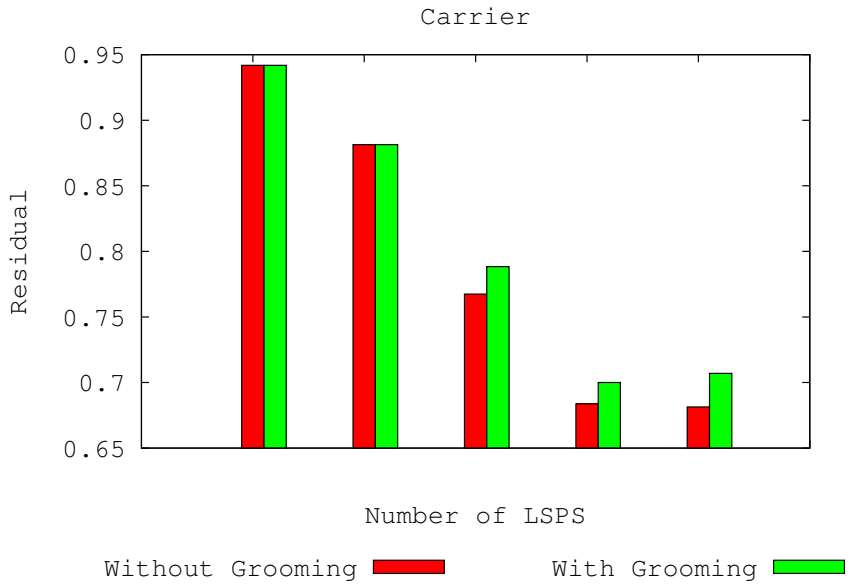
In this chapter we show that control charts could be used to monitor network conditions and allow the routing protocol to respond appropriately to them.

Two different types of control charts are evaluated and several experiments are run on synthetic data to show the main advantages of each type of control chart. EWMA-charts should be used when it is necessary to quickly react to small changes. XBar-charts should be used when major changes should be detected. Its main advantage is its simplicity and low overhead.

A grooming scheme is proposed and associated with control charts. Experiments show that better network performance can be achieved with the use of these tools.

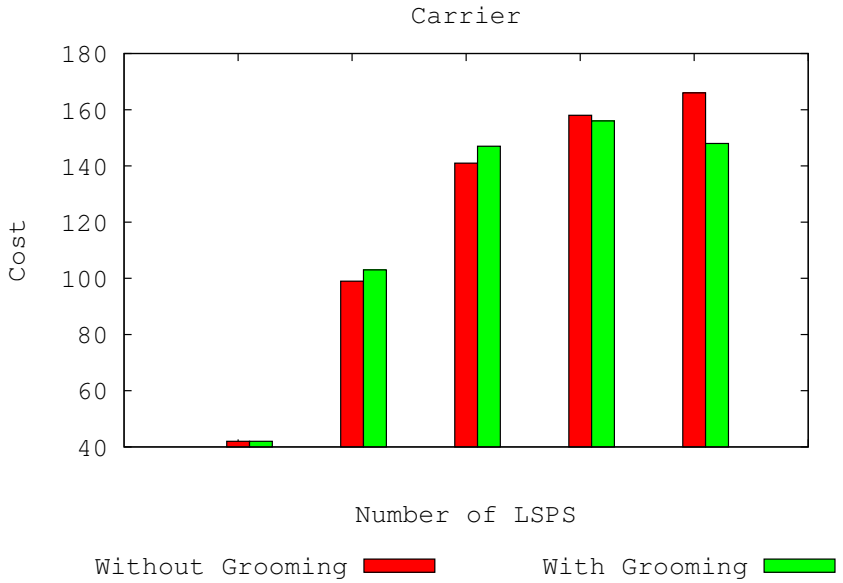


(a) Cost

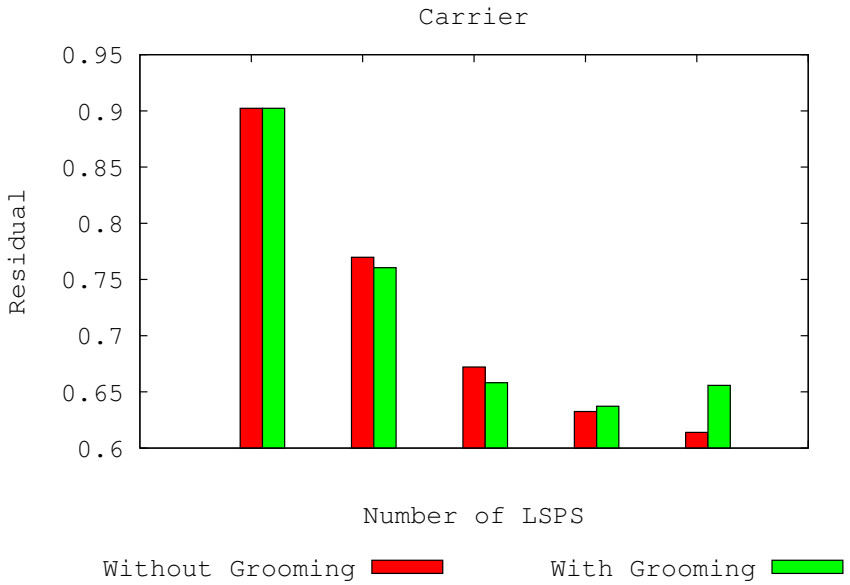


(b) % of Residual Bandwidth

Figure 6.4: AMH Grooming Results For Carrier Network

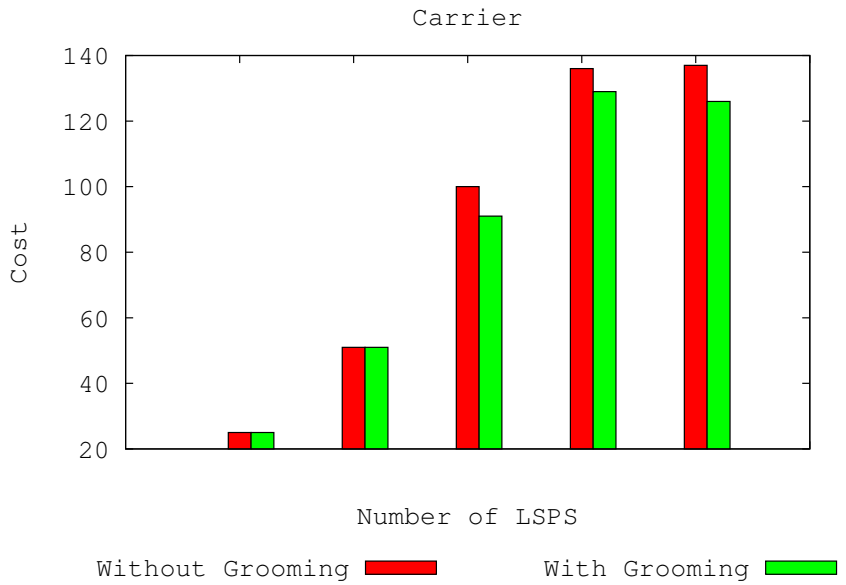


(a) Cost

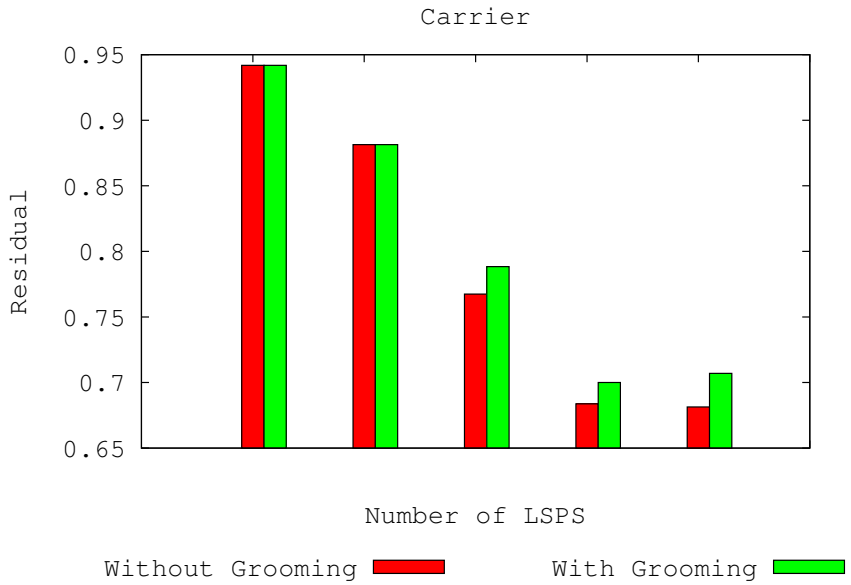


(b) % of Residual Bandwidth

Figure 6.5: AB Grooming Results for Carrier Network



(a) Cost



(b) % of Residual Bandwidth

Figure 6.6: ALU Grooming Results for Carrier Network

Chapter 7

Conclusions

In this chapter we present a summary of this thesis, highlighting its main contributions. We also present future directions for this work.

7.1 Summary of Accomplished Work

In this thesis, the problem of offline and on-line routing of LSPs in MPLS networks has been addressed. QoS in MPLS networks can be improved with efficient ways of computing explicit routes for LSPs.

This work main contributions to the state-of-the-art are:

- We proposed an ILP model that solves the offline problem of routing LSPs in a MPLS network in an exact manner. This model tries to balance the network load while minimizing the network rejection rate.

- We proposed a genetic algorithm that solves the offline problem of routing LSPs in a MPLS network in an heuristic manner. This algorithm is based on various policies, which route the LSPs using different criteria, and based on the combination of these policies with adaptive movements. The adaptive movements are responsible for network grooming in cases where the policy is not flexible enough. The execution time of the GA is much better than the execution time of CPLEX. The use of explicit routes computed by our approach is also better than the use of MPLS default routing scheme.

- We proposed and verified the use of the genetic algorithm to solve the on-line problem of routing LSPs in a MPLS network. The genetic algorithm can automatically solve the problem of finding the best policy for different topologies and traffic demands.

- We proposed a network grooming technique and a traffic change detection mechanism. We showed how the grooming technique can be applied to rerouting in case of an increase in the availability of network resources. We also showed how the link utilization changes detection mechanism can be used to trigger the grooming process.

The ILP model was solved using CPLEX. Various issues concerning the execution time necessary to solve the model were noticed and the genetic algorithm was developed to address theses issues. While being very useful as a formal tool to define the problem, the ILP model resolution is very timing consuming, and the ILP model being classified as a NP-hard problem.

Simulations were conducted in order to check the performance of the genetic algorithm in different traffic scenarios, different topologies, different policies, with and without adaptive movements, and offline and on-line. The simulation results were compared to optimal results obtained through the ILP model. The obtained results with the genetic algorithm were very satisfactory showing that it is able to extract the best performance of various distinct policies for various situations.

We also investigated the possibility of using the genetic algorithm for network grooming, which can happen when a new network element is configured, LSPs are torn down or a network element recovers from failure. We presented a utilization detection mechanism based on control charts for triggering the grooming process.

The utilization changes detection mechanism was used in the context of defining the moment the grooming should take place. Simulation results indicate that we can achieve better performance using the proposed mechanism.

7.2 Future Work

There are several possibilities for research in order to extend the work presented in this thesis. The first one is to continue the research of genetic algorithms for on-line routing. As discussed in this thesis, we believe that genetic algorithms are well suited for on-line routing, due to

its nature of being adaptive. The genetic algorithm proposed could be extended to include other policies as well as other genetic operators. New policies can be added to the population initialization phase and new ideas for crossover can be devised. The genetic algorithm could also be modified to treat the routing problem with a multi-objective approach, which brings flexibility for network administrators.

Another important facet of the routing process that could be investigated is its temporal aspect. The granularity of routing decisions can vary in a thiner scale besides going from offline to on-line routing.

In this thesis, we showed that the use of grooming techniques associated with control charts can bring important improvements to network performance. We believe that it is very important to extend the monitoring tools capabilities since they trigger the grooming process. An important extension of this work would be the design of control charts with supplementary rules.

Still regarding the grooming process, an aspect that we have not addressed in this thesis and that should be investigated in future research is the choice of traffic prediction tools. Together with monitoring tools, prediction tools could lead the grooming process to a better result. In this work, we assume that the grooming process will happen after the detection of an out-of-control situation by the control charts. With the assistance of traffic prediction tools the grooming process would have a preventive function.

Another possibility is to analyze the influence of network topology in the policies proposed and in routing algorithms in general. It was noticed that networks with the same load have completely different behavior due to its topology and number of alternative routes.

As future work there is also the development of an usable and portable version of the simulator and the writing of its technical documentation. In its current stage, due to several modifications implemented during the thesis, the GA implementation readability and organization is somewhat compromised. This would allow other students from the lab to use the genetic algorithm implementation and collaborate on its development. The integration/mapping of the GA output into the NS simulator input could also be of great help.

Bibliography

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

[Anjali, 2002] Anjali, T. e. a. (2002). A New Path Selection Algorithm for MPLS Networks Based on Available Bandwidth Estimation. *Lecture Notes in Computer Science*, 2511:205–214.

[Apostolopoulos et al., 1999] Apostolopoulos, G., Williams, D., Kamat, S., Guerin, R., Orda, A., and Przygienda, T. (1999). RFC 2676: QoS Routing Mechanisms and OSPF Extensions.

[Aukia et al., 2000] Aukia, P., Kodialam, M., Koppol, P., Sarin, H., and Suter, B. (2000). RATES: A Server for MPLS Traffic Engineering. *IEEE Network Magazine*, 14(2):34–41.

[Awduche et al., 2002] Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. (2002). RFC 3272: Overview and Principles of Internet Traffic Engineering.

[Awduche et al., 1999] Awduche, D., Malcolm, J., Agogbua, J., O'Dell, M., and McManus, J. (1999). RFC 2702: Requirements for Traffic Engineering Over MPLS.

[Awduche, 1999] Awduche, D. O. (1999). MPLS and Traffic Engineering in IP Networks. *IEEE Communications Magazine*, 37(12):42–47.

[Awduche et al., 2001] Awduche, D. O., Berger, L., Gan, D., Li, T., Srinivasan, V., and Swallow, G. (2001). RFC 3209: RSVP-TE: extensions to RSVP for LSP tunnels.

[Awduche and Jabbari, 2002] Awduche, D. O. and Jabbari, B. (2002). Internet Traffic Engineering using Multiprotocol Label Switching (MPLS). *Computer Networks*, 40:111–129.

[Bertsekas and Gallager, 1992] Bertsekas, D. and Gallager, R. (1992). *Data Networks*. Prentice Hall.

[Black, 2001] Black, U. (2001). *MPLS and Label Switching Networks*. Prentice Hall.

[Blake et al., 1998] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. (1998). RFC 2475: An Architecture for Differentiated Services.

[Boutaba et al., 2002] Boutaba, R., Szeto, W., and Iraqi, Y. (2002). DORA: Efficient Routing for MPLS Traffic Engineering. *Journal of Network and Systems Management*, 10(3):309–325.

[Boyle et al., 2002] Boyle, J., Gill, V., Hannan, A., Cooper, D., Awduche, D., Christian, B., and Lai, W. (2002). RFC 3346: Applicability Statement for Traffic Engineering with MPLS.

[Boyle et al., 2000] Boyle, J., Hannan, A., Gill, V., Cooper, D., Awduche, D., Christian, B., and Lai, W. (2000). RFC 2998: A Framework for Integrated Services Operation over Diffserv Networks.

[Brunato et al., 2002] Brunato, M., Battiti, R., and Salvadori, E. (2002). Load Balancing in WDM Networks through Adaptive Routing Table Changes. *Lecture Notes in Computer Science*, 2345:289–301.

[Buriol, 2003] Buriol, L. S. (2003). *Roteamento do Tráfego na Internet: Algoritmos para Projeto e Operação de Redes com Protocolo OSPF*. PhD thesis, Unicamp.

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

[Capone et al., 2006] Capone, A., Fratta, L., and Martignon, F. (2006). Dynamic Online QoS Routing Schemes: Performance and Bounds. *Computer Networks*, 50(7):966–981.

[Chou, 2004] Chou, C. T. (2004). Traffic Engineering for MPLS-based Virtual Private Networks. *Computer Networks*, 44(3):319–333.

[Coelho, 1998] Coelho, C. A. C. (1998). An Updated Survey of GA-Based Multiobjective Optimization Techniques. *ACM Computing Surveys*, 32(2):109–143.

[Crawley et al., 1998] Crawley, E., Nair, F., Rajagopalan, B., and Sandick, H. (1998). RFC2386: A Framework for QoS Based Routing in the Internet.

[Davie and Rekhter, 2000] Davie, B. and Rekhter, Y. (2000). *MPLS: Technology and Applications*. Morgan Kaufmann.

[Dias and Camponogara, 2003] Dias, R. and Camponogara, e. a. (2003). Otimização Lagrangeana Em Engenharia de Tráfego para Redes IP sobre MPLS. *In Proceedings of the XXI Simpósio Brasileiro de Redes de Computadores*, 1:475–490.

[Dias et al., 2003] Dias, R. A., Camponogara, E., Farines, J.-M., Willrich, R., and Campestrini, A. (2003). Implementing Traffic Engineering in MPLS-Based IP Networks with Lagrangean Relaxation. *In Proceedings of IEEE ISCC*, 1:373–378.

[Ericsson et al., 2002] Ericsson, M., Resende, M. G. C., and Pardalos, P. M. (2002). A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *J. Comb. Optim.*, 6(3):299–333.

[Feldmann and Rexford, 2001] Feldmann, A. and Rexford, J. (2001). IP Network Configuration for Intra-domain Traffic Engineering. *IEEE Network Magazine*, 15(5):46–57.

[Figueiredo et al., 2004] Figueiredo, G. B., da Fonseca, N. L. S., and Monteiro, J. A. S. (2004). A Minimum Interference Routing Algorithm. *IEEE Communications Society*, 4:1942–1947.

[Fortz et al., 2002] Fortz, B., Rexford, J., and Thorup, M. (2002). Traffic Engineering With Traditional IP Routing Protocols. *IEEE Communications Magazine*, 40(10):118–124.

[Fortz and Thorup, 2000] Fortz, B. and Thorup, M. (2000). Internet Traffic Engineering by Optimizing OSPF Weights. *IEEE INFOCOM*, 2(4):519–528.

[Fourer, 2002] Fourer, R. (2002). IMA Tutorial: AMPL - Hands On Session.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. (1979). *Computers and Intractability*. Freeman.

[Girish et al., 2000] Girish, M., Zhou, B., and Hu, J.-Q. (2000). Formulation of the Traffic Engineering Problems in MPLS Based IP Networks. *In Proceedings of IEEE ISCC*, 1:214–219.

[Girish et al., 2003] Girish, M. K., Zhou, B., and Hu, J.-Q. (2003). An Algorithm for Rerouting in Traffic Engineering of MPLS Based IP Networks. *In Proceedings of IP Operations and Management - IPOM*, 1:115–118.

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

[Guerin et al., 1997] Guerin, R., Orda, A., and D., W. (1997). QoS Routing Mechanisms and OSPF Extensions. *IEEE Globecom*, 3:1903–1908.

[Hansen, 1963] Hansen, B. L. (1963). *Quality Control: Theory and Applications*. Prentice-Hall.

[Henig, 1985] Henig, M. I. (1985). The Shortest Path Problem with Two Objective Functions. *European Journal of Operational Research*, 15:281–291.

[Holland, 1973] Holland, J. (1973). Genetic Algorithms and the Optimal Allocations of Trials. *SIAM Journal of Computing*, 2(2):88 – 105.

[Hong et al., 2003] Hong, L., Dong, B., and Wei, D. (2003). An Explicit Routing Optimization Algorithm for Internet Traffic Engineering. *In Proceedings of the International Conference on Communication Technology - ICCT 2003*, 1:445–449.

[ILOG, 2002] ILOG (2002). ILOG AMPL CPLEX System 8.0 - User's Guide. http://www.ilog.com/products/cplex.

[Kar et al., 2000] Kar, K., Kodialam, M., and Lakshman, T. V. (2000). Minimum Interference Routing of Bandwidth Guaranteed Tunnels with MPLS Traffic Engineering Applications. *IEEE Journal on Selected Areas in Communications*, 18(12):921–940.

[Katz et al., 2003] Katz, D., Yeung, D., and Kompella, K. (2003). RFC 3630: Traffic Engineering (TE) Extensions to OSPF Version 2.

[Lee et al., 1995] Lee, W., Hluchyi, M., and Humblet, P. (1995). Routing Subject to Quality of Service Constraints in Integrated Communication Networks. *IEEE Network*, 9(4):46–55.

[Li and Rekhter, 1998] Li, T. and Rekhter, Y. (1998). RFC 2430: Provider Architecture for Differentiated Services and Traffic Engineering (PASTE).

[Liu, 2003] Liu, X. (2003). Capacity and Flow Assignment Issues in MPLS Networks. *International Journal of Network Management*, 13(3):173–179.

[Liu et al., 2000] Liu, Z., Sun, Y., and Xue, X. (2000). A Static Routing Algorithm Used in the Internet Traffic Engineering. *In Proceedings of 7th Asian-Pacific Conference on Circuits and Systems (IEEE APCCAS)*, 1:231–234.

[McCanne and Floyd, 2003] McCanne, S. and Floyd, S. (2003). Network Simulator 2.

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

[Montgomery, 1990] Montgomery, D. C. (1990). *Introduction to Statistical Quality Control*. John Wiley & Sons.

[Newhauser and Wolsey, 1998] Newhauser, G. L. and Wolsey, L. A. (1998). *Integer and Combinatorial Optimization*. John Wiley & Sons.

[Nobre et al., 2005] Nobre, E., Ponte, P. R. X., Fernandes, M., and Celestino Jr, J. (2005). IntelliDylba: Um Esquema de Balanceamento de Carga para Redes MPLS Com Aprendizado Dessassistido Baseado em Lógica Difusa e Algoritmos Genéticos. *In Proceedings of the XXIII Simpósio Brasileiro de Redes de Computadores*, 1:623–638.

[Oliveira et al., 2004a] Oliveira, A., Habib, E., and Mateus, G. (2004a). An Adaptive Routing Scheme for Traffic Engineering in IP Networks vover MPLS. *In Proceedings of XII Latin-Ibero-American Conference on Operations Research*, 1:pages in CD.

[Oliveira and Mateus, 2005a] Oliveira, A. and Mateus, G. (2005a). MPLS: Quantitative Analysis of LSP Setups(Poster). *In IEEE INFOCOM Student Workshop*, 1:pages in CD.

[Oliveira and Mateus, 2005b] Oliveira, A. and Mateus, G. (2005b). Traffic Engineering: Control Charts and LSPs. *In Proceedings of 12th International Conference on Telecommunications - ICT 2005*, 1:pages in CD.

[Oliveira et al., 2004b] Oliveira, A., Ravetti, M., and Mateus, G. (2004b). Using Control Charts for Traffic Change Detection in MPLS Networks(Poster). *In Proceeding of IEEE Mascots*, pages 61–64.

[Oliveira et al., 2005] Oliveira, A., Ravetti, M., and Mateus, G. (2005). Traffic Engineering in MPLS Networks: ARS An Adaptive Routing Scheme Based on Control Charts. *Proceeding of XXIII Simpósio Brasileiro de Redes de Computadores*, 1:595–608.

[Reeves, 1993] Reeves, C. R. (1993). *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Press.

[Resende and Ribeiro, 2003] Resende, M. G. C. and Ribeiro, C. C. (2003). A GRASP with Path-Relinking for Private Virtual Circuit Routing. *Networks*, 41(3):104–114.

[Rosen et al., 2001] Rosen, E., Viswanathan, A., and Callon, R. (2001). RFC 3031: Multiprotocol Label Switching Architecture.

[Salvadori and Battiti, 2003] Salvadori, E. and Battiti, R. (2003). A Load Balancing Scheme for Congestion Control in MPLS Networks. *In Proceedings of IEEE ISCC*, 2:951–956.

[Savage et al., 1999] Savage, S., Collins, A., and E., H. (1999). The End-to-End Effects of Internet Path Selection. *ACM SIGCOMM*, 29(4):289–299.

[Seaman et al., 2000] Seaman, M., Smith, A., Crawley, E., and Wroclawski, J. (2000). RFC 2815: Integrated Service Mappings on IEEE 802 Networks.

[Suri et al., 2001] Suri, S., Waldvogel, M., and Warkhede, P. R. (2001). Profile-based Routing: A New Framework for MPLS Traffic Engineering. *Lecture Notes in Computer Science*, 2156:138–157.

[Sytsma and Manley, 1999] Sytsma, S. and Manley, K. (1999). Common Control Chart Cookbook. http://www.sytsma.com/tqmtools/charts.html.

[Uhlig, 2005] Uhlig, S. (2005). A Multiple-objectives Evolutionary Perspective to Interdomain Traffic Engineering. *International Journal of Computational Intelligence and Applications*, 5(2):1–16.

[Vanderbei, 1997] Vanderbei, R. J. (1997). *Linear Programming: Foundations and Extensions*. Kluwer Academic Publishers.

[Wang et al., 2002] Wang, B., Su, X., and Chen, C. P. (2002). A New Bandwidth Guaranteed Routing Algorithm for MPLS Traffic Engineering. *Proceedings of ICC*, 2:1001–1005.

[Xiao et al., 1999] Xiao, X., Hannan, A., Bailey, B., and Ni, L. M. (1999). Traffic Engineering with MPLS in the Internet. *In Proceedings of IEEE Network*, 1:32–37.

[Yilmaz and Matta, 2002] Yilmaz, S. and Matta, I. (2002). On the Scalability-Performance Tradeoffs in MPLS and IP Routing. *Technical Report - BUCS-TR-2002-013*.

[Y.Wang and Z.Wang, 1999] Y.Wang and Z.Wang (1999). Explicit Routing Algorithms for Internet Traffic Engineering. *In Proceedings of 8th Computer Communications and Networks*, 1:582–588.

[Zhang and Wu, 2005] Zhang, S. and Wu, Z. (2005). Designs of Control Charts With Supplementary Runs Rules. *Computers and Industrial Engineering*, 49(1):76–97.

Appendix A

Detailed Results

A.1 Chapter 4

A.1.1 Genetic Algorithm Input Parameter Analysis

The graphs in Figures A.1- A.5 plot the fitness obtained with each possible parameter combination (200 points in the abscissa) with traffic varying from 10 to 50 LSP requests with a step of 10.

It can be noticed that all graphics present clusters of regions where the fitness of the solutions are very similar.

When traffic load is low (10 and 20) requests, the results for all combinations are practically the same. When traffic load increases, the parameters start to make a difference but even so it can be noticed that all graphs present horizontal lines, representing clusters of good solutions.

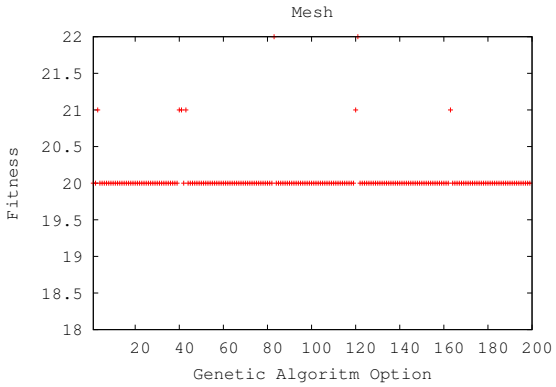
A.1.2 CPLEX and GA Comparison

The solutions found using CPLEX were compared with the solutions found by the best execution of the genetic algorithm and are shown in Figures A.6 - A.10.

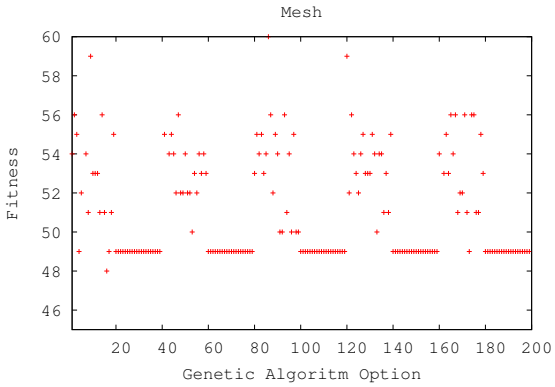
The graphs show CPLEX results for the different models described in Chapter 3 for different topologies:

- MinRej - model that takes into account the number of rejections but does not try to balance the load. It is a lower bound for our two-step model.
- MinRejBal - model that takes into account the number of rejections and tries to balance the load. Gives the results for our two-step model.
- Genetico - heuristic that takes into account the number of rejections and tries to balance the load using policies.

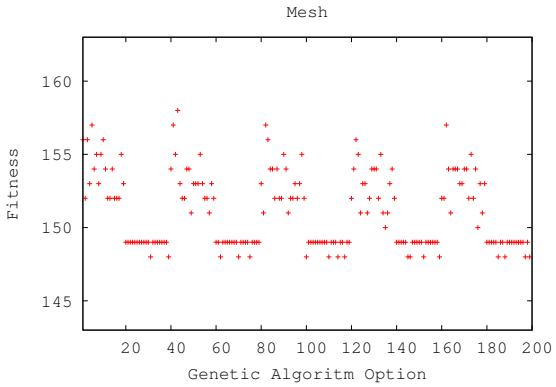
For the scenarios with light traffic, the GA results are similar to the lower hound, where there is no need for load balancing. When the traffic load starts to increase, the GA solution is between the ones found when the traffic is balanced and the solutions found when it is not balanced.



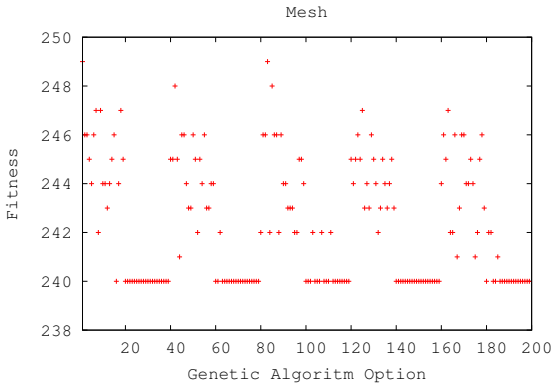
(a) 10



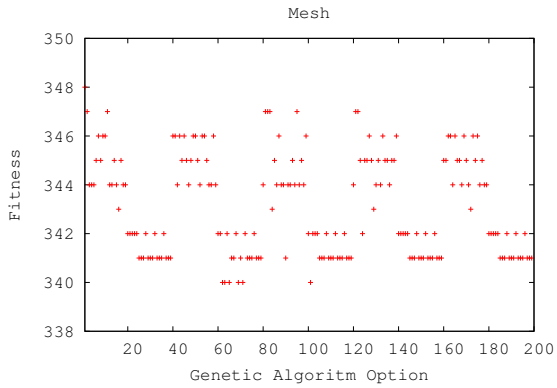
(b) 20



(c) 30

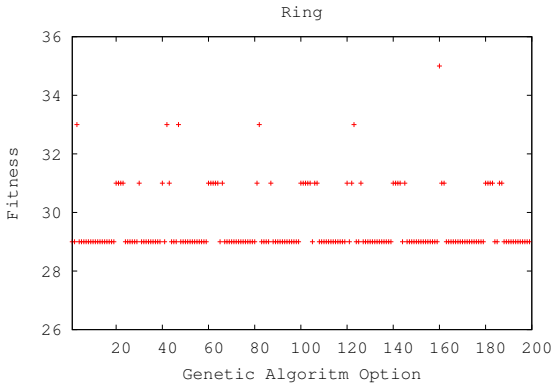


(d) 40

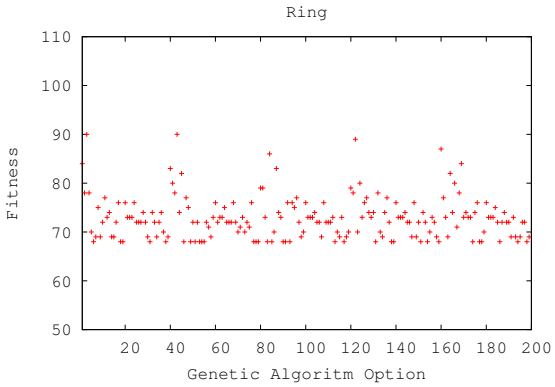


(e) 50

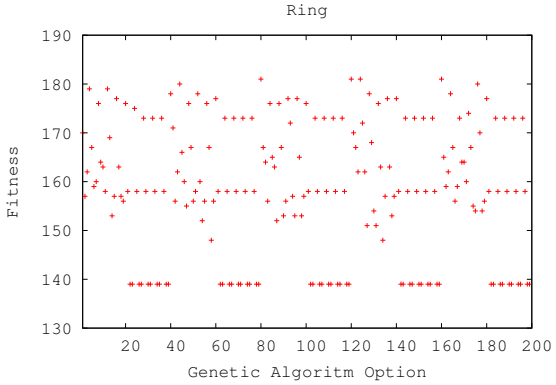
Figure A.1: Various Results For Mesh Network Using Genetic Algorithms



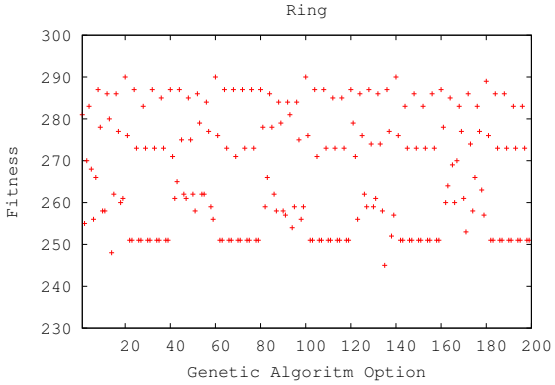
(a) 10



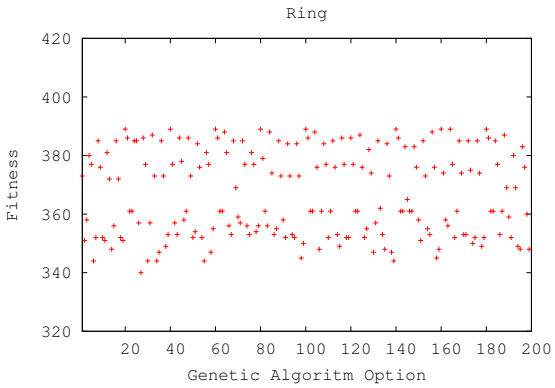
(b) 20



(c) 30

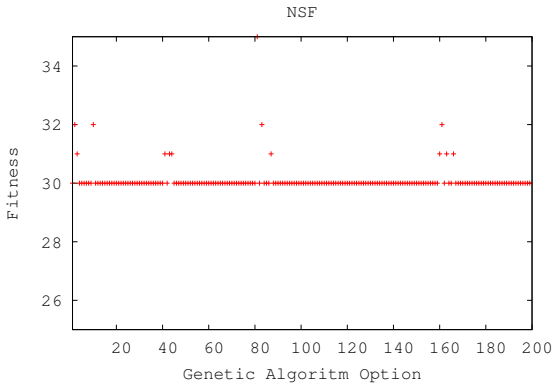


(d) 40

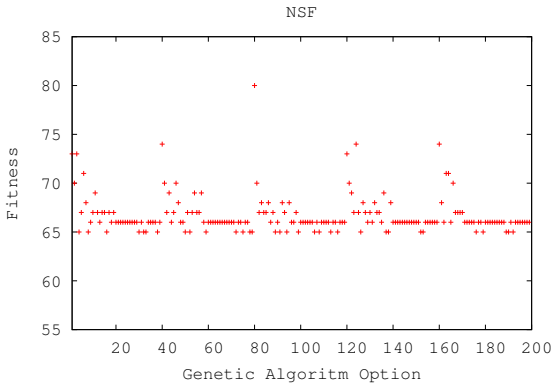


(e) 50

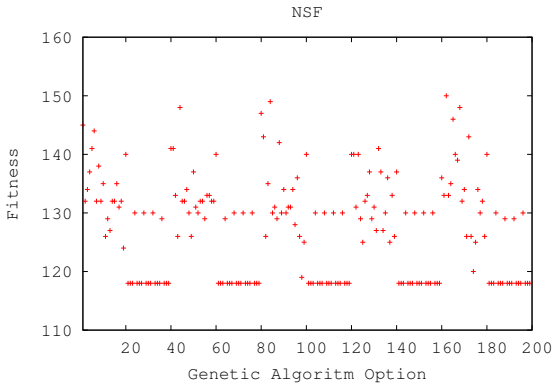
Figure A.2: Various Results For Ring Network Using Genetic Algorithms



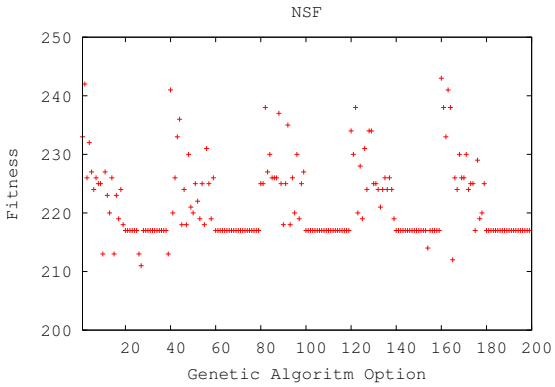
(a) 10



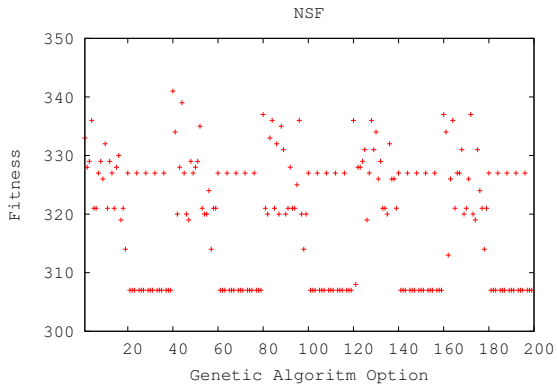
(b) 20



(c) 30

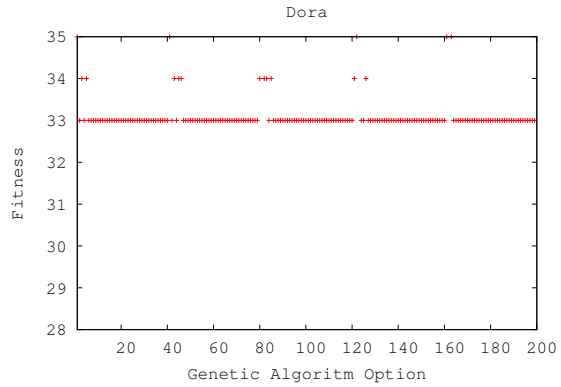


(d) 40

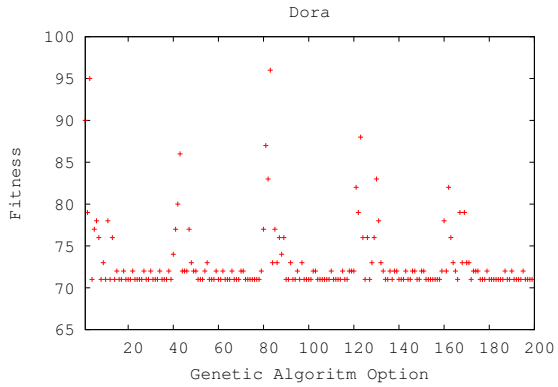


(e) 50

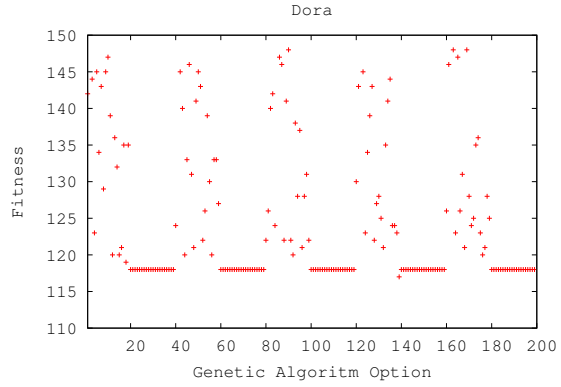
Figure A.3: Various Results For NSF Network Using Genetic Algorithms



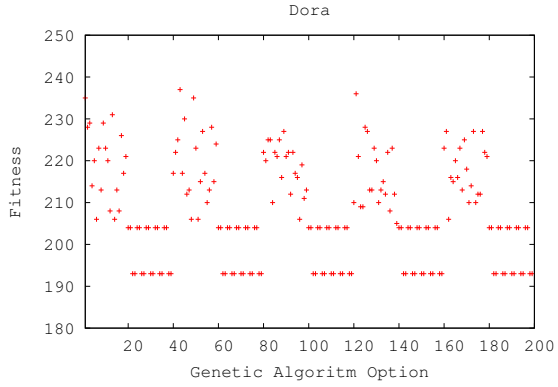
(a) 10



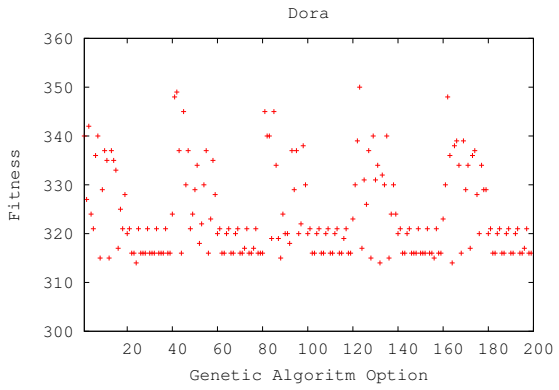
(b) 20



(c) 30

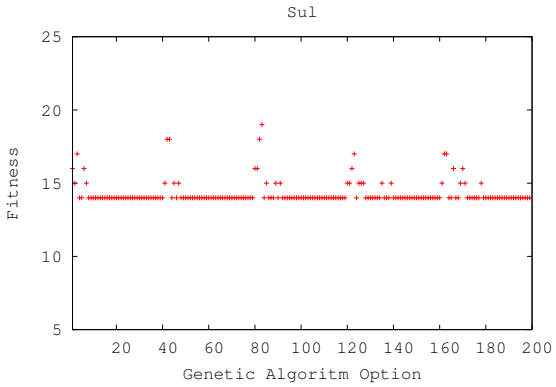


(d) 40

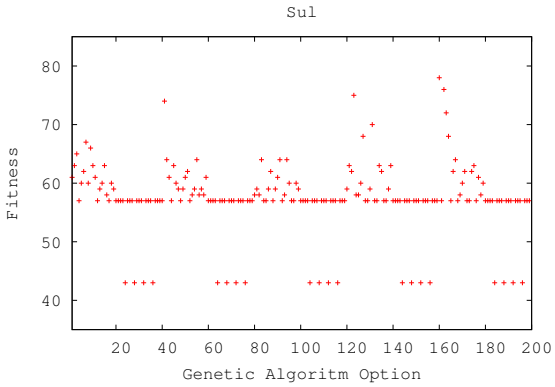


(e) 50

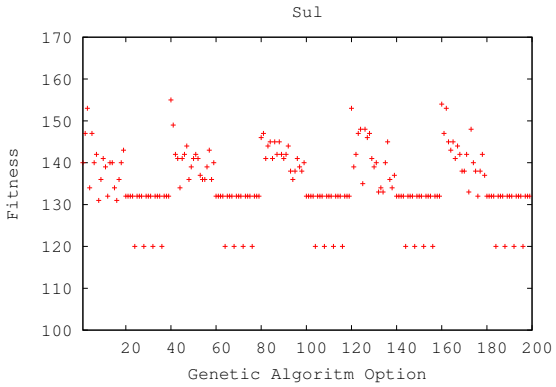
Figure A.4: Various Results For Dora Network Using Genetic Algorithms



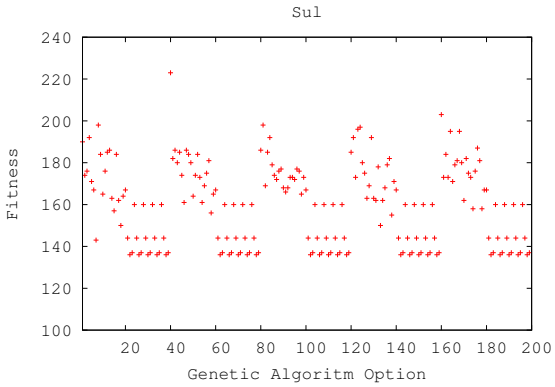
(a) 10



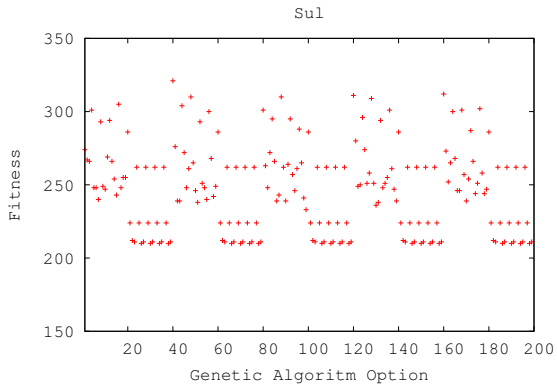
(b) 20



(c) 30



(d) 40



(e) 50

Figure A.5: Various Results For Sul Network Using Genetic Algorithms

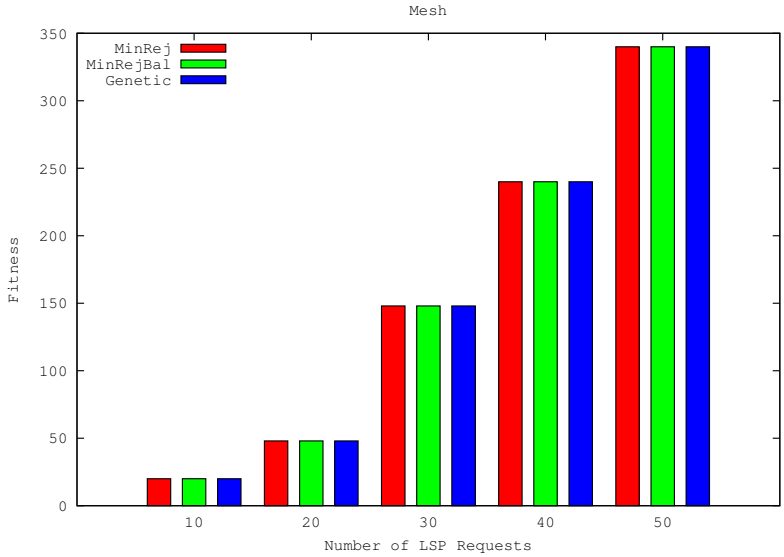


Figure A.6: CPLEX and GA Results Comparison for the Mesh Network

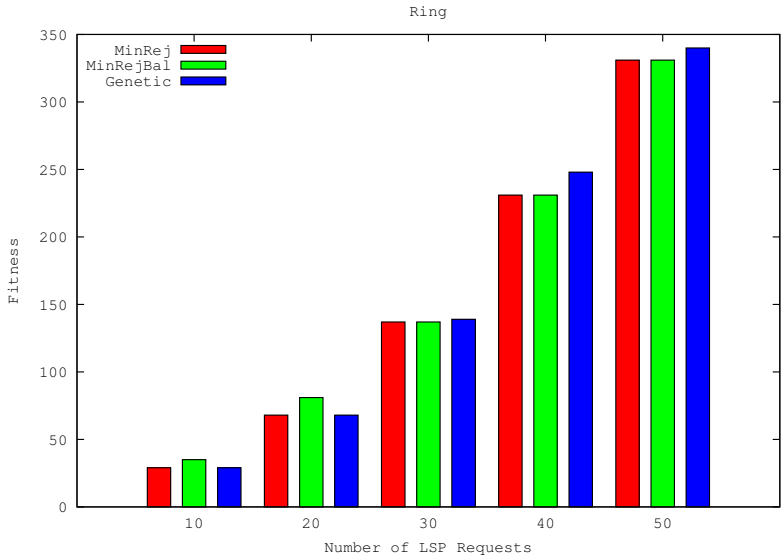


Figure A.7: CPLEX and GA Results Comparison for the Ring Network

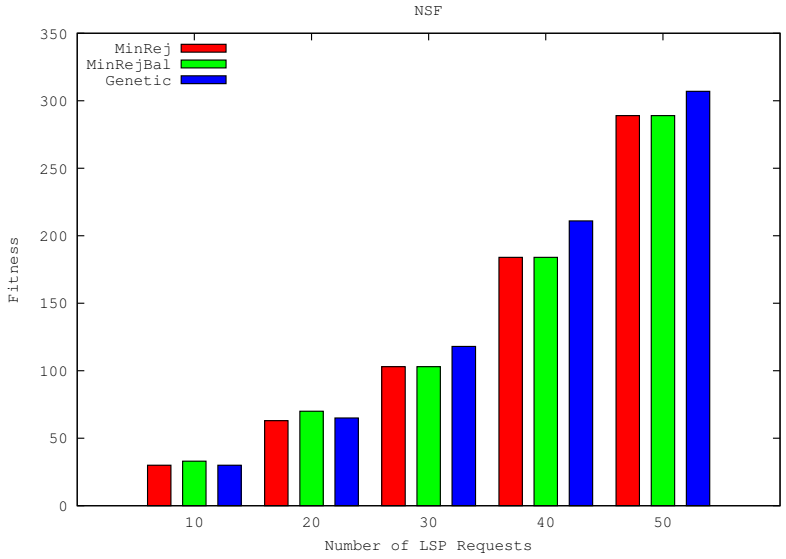


Figure A.8: CPLEX and GA Results Comparison for the NSF Network

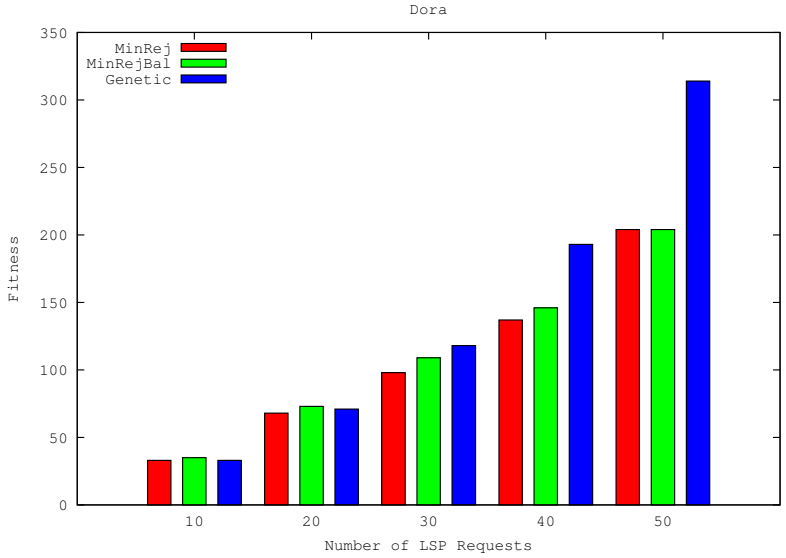


Figure A.9: CPLEX and GA Results Comparison for the Dora Network

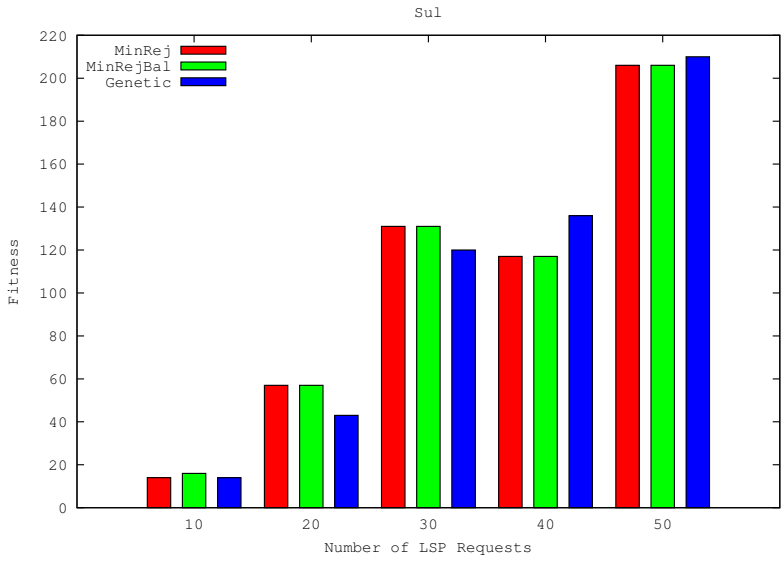


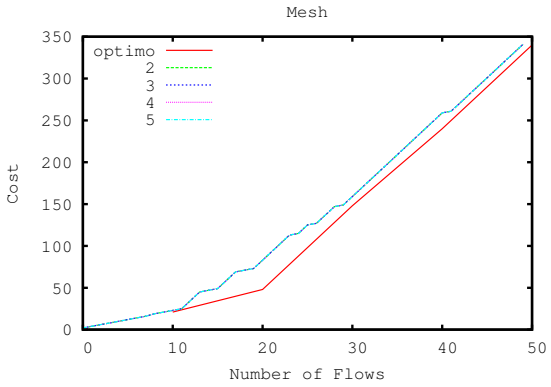
Figure A.10: CPLEX and GA Results Comparison for the Sul Network

A.2 Chapter 5

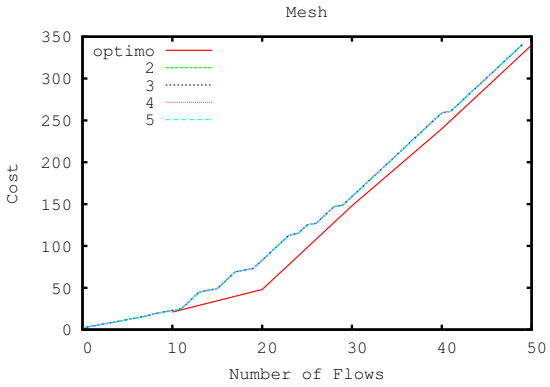
A.2.1 Policies Configuration - Number of Alternative Routes

The optimal result obtained using the ILP model described in Chapter 3 is compared to the result obtained using the policies described for the initialization phase of the population for the GA described in Chapter 4.

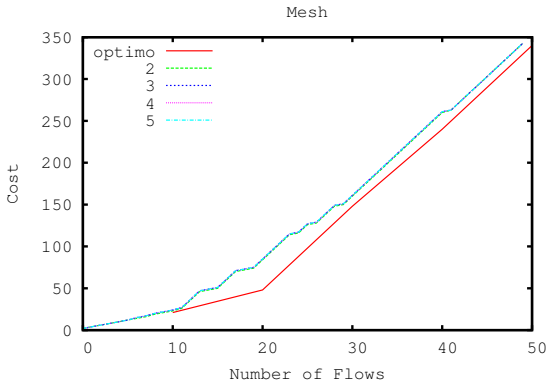
The two best number of alternative routes for most of the policies is four or five. The only exception is for Adaptive Balance policy that has the best performance when the number of alternative routes is two for the topologies Dora, NSF and Carrier. The results are shown in Figures A.11 - A.15.



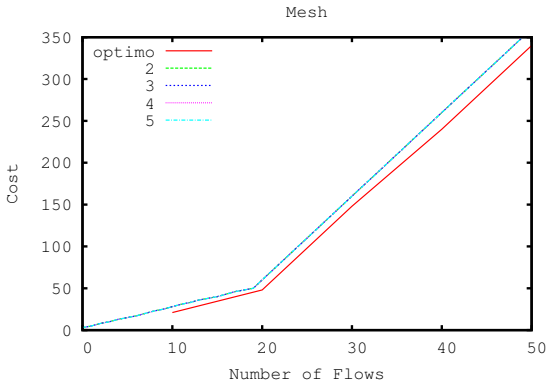
(a) Min-Hop Policy



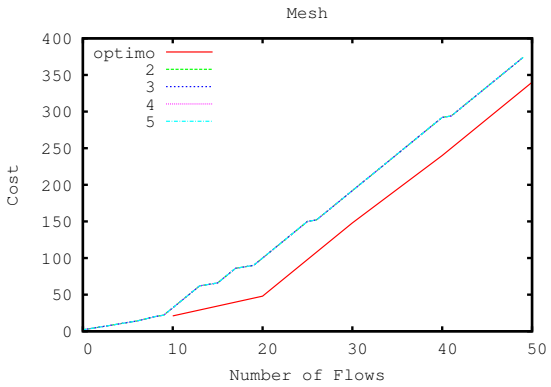
(b) Adaptive Min-Hop Policy



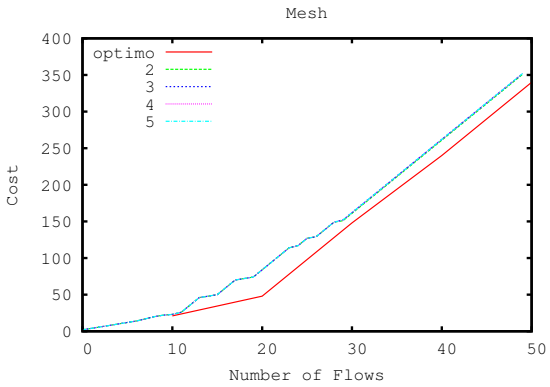
(c) Balance Policy



(d) Adaptive Balance Policy

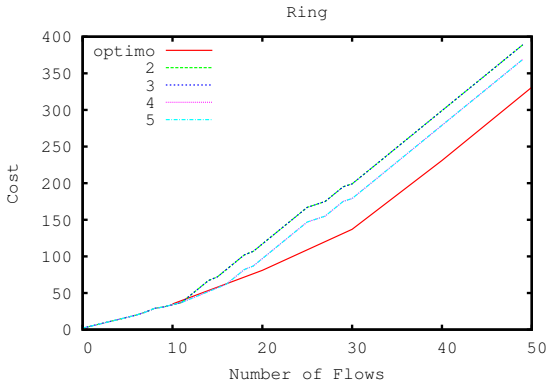


(e) Limited Utilization Policy

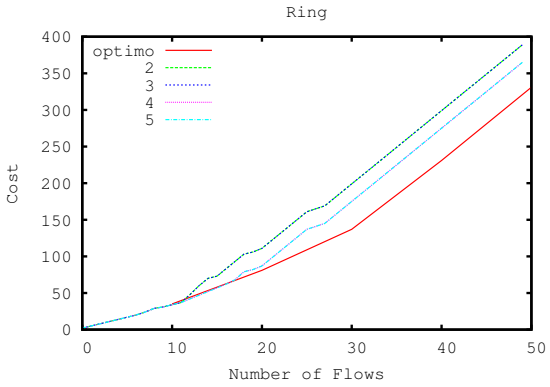


(f) Adaptive Limited Utilization Policy

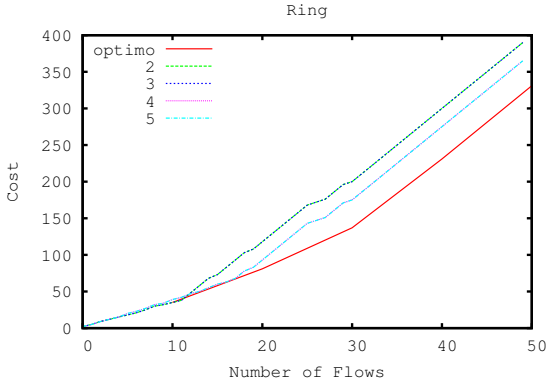
Figure A.11: Influence of Number of Alternative Routes on the Policies Cost For Mesh



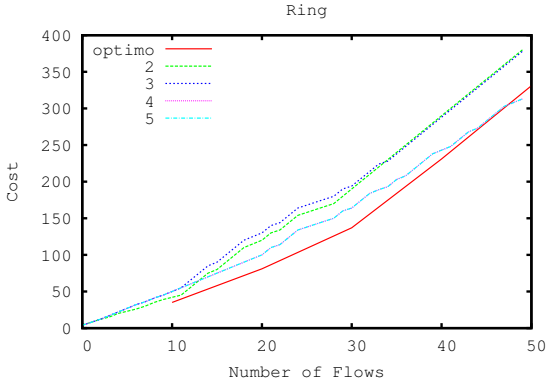
(a) Min-Hop Policy



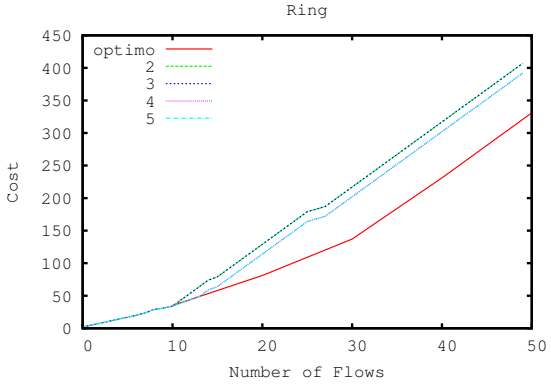
(b) Adaptive Min-Hop Policy



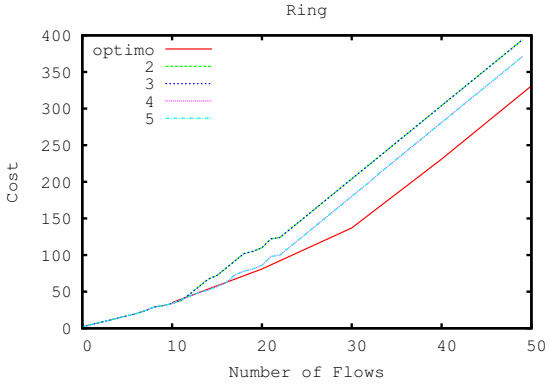
(c) Balance Policy



(d) Adaptive Balance Policy

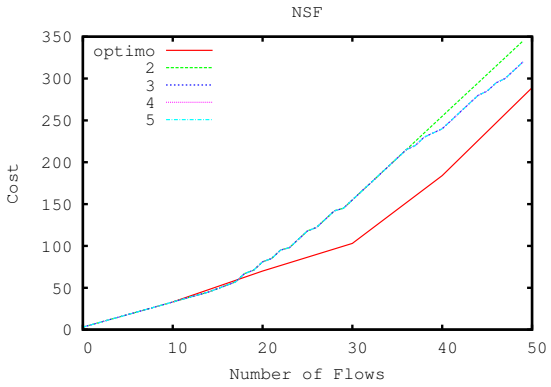


(e) Limited Utilization Policy

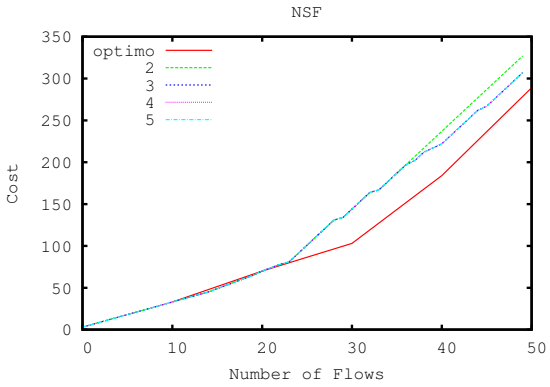


(f) Adaptive Limited Utilization Policy

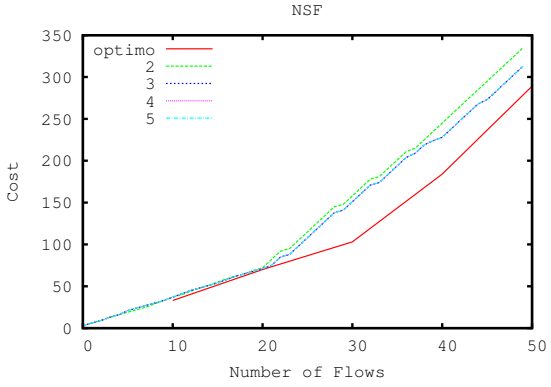
Figure A.12: Influence of Number of Alternative Routes on the Policies Cost For Ring



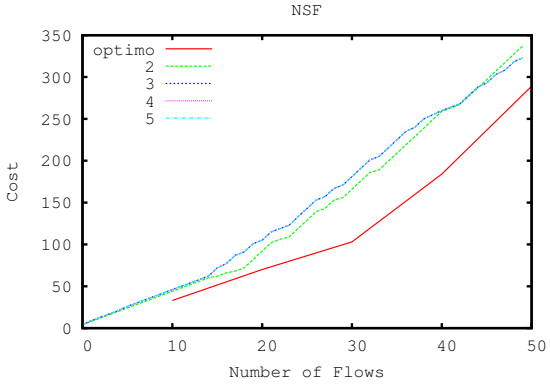
(a) Min-Hop Policy



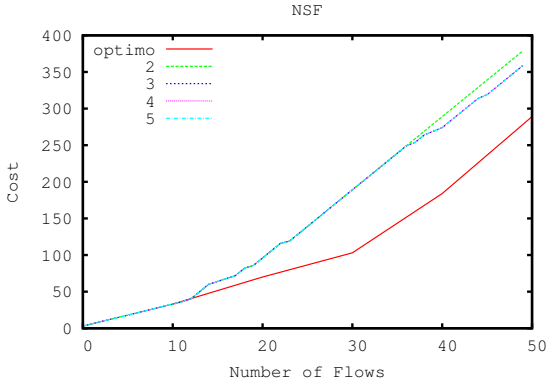
(b) Adaptive Min-Hop Policy



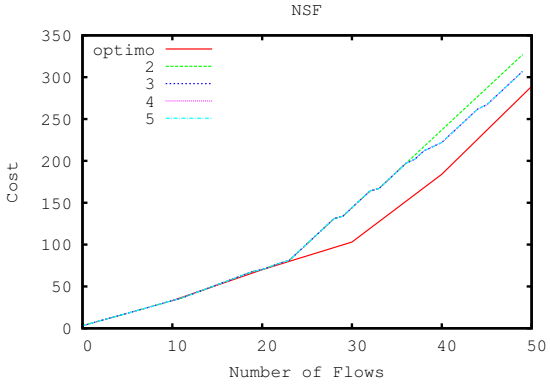
(c) Balance Policy



(d) Adaptive Balance Policy

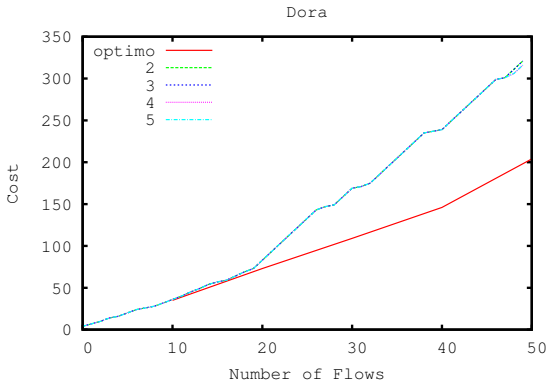


(e) Limited Utilization Policy

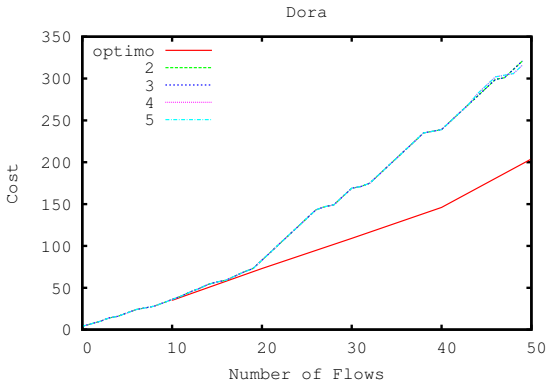


(f) Adaptive Limited Utilization Policy

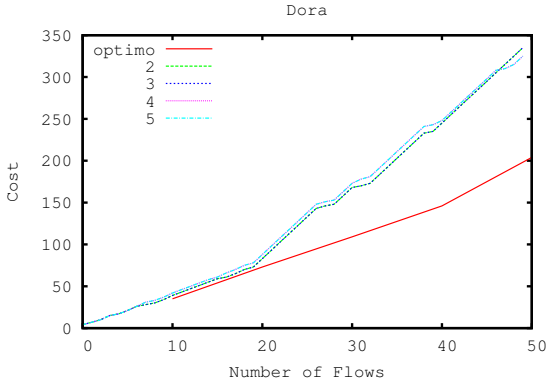
Figure A.13: Influence of Number of Alternative Routes on the Policies Cost For NSF



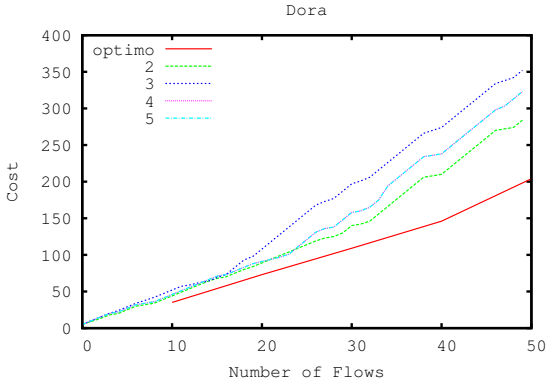
(a) Min-Hop Policy



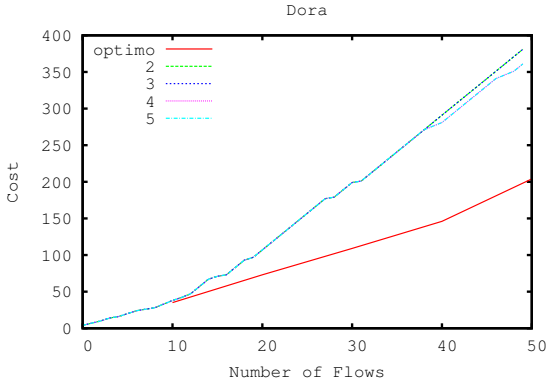
(b) Adaptive Min-Hop Policy



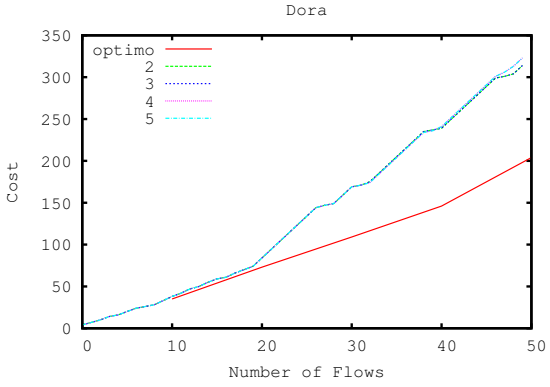
(c) Balance Policy



(d) Adaptive Balance Policy

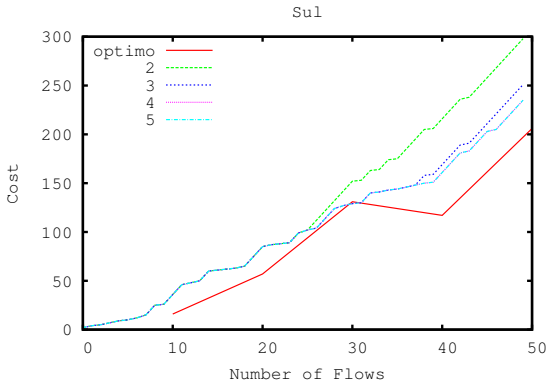


(e) Limited Utilization Policy

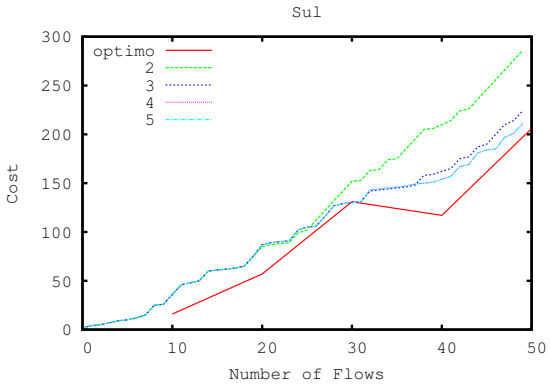


(f) Adaptive Limited Utilization Policy

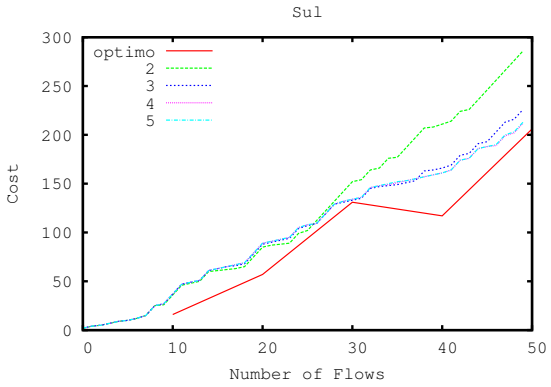
Figure A.14: Influence of Number of Alternative Routes on the Policies Cost For Dora



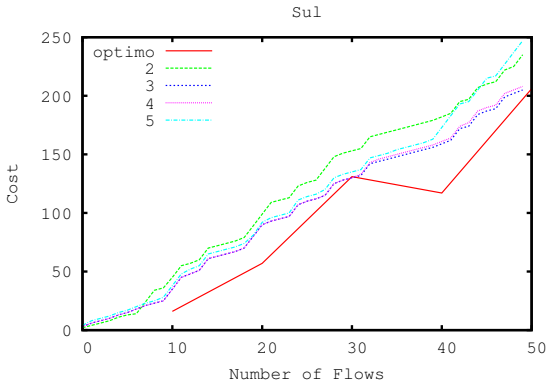
(a) Min-Hop Policy



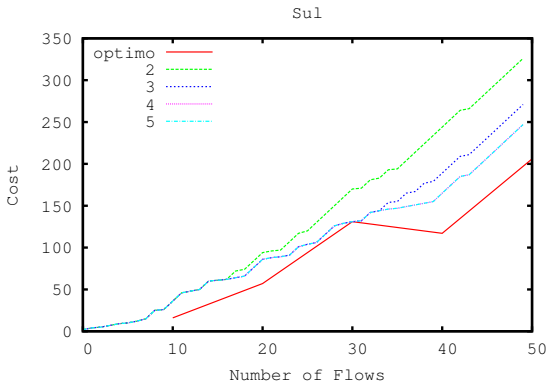
(b) Adaptive Min-Hop Policy



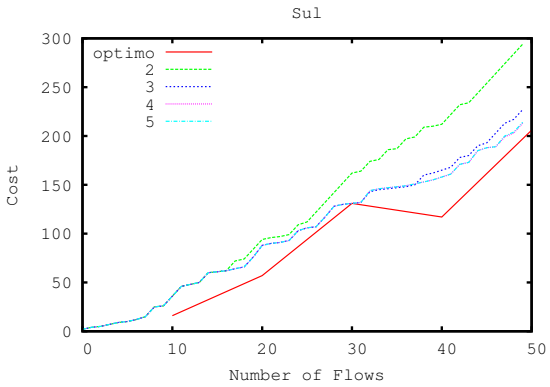
(c) Balance Policy



(d) Adaptive Balance Policy



(e) Limited Utilization Policy



(f) Adaptive Limited Utilization Policy

Figure A.15: Influence of Number of Alternative Routes on the Policies Cost For Sul

A.2.2 Policies Comparison - Cost

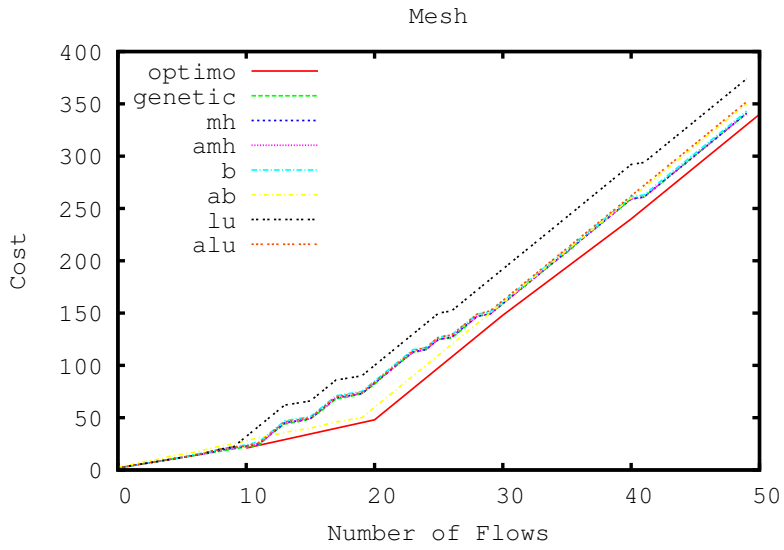


Figure A.16: Cost for Each Policy Over the Number of Flows For Mesh Network

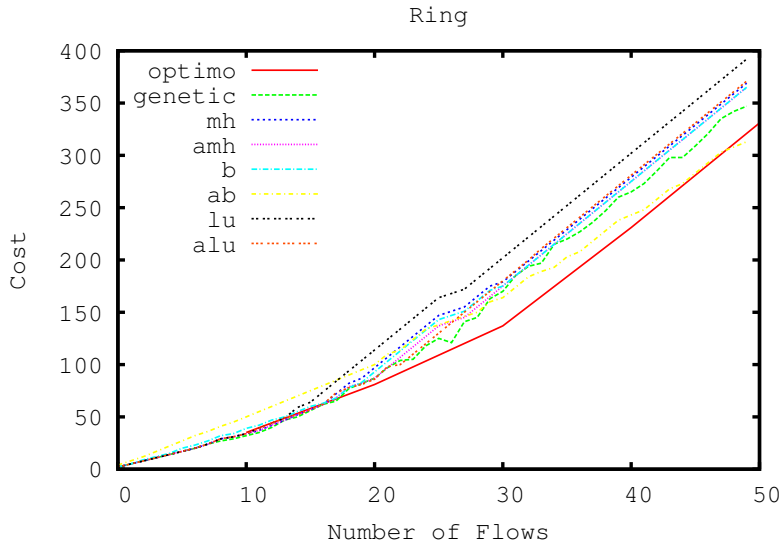


Figure A.17: Cost for Each Policy Over the Number of Flows For Ring Network

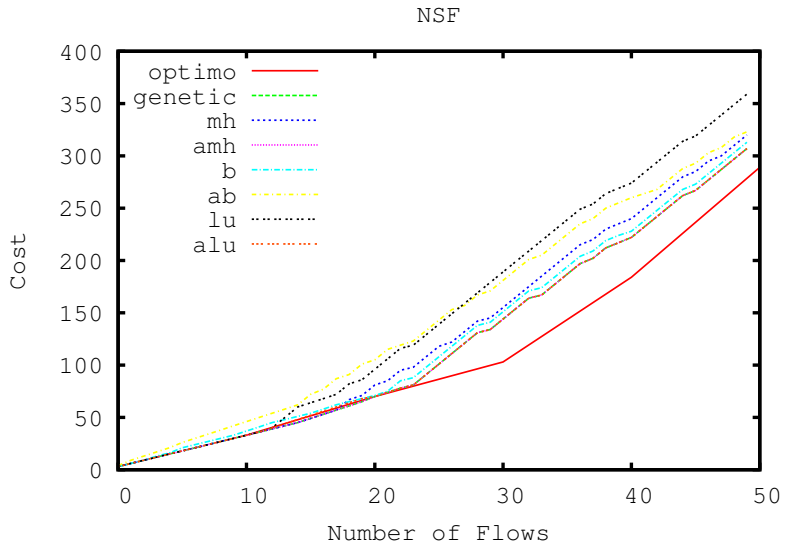


Figure A.18: Cost for Each Policy Over the Number of Flows For NSF Network

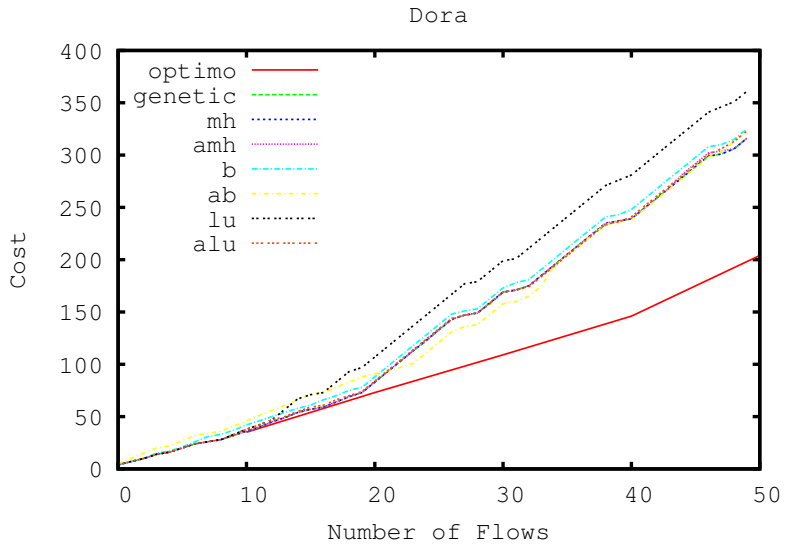


Figure A.19: Cost for Each Policy Over the Number of Flows For Dora Network

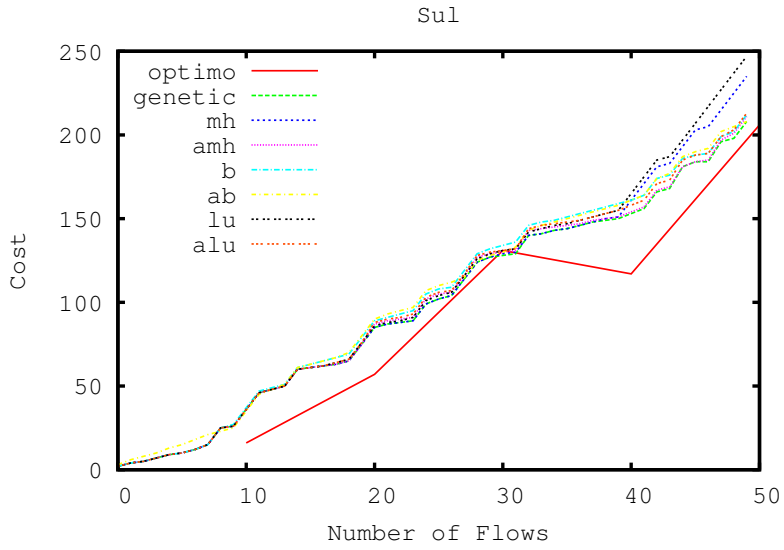


Figure A.20: Cost for Each Policy Over the Number of Flows For Sul Network

Once we find the best configuration for each policy, we compare the policies considering the total cost. The results are shown in Figures A.16 - A.20.

A.2.3 Policies Comparison - Rerouting

Figures A.21- A.25 present the number of reroutes needed by each policy as the number of flows increases.

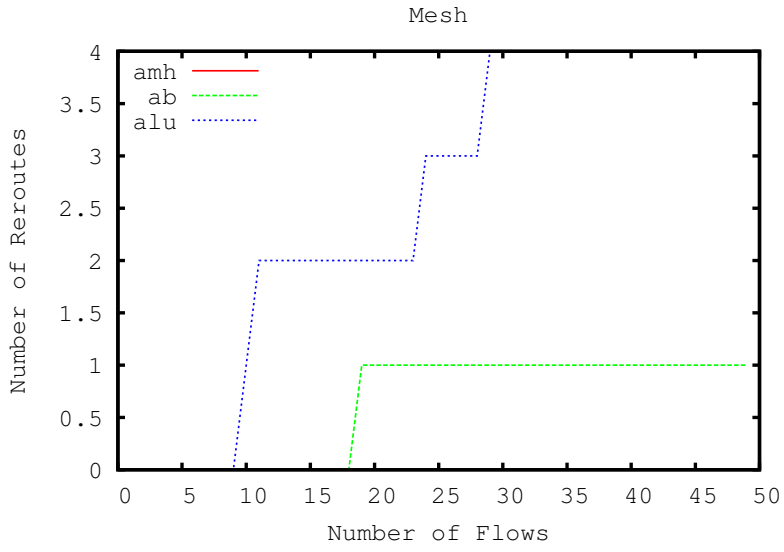


Figure A.21: Reroutes over Number of Flows For Mesh Network

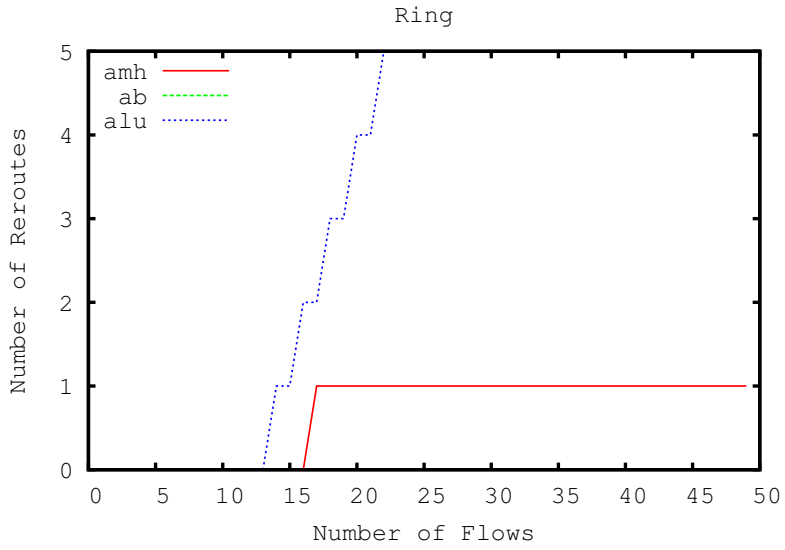


Figure A.22: Reroutes over Number of Flows For Ring Network

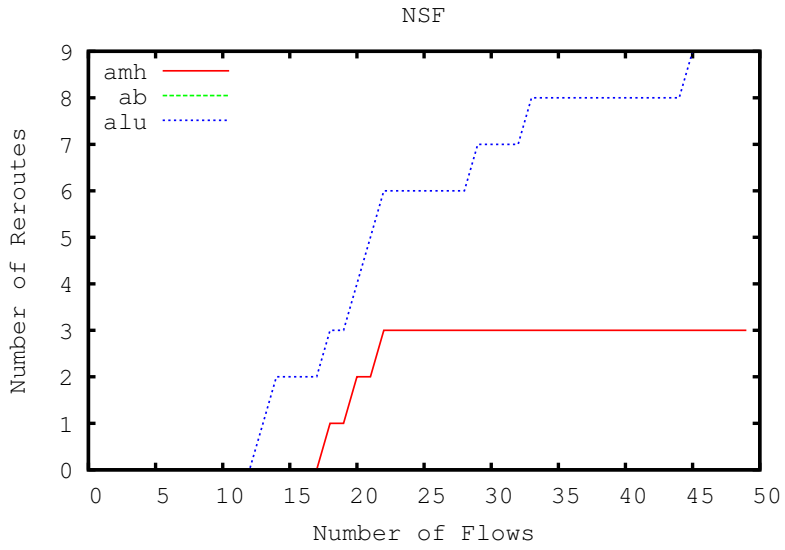


Figure A.23: Reroutes over Number of Flows For NSF Network

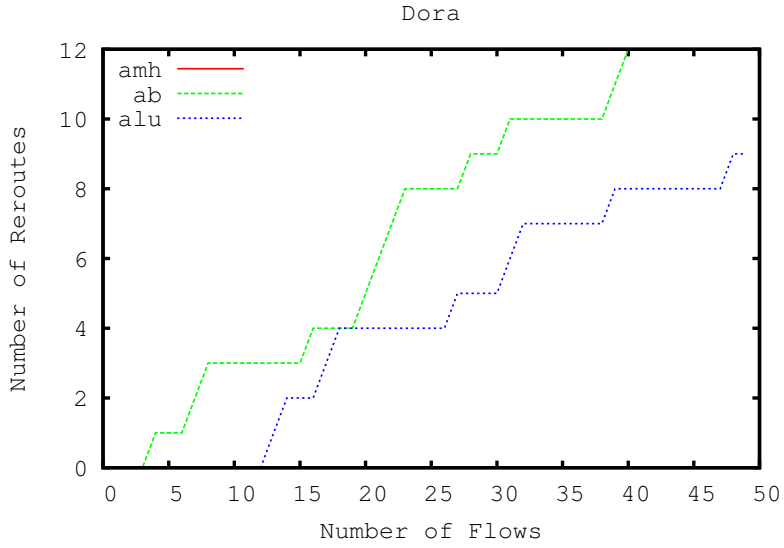


Figure A.24: Reroutes over Number of Flows For Dora Network

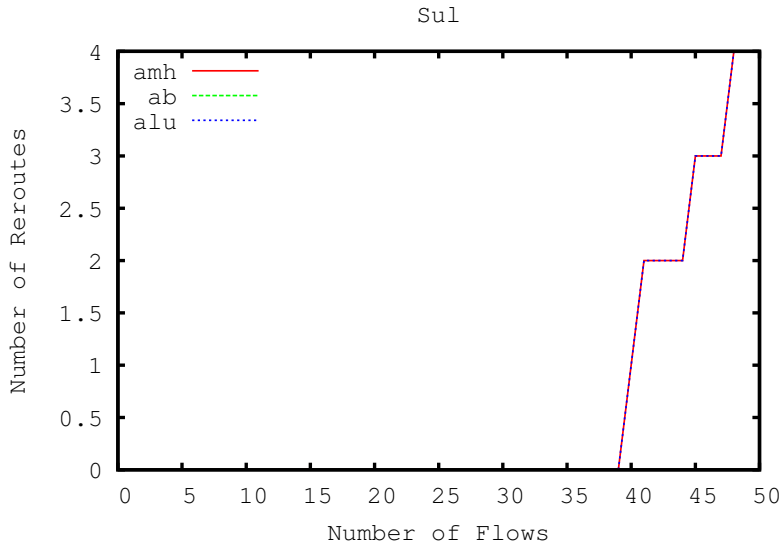


Figure A.25: Reroutes over Number of Flows For Sul Network

A.2.4 Setting Dynamic Paths

Figures A.26- A.30 present the cost of each policy as a function of the LSP holding time.

This experiment shows the advantage of using the genetic algorithm instead of one single policy.

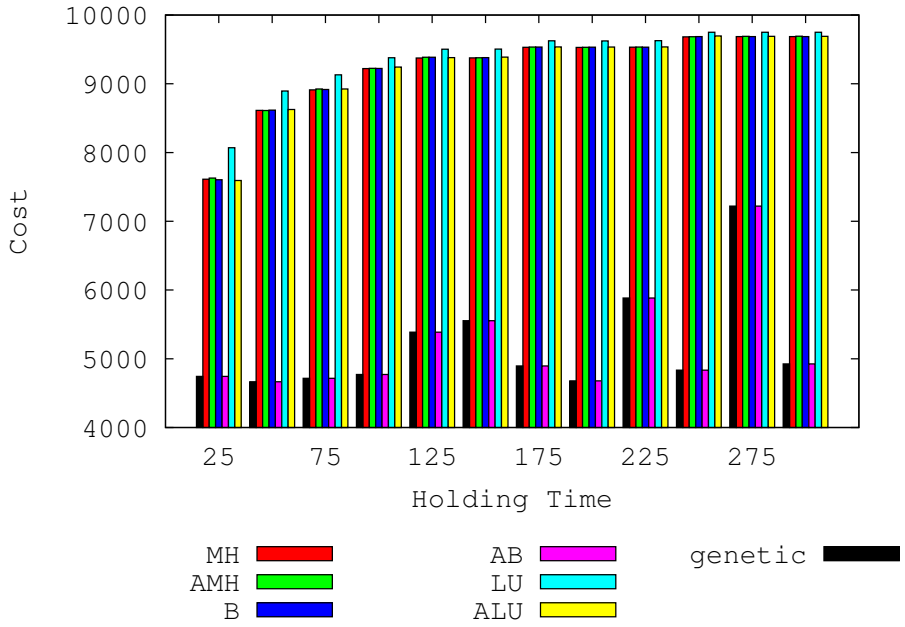


Figure A.26: Cost as a Function of LSP Holding Time For Mesh Network

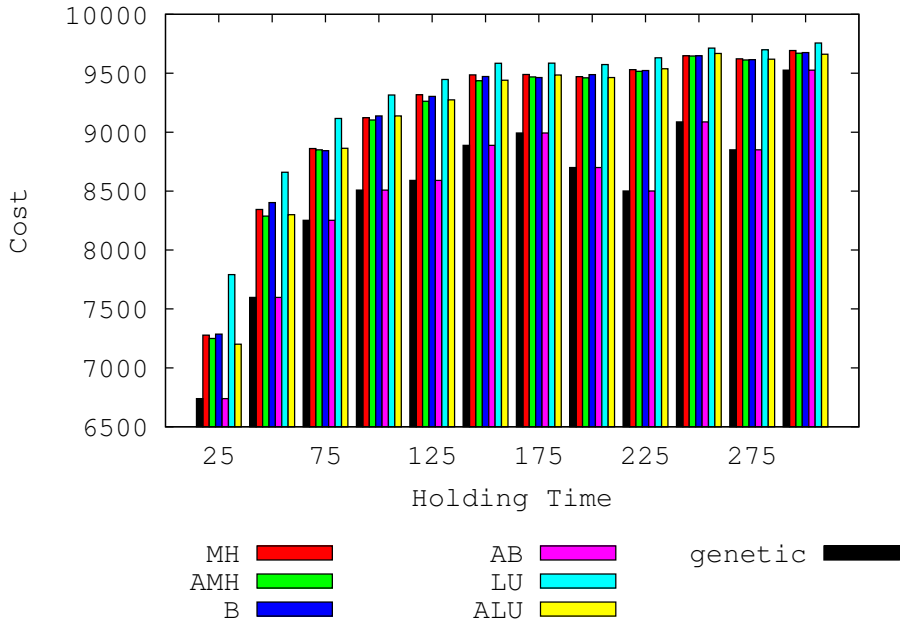


Figure A.27: Cost as a Function of LSP Holding Time For Ring Network

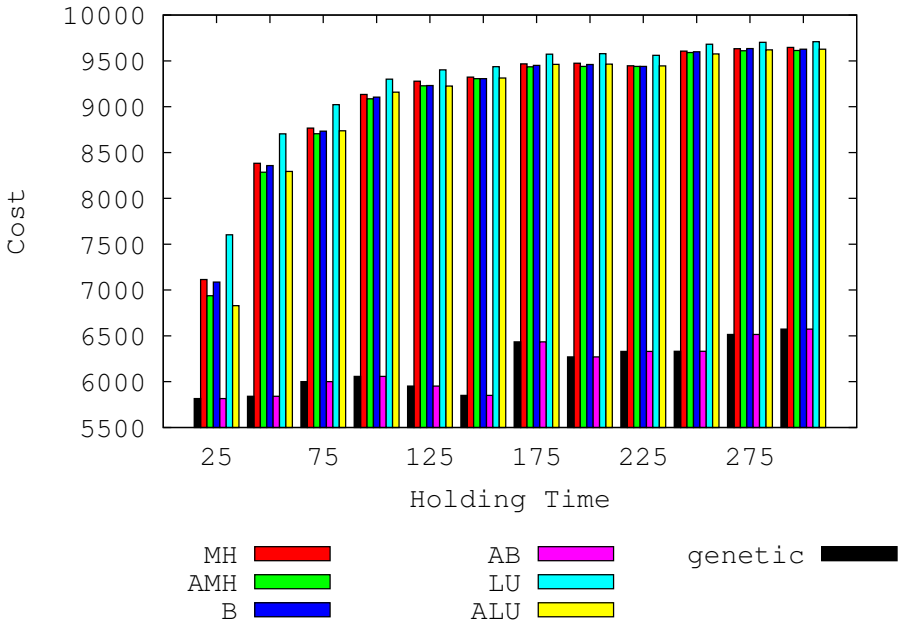


Figure A.28: Cost as a Function of LSP Holding Time For NSF Network

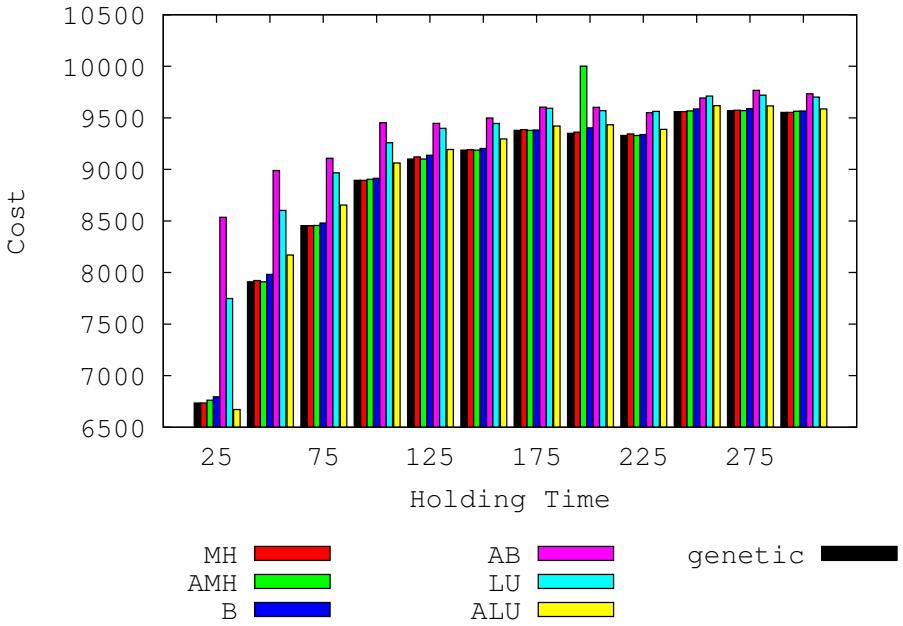


Figure A.29: Cost as a Function of LSP Holding Time For Dora Network

The GA is flexible and has the result of the best policy in most part of the time.

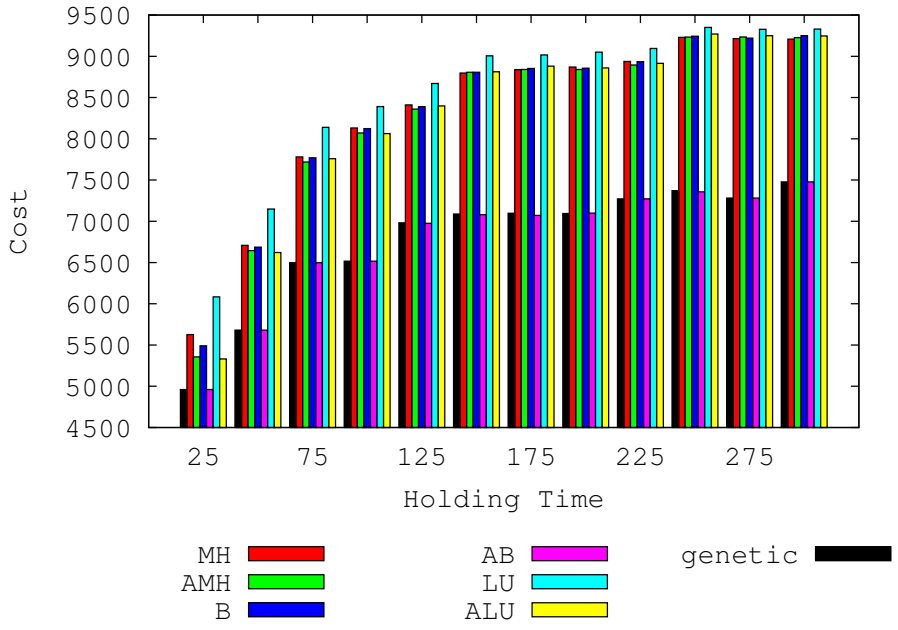


Figure A.30: Cost as a Function of LSP Holding Time For Sul Network

A.2.5 Link Failure

Figures A.31- A.35 show the performance of each policy in case of a link failure.

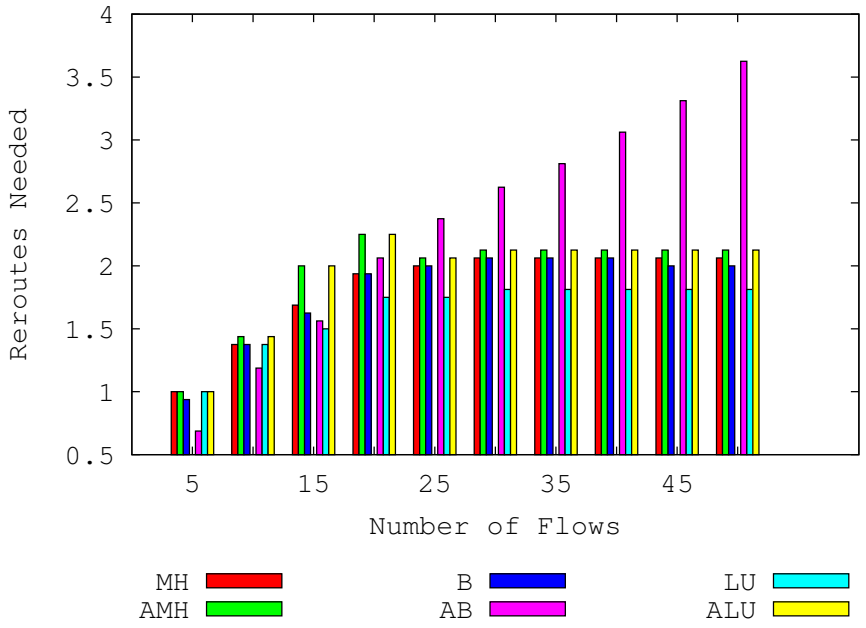


Figure A.31: Reroutes Needed for Each Policy in Case of Link Failure For Mesh Network

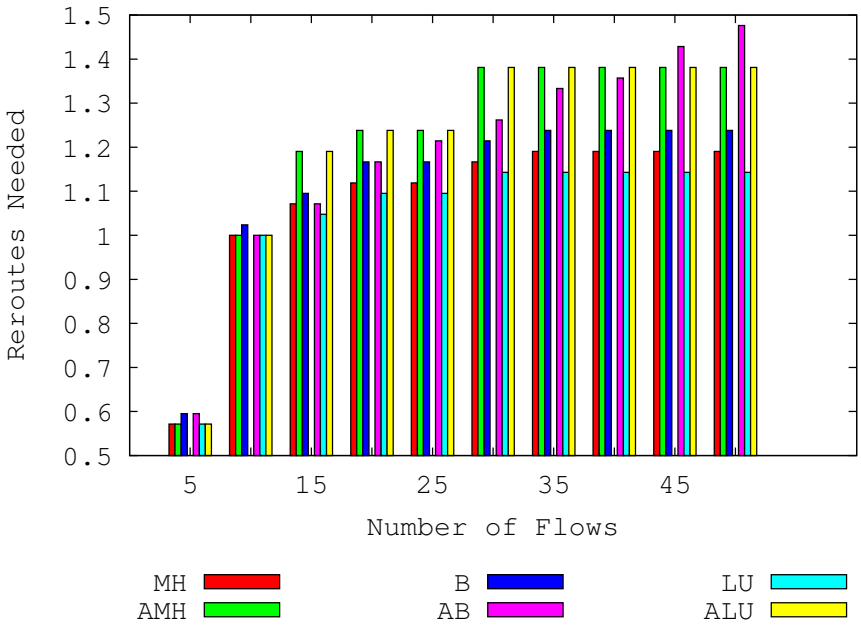


Figure A.32: Reroutes Needed for Each Policy in Case of Link Failure For ring Network

It is noticeable that the policy AB is the best for less loaded networks, but it increases the number of reroutings rapidly when the load increases. When the network becomes congested, the policy LU is the policy that requires the less number of reroutings. For the most part of topologies, the number of reroutings needed is less than 2. It can be noticed that the difference among the policies is small when the network is not congested.

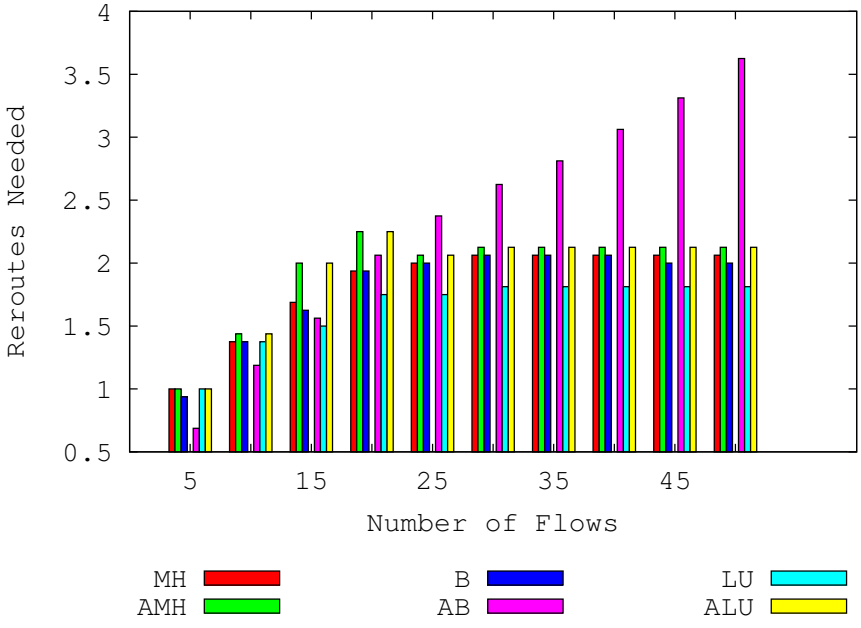


Figure A.33: Reroutes Needed for Each Policy in Case of Link Failure For NSF Network

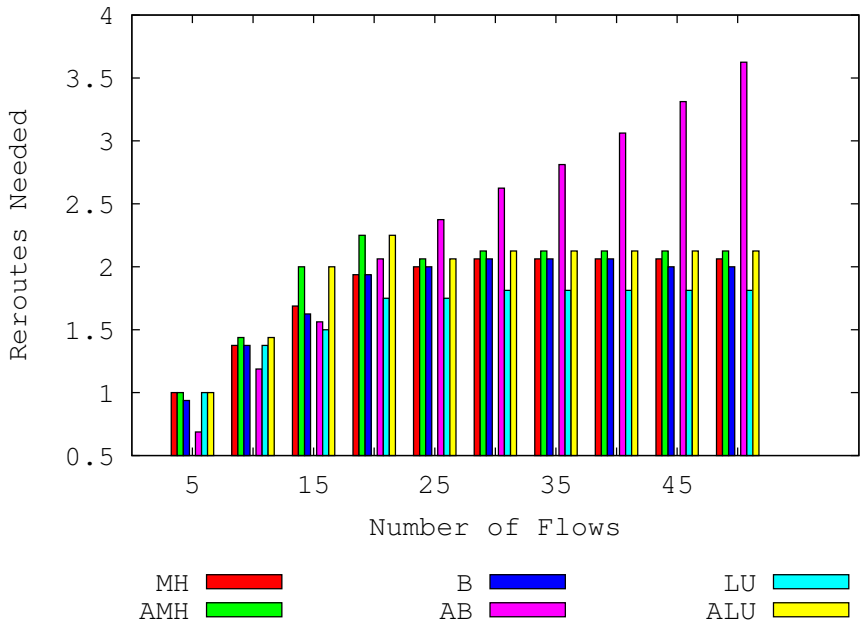


Figure A.34: Reroutes Needed for Each Policy in Case of Link Failure For Dora Network

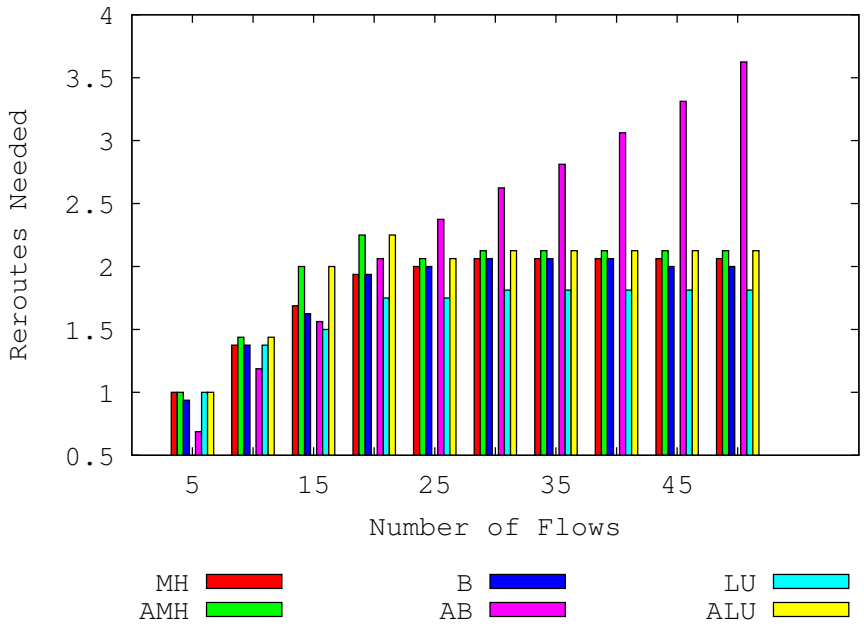


Figure A.35: Reroutes Needed for Each Policy in Case of Link Failure For Sul Network

Figures A.36 - A.40 show that for all policies the success rate in finding new routes for LSPs after a link failure is more than 50%.

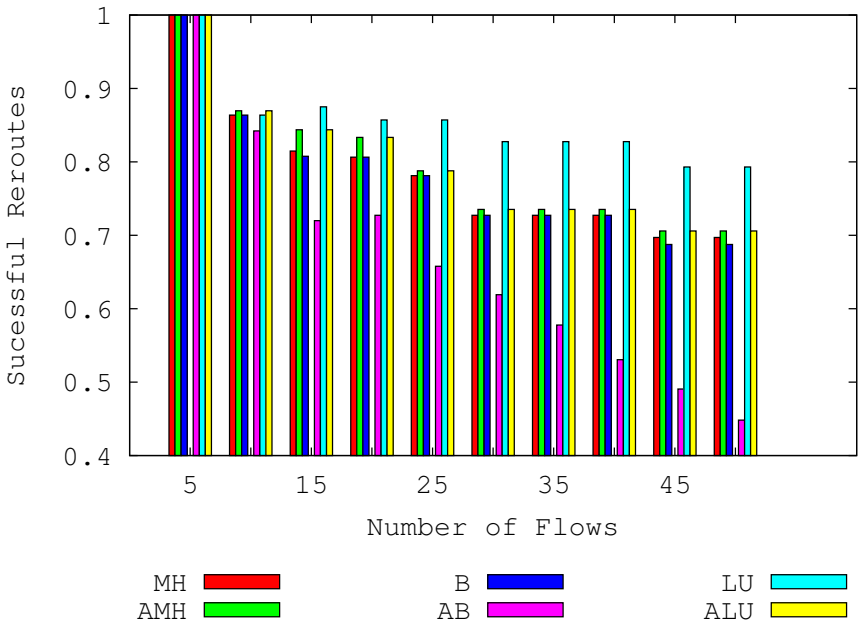


Figure A.36: % of Successful Rerouting in Case of Link Failure For Mesh Network

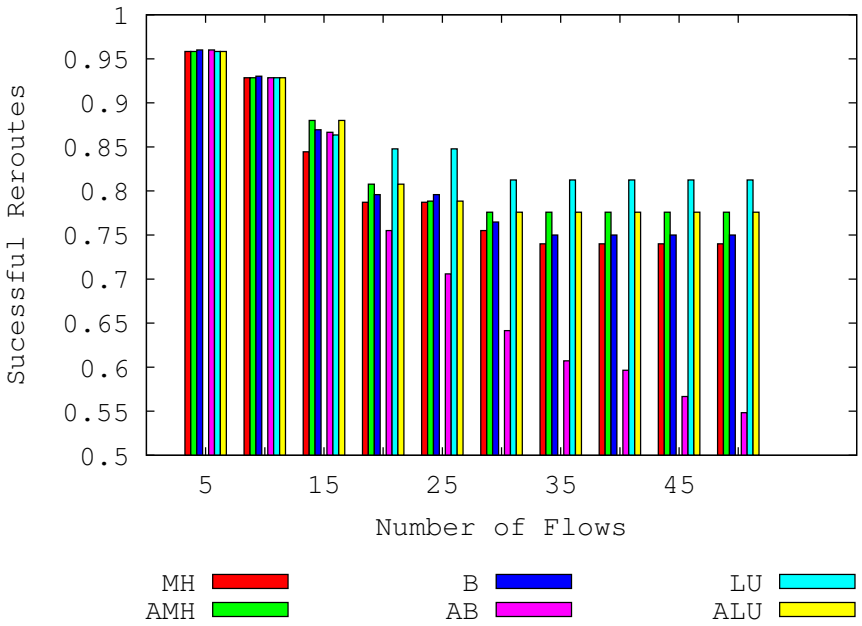


Figure A.37: % of Successful Rerouting in Case of Link Failure For Ring Network

The policy LU is the best since it leaves more space for future requests. The policy AB has already balanced the traffic and has fewer options in case of link failures.

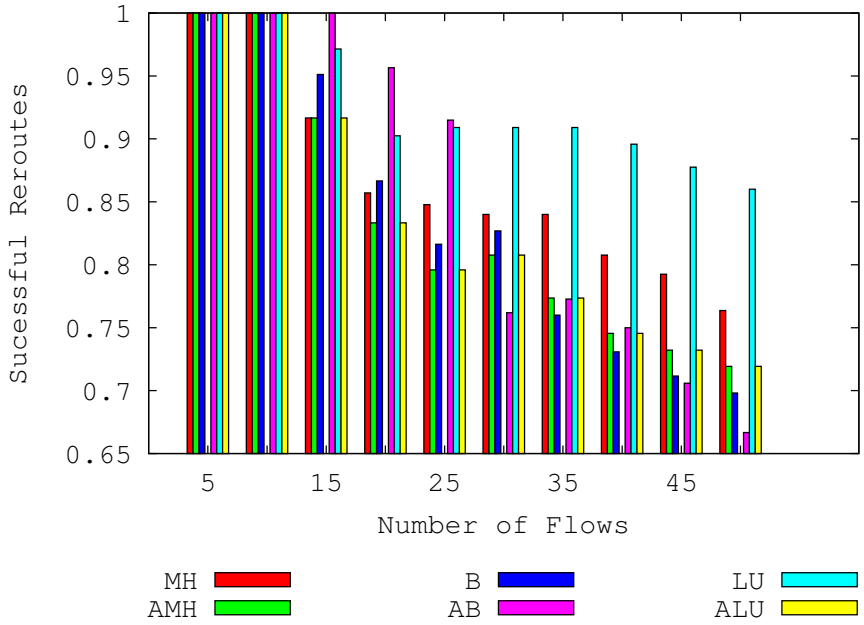


Figure A.38: % of Successful Rerouting in Case of Link Failure For NSF Network

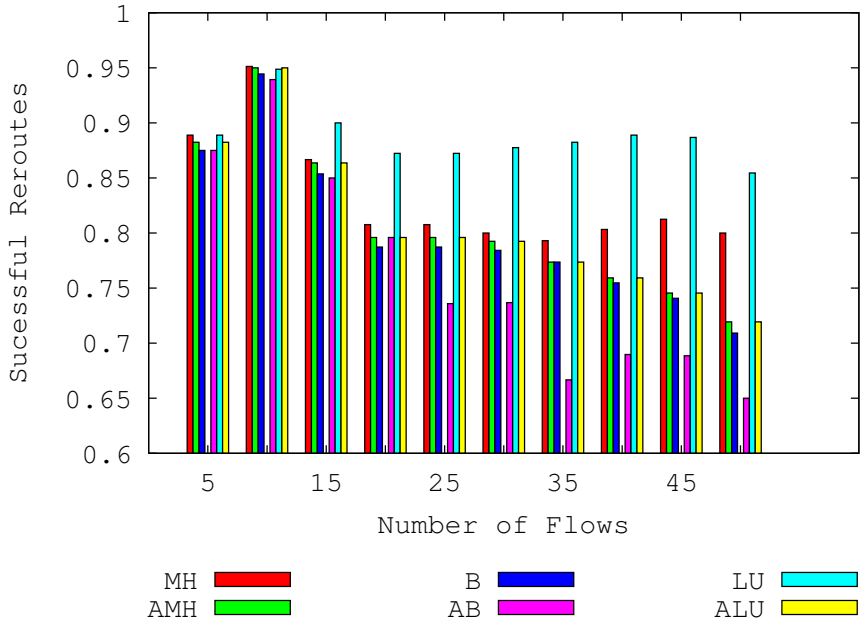


Figure A.39: % of Successful Rerouting in Case of Link Failure For Dora Network

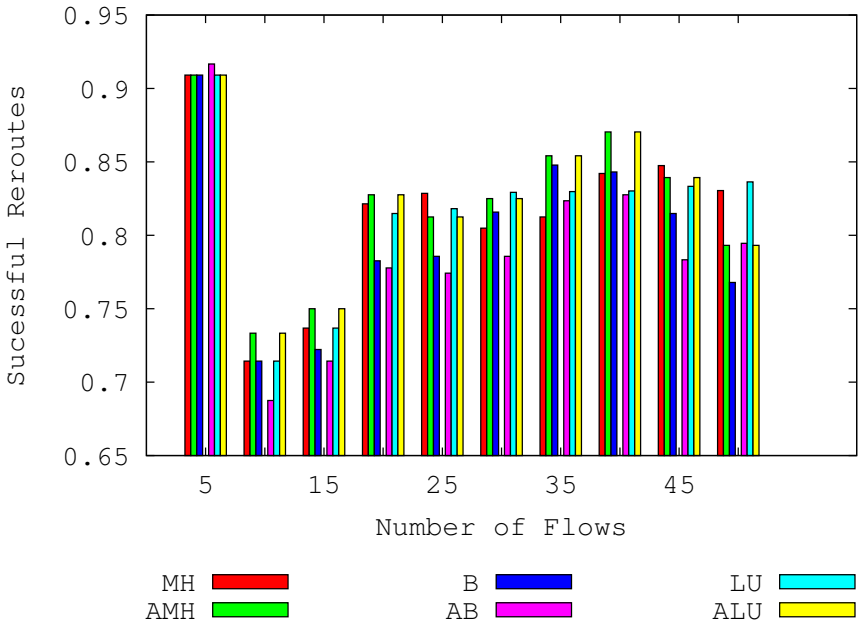


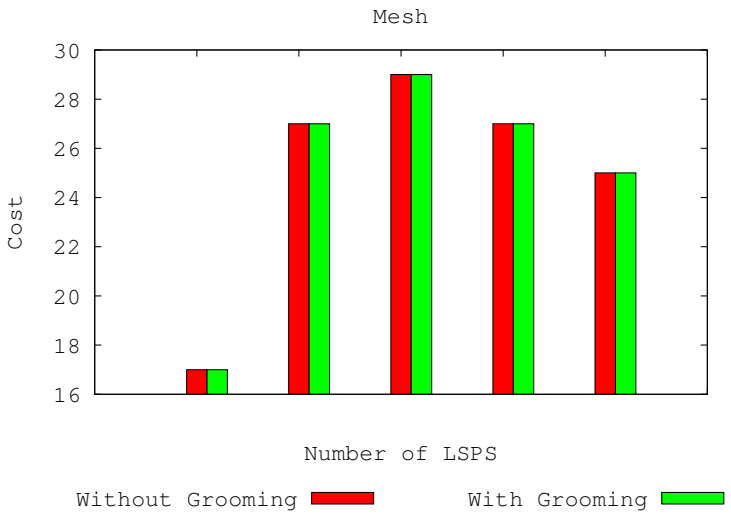
Figure A.40: % of Successful Rerouting in Case of Link Failure For Sul Network

A.3 Chapter 6

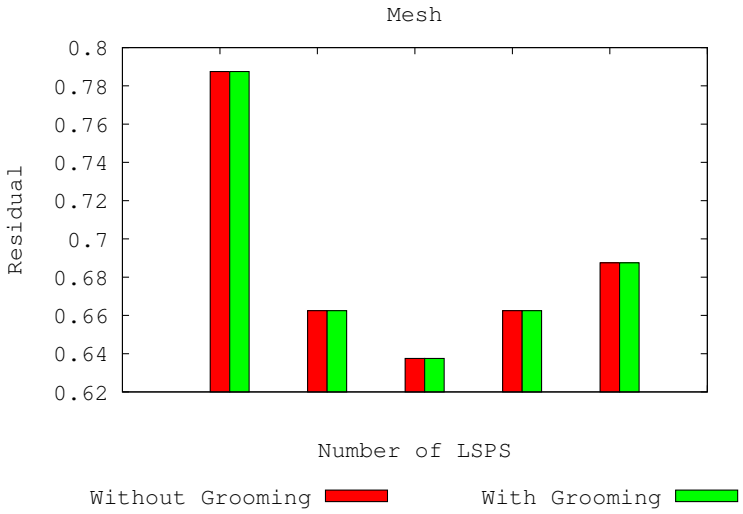
A.3.1 Traffic Changes Detection Scheme

The metric used to compare the benefits of grooming are residual bandwidth and number of hops (cost) and the results are presented for the policies that are adaptive in Figures A.41 - A.55 for all network topologies.

When we vary the policy, we vary the network status in terms of bandwith utilization in each link. For all policies and topologies, the cost decreased and the residual bandwidth increased when the grooming process was used in comparison with the case when no grooming was done, specially in the cases where the network was more congested, i.e. number of flows greater than 30.

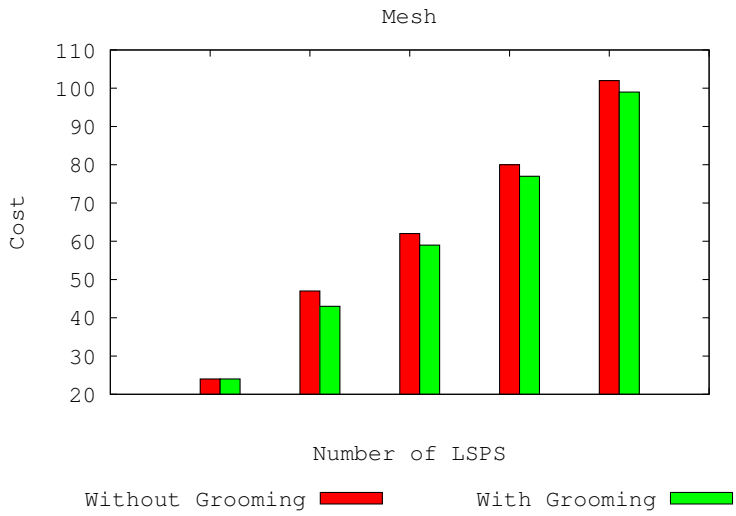


(a) Cost

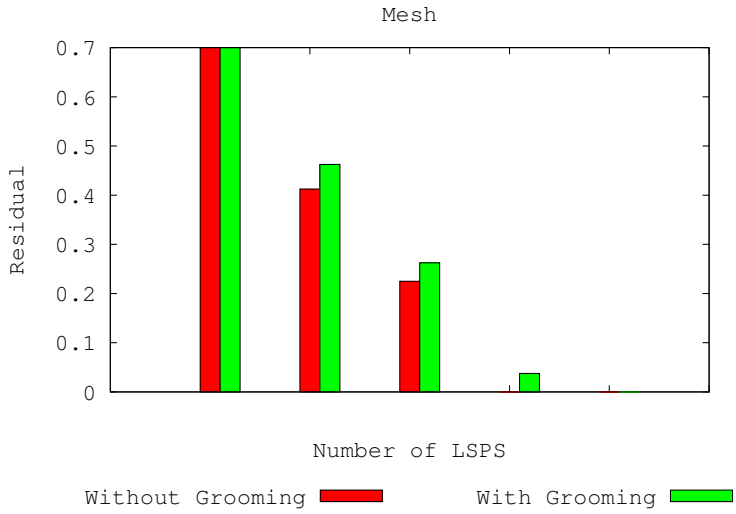


(b) % of Residual Bandwidth

Figure A.41: AMH Grooming Results For Mesh Network

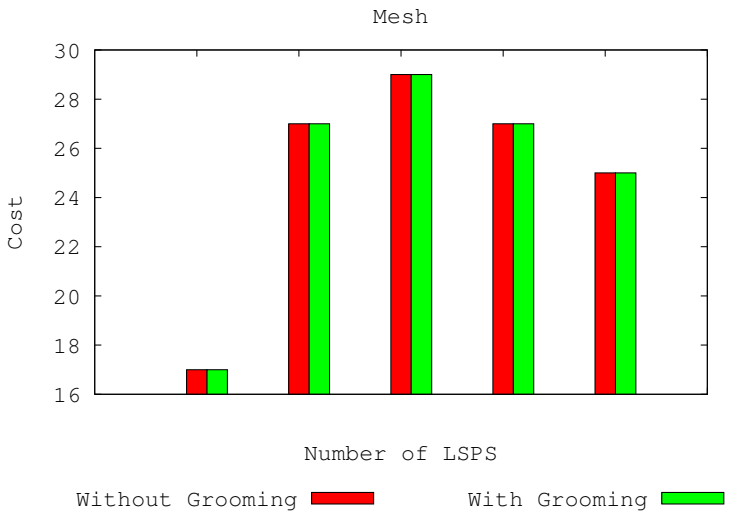


(a) Cost

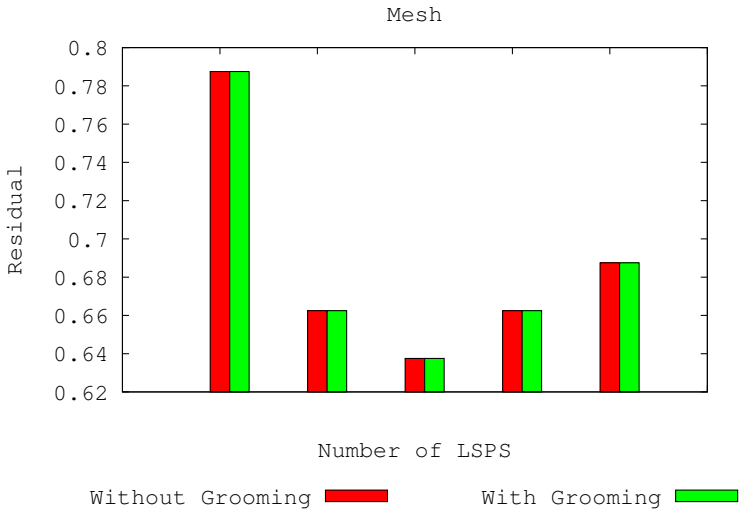


(b) % of Residual Bandwidth

Figure A.42: AB Grooming Results for Mesh Network

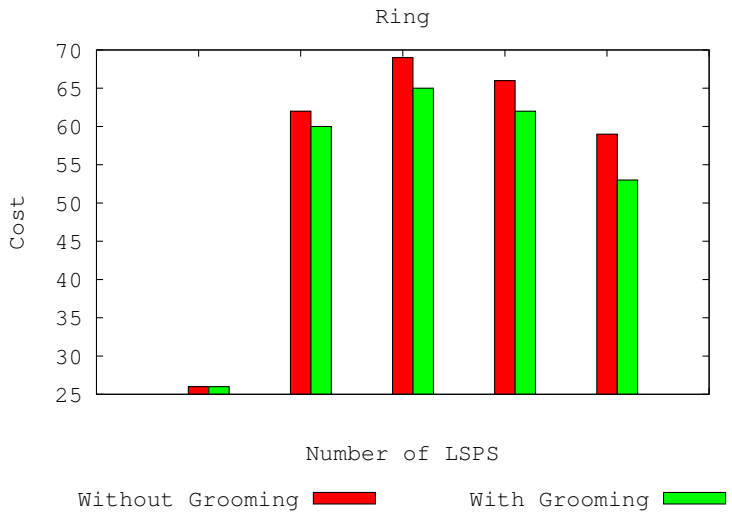


(a) Cost

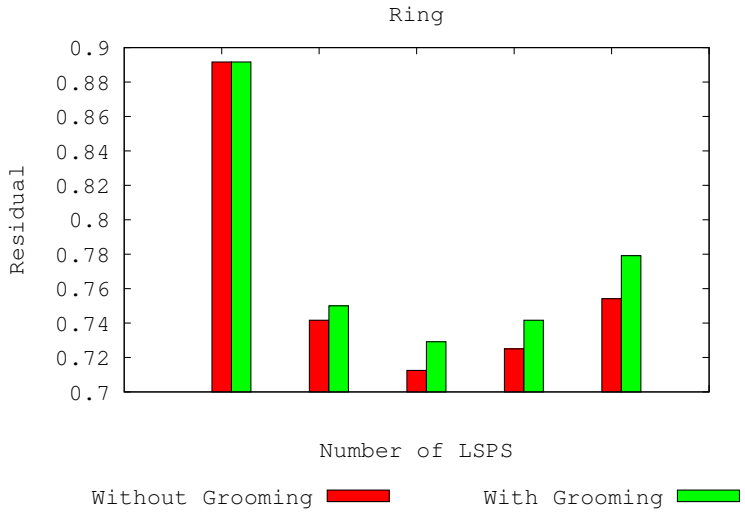


(b) % of Residual Bandwidth

Figure A.43: ALU Grooming Results for Mesh Network

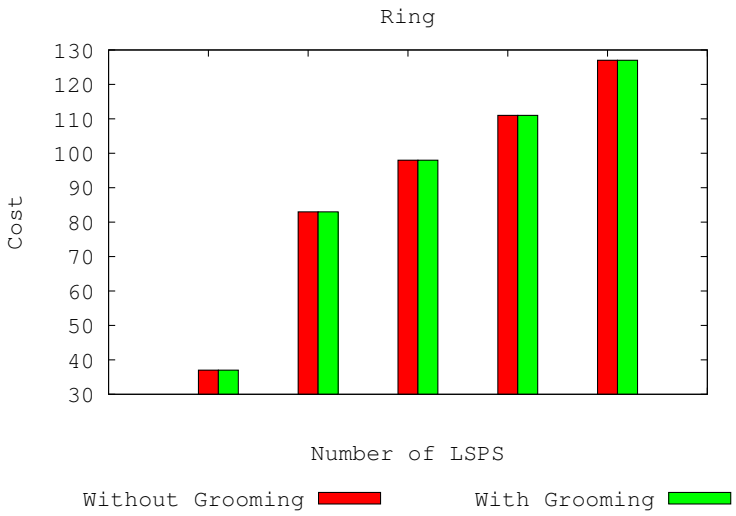


(a) Cost

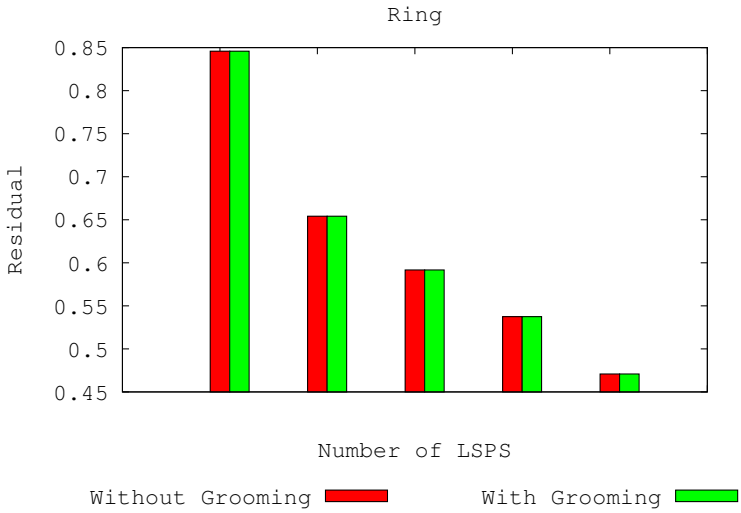


(b) % of Residual Bandwidth

Figure A.44: AMH Grooming Results for Ring Network

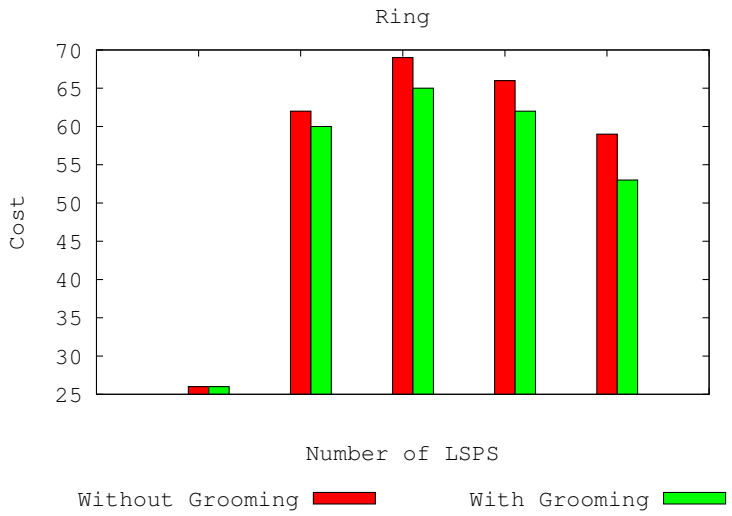


(a) Cost

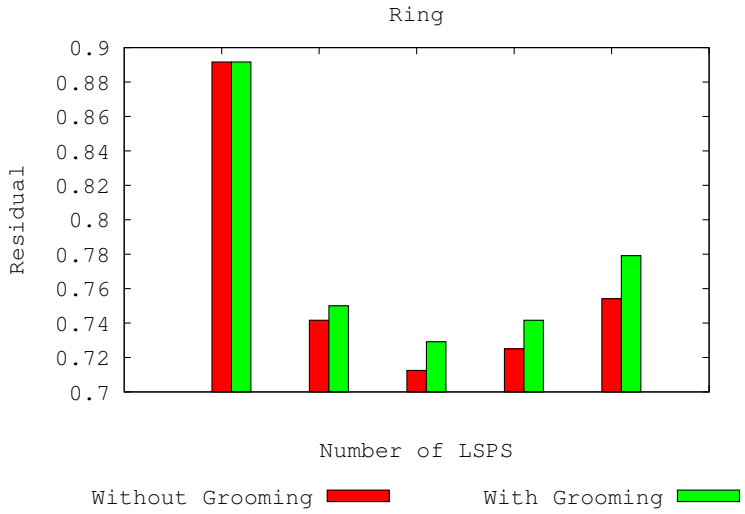


(b) % of Residual Bandwidth

Figure A.45: AB Grooming Results for Ring Network

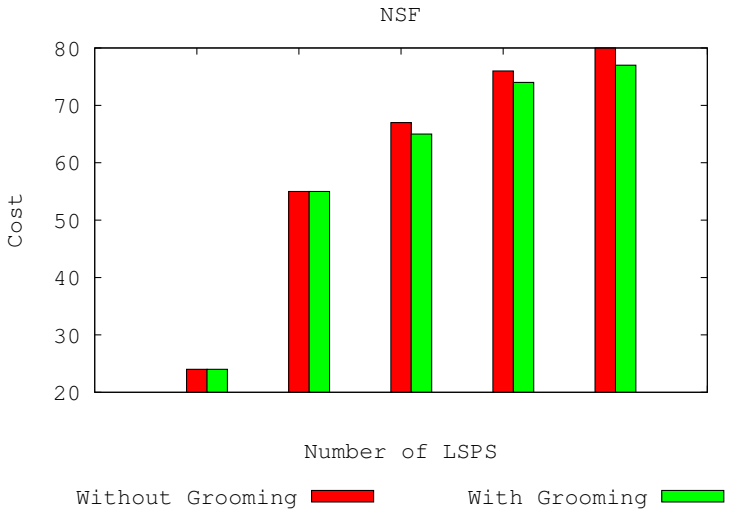


(a) Cost

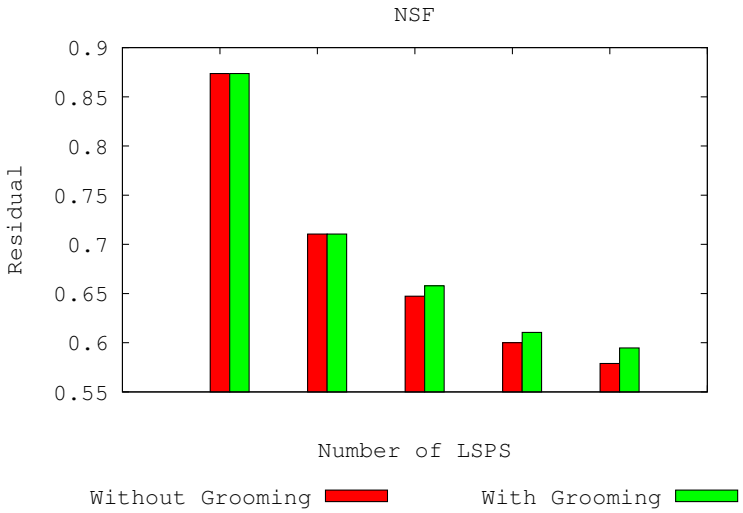


(b) % of Residual Bandwidth

Figure A.46: ALU Grooming Results for Ring Network

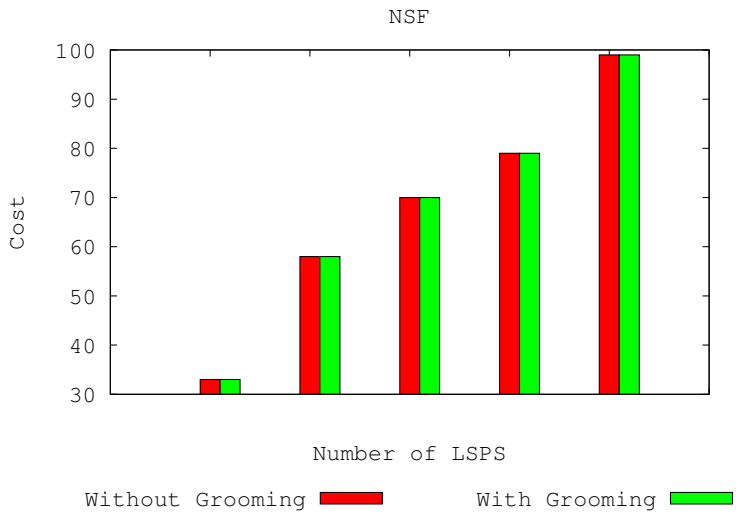


(a) Cost

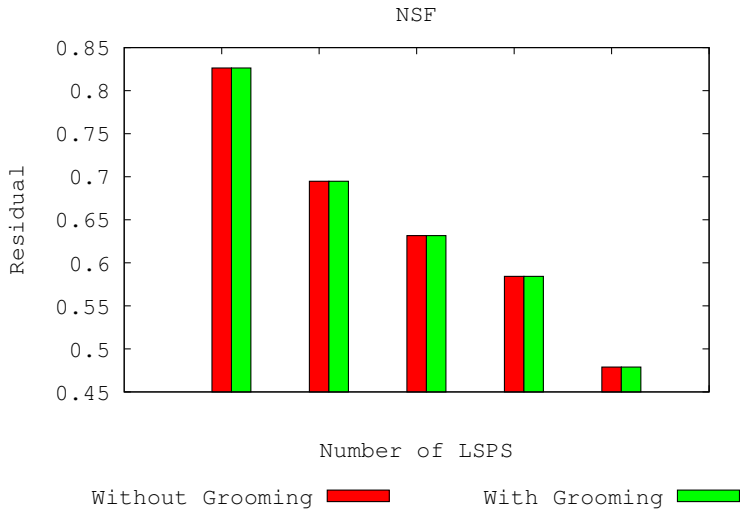


(b) % of Residual Bandwidth

Figure A.47: AMH Grooming Results for NSF Network

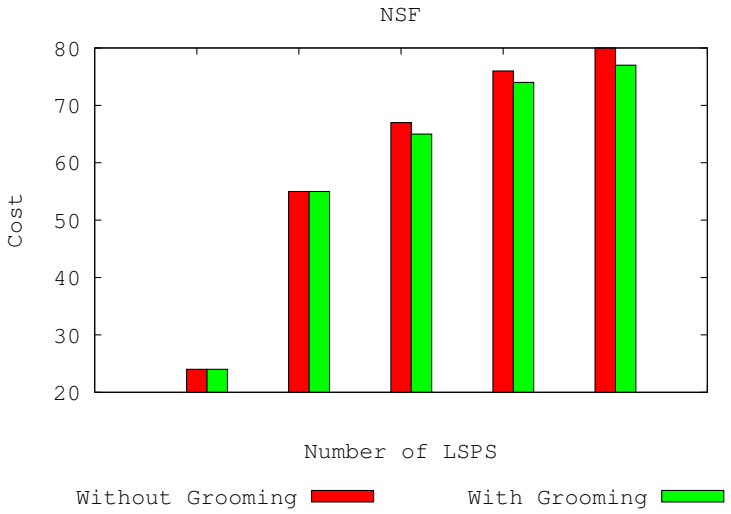


(a) Cost

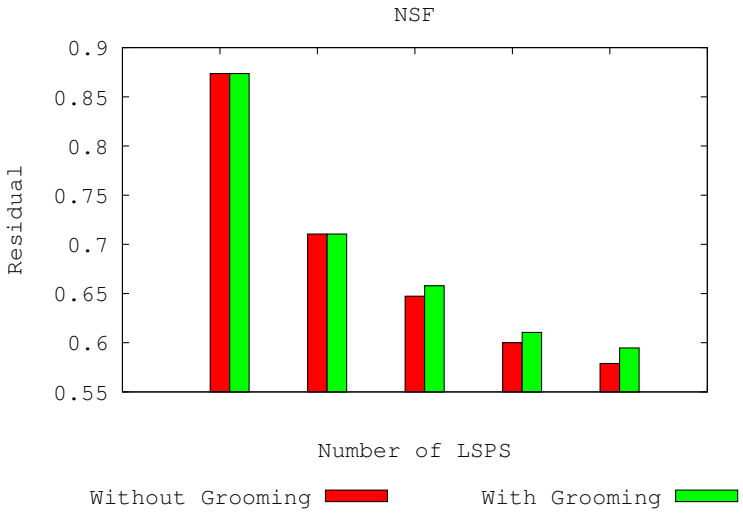


(b) % of Residual Bandwidth

Figure A.48: AB Grooming Results for NSF Network

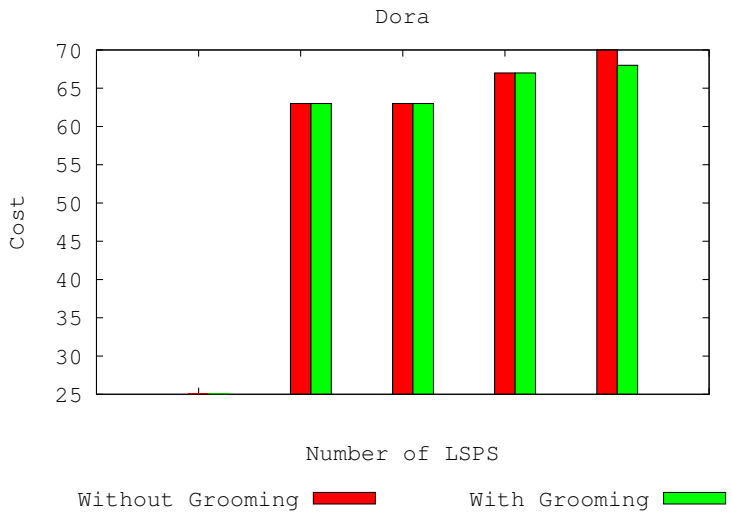


(a) Cost

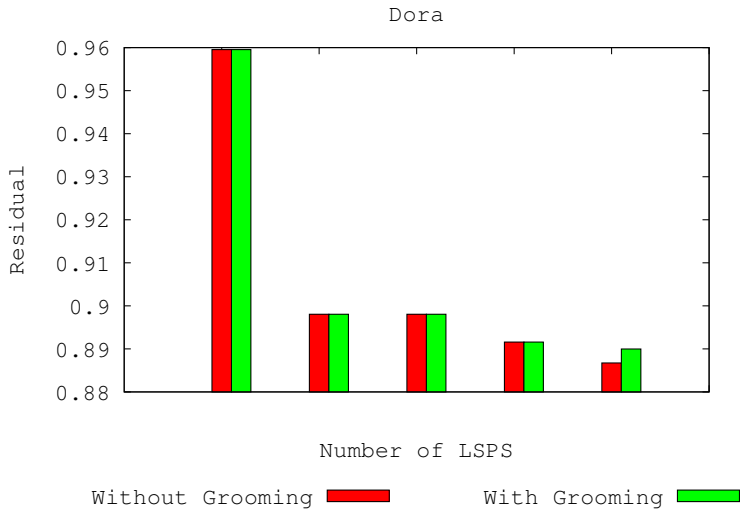


(b) % of Residual Bandwidth

Figure A.49: ALU Grooming Results for NSF Network

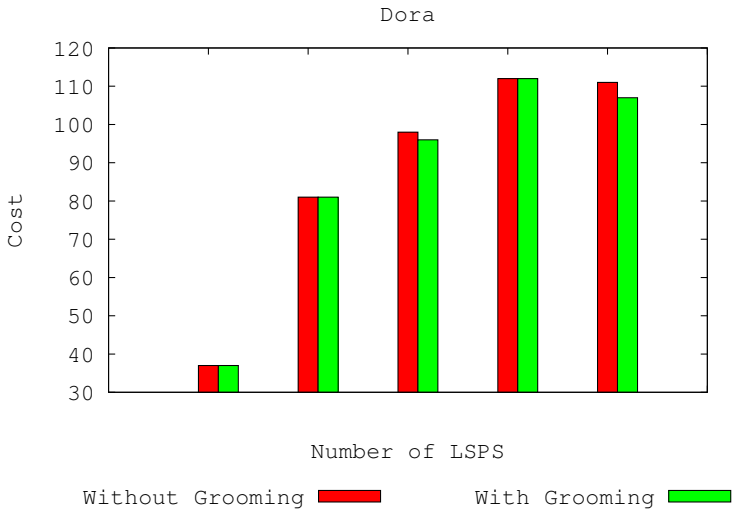


(a) Cost

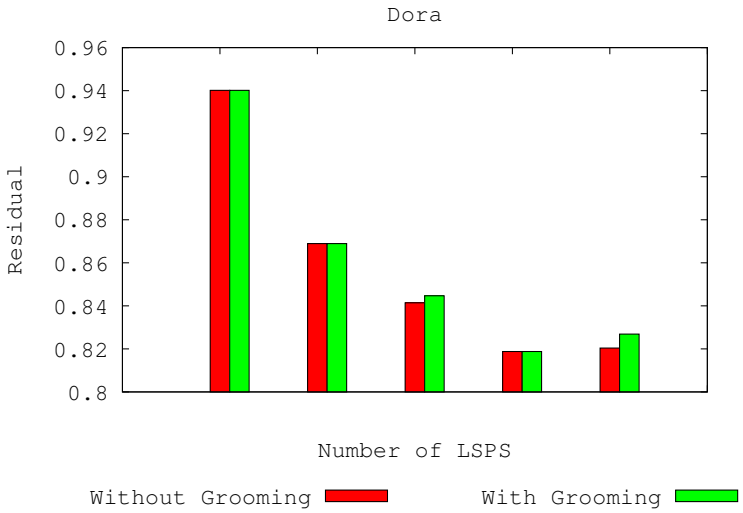


(b) % of Residual Bandwidth

Figure A.50: AMH Grooming Results for Dora Network

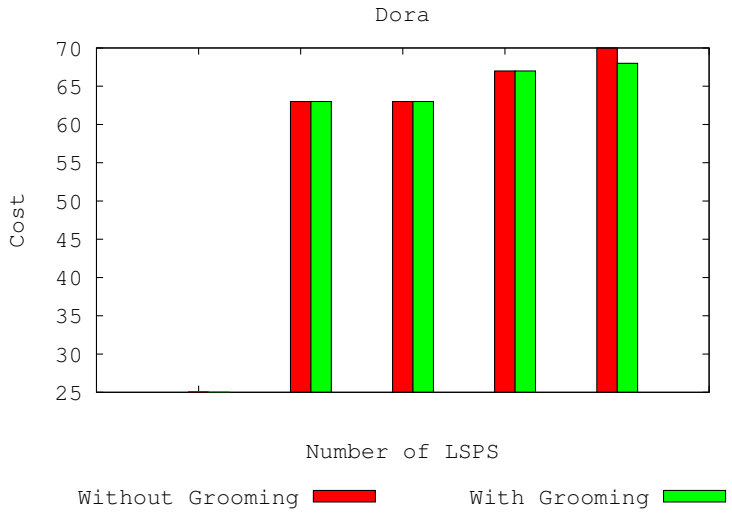


(a) Cost

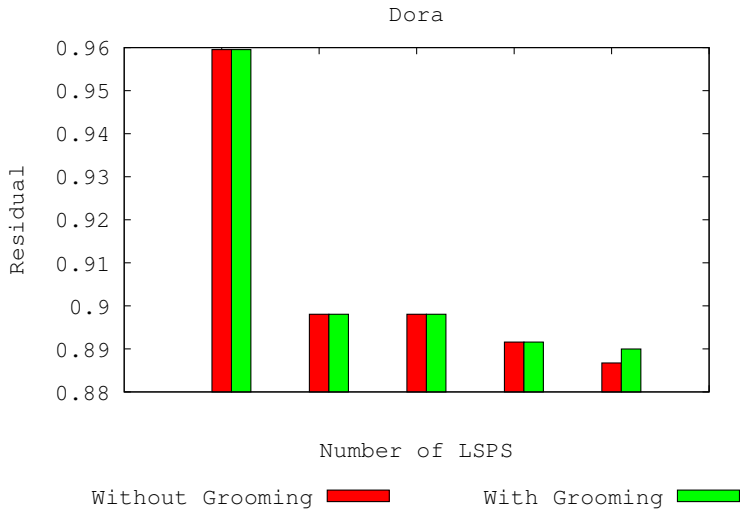


(b) % of Residual Bandwidth

Figure A.51: AB Grooming Results for Dora Network

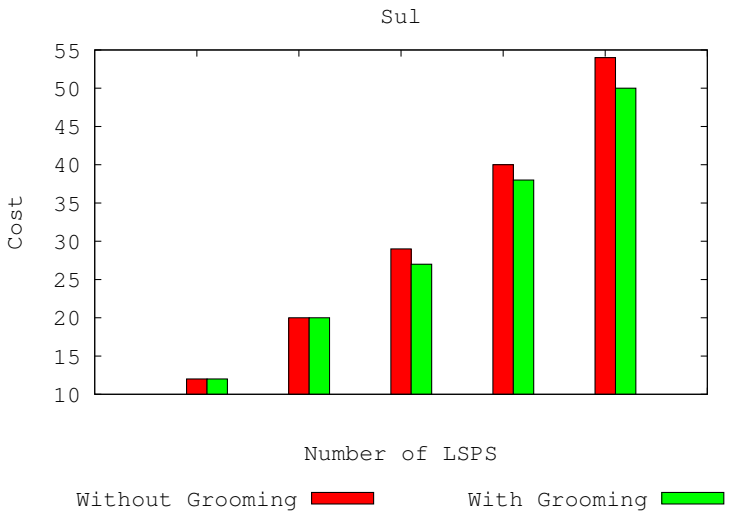


(a) Cost

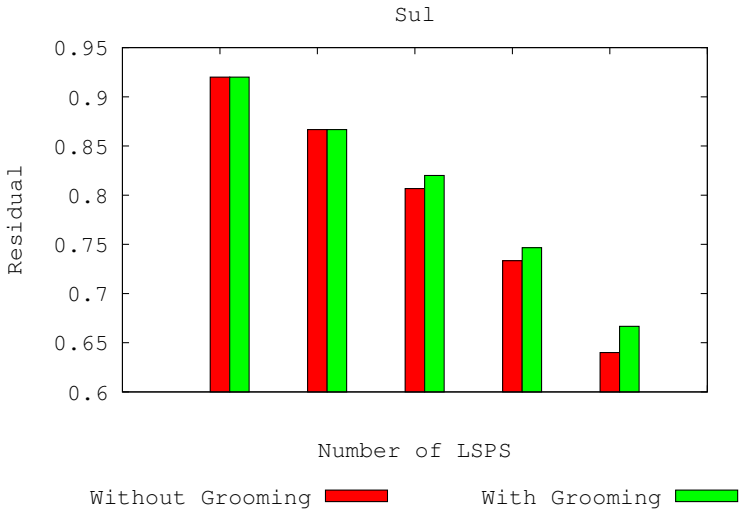


(b) % of Residual Bandwidth

Figure A.52: ALU Grooming Results for Dora Network

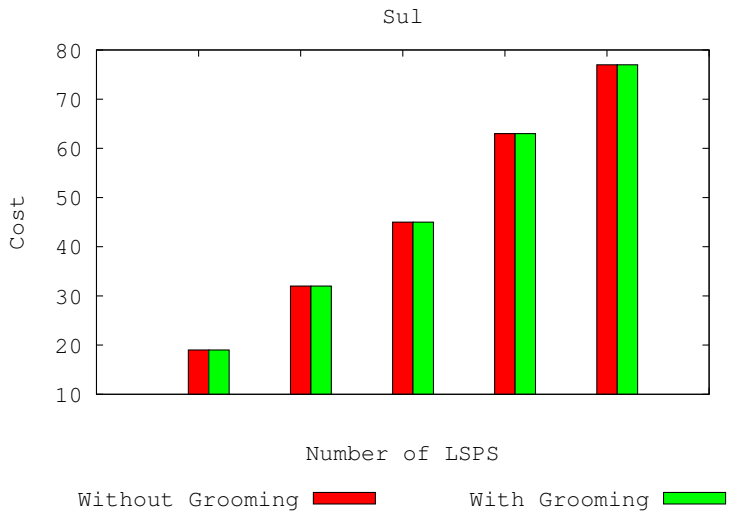


(a) Cost

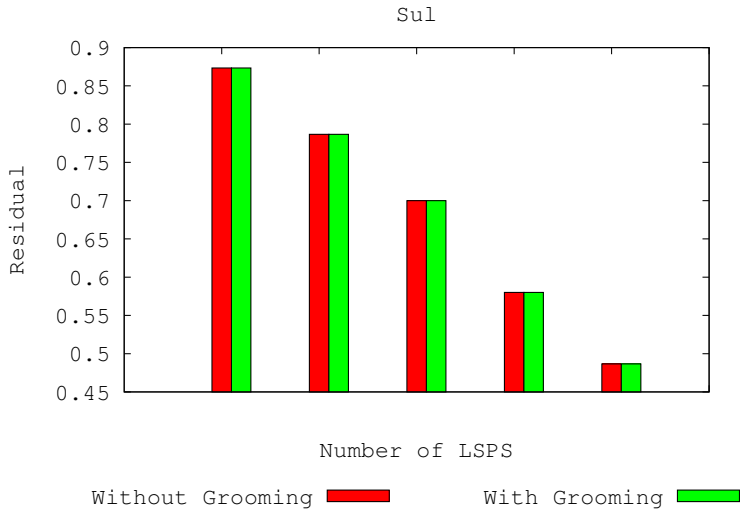


(b) % of Residual Bandwidth

Figure A.53: AMH Grooming Results for Sul Network

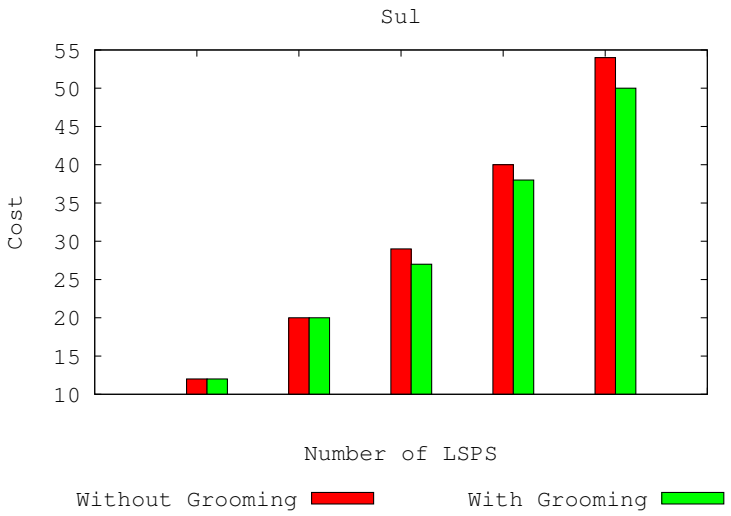


(a) Cost

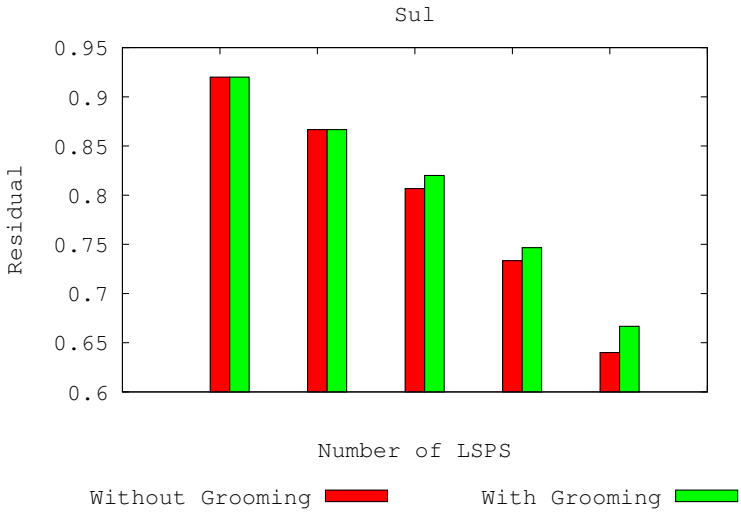


(b) % of Residual Bandwidth

Figure A.54: AB Grooming Results for Sul Network



(a) Cost



(b) % of Residual Bandwidth

Figure A.55: ALU Grooming Results for Sul Network