Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação
Curso Sistemas de Informação

Aplicação de algoritmo genético multiobjetivo na otimização de fluxos multimídia em redes MPLS

Marcelo Bráulio Pedras

Diamantina, Março de 2013

Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação
Curso Sistemas de Informação

Aplicação de algoritmo genético multiobjetivo na otimização de fluxos multimídia em redes MPLS

Marcelo Bráulio Pedras

Monografia submetida à Banca Examinadora designada pelo curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisito para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Alessandro Vivas Andrade

Diamantina, Março de 2013

Marcelo Bráulio Pedras

Aplicação do algoritmo genético multiobjetivo na otimização de fluxos multimídia em redes MPLS

Monografia submetida à Banca Examinadora designada pelo curso de Sistemas de Informação da Universidade Federal dos Vales do Jequitinhonha e Mucuri, como requisito para obtenção do título de Bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

Prof. Dr. Alessandro Vivas Andrade (Orientador)
Universidade Federal dos Vales do Jequitinhonha e Mucuri

Profa. Dra. Luciana Pereira de Assis
Universidade Federal dos Vales do Jequitinhonha e Mucuri

Prof. MSc. Rafael Santin
Universidade Federal dos Vales do Jequitinhonha e Mucuri

Diamantina, Março de 2013

Agradecimentos

Agradeço a Deus por me conceder forças para completar mais essa etapa na minha vida. A minha mãe, Maria Lúcia, meu pai, Rubens Silvestre, meu chefe e amigo Daniel Gomes, meu orientador, Dr. Alessandro Vivas e a todos os amigos e familiares que me apoiaram nessa caminhada.

Resumo

Com a evolução das tecnologias e da utilização da internet, houve um aumento no número de aplicações que utilizam redes, principalmente as aplicações em tempo real, como voz sobre IP e videoconferência. Estes tipos de aplicações são sensíveis ao atraso da comunicação, variação de atraso, perda de pacotes e taxa de transmissão. Essas métricas definem a Qualidade de Serviço (QoS) de uma rede. As redes IP atuais não estão preparadas para atender as especificidades desses tipos de aplicação, uma vez que prestam o serviço de melhor esforço (best-effort). Ou seja, não há garantias que os pacotes chegarão em ordem e dentro das métricas estimadas para que o fluxo obtenha sucesso na transmissão, uma vez que cada pacote segue de forma independente, e a qualidade de envio depende da ocupação atual da rede. Caso alguns enlaces estejam sobrecarregados, o pacote sofrerá um atraso adicional, podendo até ser descartado, prejudicando o desempenho de aplicações multimídia.

Devido a flexibilidade e robustez do protocolo IP, surgiram algumas propostas para que esse fosse reformulado, suportando, também, um serviço que garantisse Qualidade de Serviço. Umas das possíveis abordagens é a utilização da arquitetura MPLS (Multiprotocol Label Switching), pois ela permite que rotas explícitas sejam construídas possibilitando a utilização de Engenharia de Tráfego (TE), para que as rotas atendam as restrições impostas pelos fluxos. O número de critérios escolhidos para a otimização das rotas define, também, a abordagem para a solução do problema de construção das rotas.

Devido a complexidade elevada para a criação de rotas que levem em consideração vários critérios de QoS, uma solução exata baseada em modelos matemáticos não é viável em tempo computacional, sendo necessário o uso de heurísticas. Heurísticas são métodos não exatos que buscam a solução ótima através de uma busca aleatória guiada baseada na teoria das probabilidades. Não há garantias que a solução ótima seja encontrada, embora seja possível aproximar-se dessa, em muitos casos, em um tempo computacionalmente viável. Uma das heurísticas bio-inspiradas mais utilizadas em problemas de otimização é o Algoritmo Genético (AG).

Este trabalho propõe um algoritmo genético multi-objetivo para resolver o problema de definição de rotas e implementa diferentes operadores genéticos afim de avaliá-los em algumas das topologias mais utilizadas em problemas correlatos a fim de verificar quais são os operadores mais adequados para a solução do problema proposto.

Através dos resultados obtidos é possível atestar que os operadores genéticos de seleção, cruzamento e mutação, respectivamente, seleção por ranking e roleta-ranking; cruzamento ortogonal e mutação por balanceamento de carga são superiores aos demais operadores genéticos testados nas topologias Carrier, Dora, Mesh, NFS, Ring e Sul.

Abstract

With the evolution of technology and the use of the internet, there was an increase in the number of applications using networks, especially real-time applications, such as voice over IP and video conferencing. These types of applications are sensitive to delay of communication, delay of variation, packet loss and throughput. These metrics define the Quality of Service (QoS) on a network. The current IP networks are not prepared to meet the specifics of these types of applications, since they provide the best-effort service. That is, there is no guarantee that packets will arrive in order and within the estimated metrics for the flow in order to achieve a successful transmission, since each package follows independently, and the quality of transmission depends on the current occupation of the network. If some links are overloaded, the package will suffer additional delay, and may even be dropped, affecting the performance of multimedia applications.

Due to flexibility and robustness of the IP protocol, there have been some proposals for its reformulation, supporting also a service that guarantees Quality of Service. One possible approach is the use of MPLS architecture (Multiprotocol Label Switching), since it allows explicit routes to be constructed, enabling the use of Traffic Engineering (TE), so that the routes meet the restrictions imposed by the flows. The chosen number of criteria for routes optimization also defines the approach for solving the problem of routes construction.

Due to high complexity of creating routes that take into account multiple QoS criteria, an exact solution based on mathematical models is not feasible in computational time, requiring the use of heuristics. Heuristics are inexact methods that pursuit the optimal solution through a guided random search based on probabilities theory. There is no guarantee that the optimal solution is found, although it is possible to approach this, in many cases, in a computationally feasible time. One of the most bio-inspired heuristics used in optimization problems is the Genetic Algorithm (GA).

This paper proposes a multi-objective genetic algorithm to solve the problem of defining routes and implements different genetic operators in order to evaluate them in some of the

most commonly used topologies on related problems, in order to check which operators are more suitable for the solution of the proposed problem.

Through the results it is possible to say that the genetic operators of selection, crossover and mutation, respectively, ranking selection and ranking-roulette; orthogonal crossover and mutation by load balancing are superior to other genetic operators tested in the Carrier, Dora, Mesh, NFS, Ring and Sul topologies.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

As primeiras aplicações que surgiram com a Internet foram a transferência de arquivos, e-mail e acesso remoto. Estas aplicações usavam o Internet Protocol (IP), sem garantias, provendo o serviço chamado de *best effort* (melhor esforço), em conjunto com o *Transmission Control Protocol* (TCP), possuindo mecanismos de retransmissão e controle de congestionamento para tratar a falta de confiabilidade não provida pelo protocolo IP. Nos últimos anos, novas aplicações surgiram, como videoconferência e voz sobre IP. Essas aplicações demandam maior largura de banda, além de métricas mais restritas; como menor variação de atraso, atraso total, taxa de transmissão e perda de pacotes. Embora os mecanismos de retransmissão e controle de congestionamento proporcionem confiabilidade ao protocolo TCP [Postel (1981b)], esses mecanismos introduzem obstáculos adicionais às aplicações em tempo real.

Assim, com a evolução da tecnologia e da utilização da internet, houve um aumento no número de aplicações que operam em redes, principalmente as aplicações em tempo real. Esse tipo de aplicação usa um conceito importante chamado SLA (*Service Level Agreement* ou Acordo de Nível de Serviço). Para definir esse acordo se usa métricas como: atraso da comunicação, perda de pacotes e disponibilidade de serviços. Surge então o conceito de QoS (Qualidade de Serviço), definido como qualidade da comunicação ou grau de satisfação de um usuário do serviço [Maia (2006)]. O termo Qualidade de Serviço surge nesse contexto a fim de quantificar e/ou qualificar a necessidade de cada tipo de fluxo em termos como atraso total, variação de atraso, taxa de transmissão e percentual de erros. Várias propostas surgiram a fim de garantir Qualidade de Serviço, como Serviços Integrados (IntServ/RSVP) [Shenker and Wroclawski (1997)], Serviços Diferenciados [Blake (1998)] e o *Multiprotocol Label Switching* (MPLS) [Rosen et al. (2001)].

A arquitetura MPLS, proposta por [Rosen et al. (2001)], introduz nas redes TCP/IP a capacidade de roteamento explícito, tornando possível a criação de caminhos explícitos,

possibilitando que sejam estabelecidas rotas que passem por enlaces com recursos suficientes, chamadas de LSPs (*Label Switching Path*).

Neste contexto, a TE tem como objetivo avaliar e otimizar o desempenho da rede a fim de garantir que a rotas escolhidas atendam as restrições de QoS impostas pelos fluxos. As rotas podem ser construídas utilizando vários critérios, os principais são: custo operacional, número de nós intermediários, taxa de transmissão, confiabilidade, atraso fim-a-fim e variação de atraso [Xiao and Ni (1999)]. O número de critérios escolhidos para a otimização das rotas define, também, a abordagem para a solução do problema de construção das rotas.

Em abordagens mono-objetivo, um critério é escolhido para ser a função objetivo e os demais são modelados como restrições, resultando em uma única solução. Em abordagens multi-objetivo, é possível considerar cada critério como uma função objetivo, otimizando-as simultaneamente. A abordagem multi-objetivo possui maior flexibilidade, uma vez que resulta em um conjunto de soluções, chamado Pareto-ótimo, possibilitando a escolha de uma solução baseada no estado da rede num determinado momento, além de melhorar o desempenho da rede [Maia (2006)]. Ou seja, alterando o procedimento de tomada de decisão é possível obter um solução distinta sem a necessidade de executar novamente o procedimento de otimização. Porém, otimizar várias rotas sujeitas a mais de duas restrições de QoS não é uma tarefa trivial, o qual é provado ser NP-Completo em [Shao et al. (2006)].

Devido a complexidade elevada para a criação de rotas que levem em consideração vários critérios de QoS, uma solução exata baseada em modelos matemáticos não é viável em tempo computacional, sendo necessário o uso de heurísticas. Heurísticas são métodos não exatos que buscam a solução ótima através de uma busca aleatória guiada. Não há garantias que a solução ótima seja encontrada, embora seja possível aproximar-se dessa, em muitos casos, em um tempo computacionalmente viável. Uma das heurísticas bio-inspiradas mais utilizadas em problemas de otimização é o Algoritmo Genético (AG).

Algoritmos genéticos são métodos de busca e otimização combinatória que usam como princípio a teoria da seleção natural das espécies desenvolvidas por Charles Robert Darwin [Darwin (1859)]. Um AG é definido como uma técnica de busca aleatória dirigida desenvolvida por [Holland (1975)]. Esses possuem a vantagem de trabalhar com vários pontos no espaço de busca multi-dimensional, aumentando a possibilidade de incluir a solução ótima global. Por meio de uma população inicial o AG evolui utilizando operações sucessivas de seleção, cruzamento e mutação, até que o critério de parada seja alcançado.

As técnicas de otimização utilizadas em um domínio MPLS podem ser classificadas em estática e dinâmica. Na estática, o estado da rede e as requisições a serem alocadas são conhecidas e otimização baseia-se em alcançar os níveis de QoS pré-determinados. Na

dinâmica, não existe conhecimento prévio, as requisições devem ser alocadas ao longo do tempo, considerando o estado atual da rede.

Este trabalho utiliza um algoritmo genético multi-objetivo em um ambiente MPLS estático a fim de obter um conjunto de rotas que atendam as requisições, simultaneamente, minimizando o número de rejeições, enlaces utilizados e a variação de carga nos enlaces, respeitando as restrições de capacidade dos links da rede.

Capítulo 2

Referencial Teórico

2.1 Conhecimentos Básicos

Este capítulo descreve alguns dos conceitos necessários para melhor compreensão desta monografia. Assim serão abordados assuntos como estrutura e protocolos usados na Internet, bem como algoritmos e estratégias de roteamento para alcançar Qualidade de Serviço. Variações de Algoritmos Genéticos, AGs, também serão descritas tal como algumas implementações de seus operadores genéticos e a interferência dos parâmetros no comportamento do algoritmo.

2.1.1 TCP/IP

Na metade da década de 60, os mainframes disponíveis para as organizações de pesquisa trabalhavam de forma isolada. Computadores de fabricantes diferentes eram incapazes de se comunicar. A Advanced Reserch Project Agency (ARPA), agência do departamento de defesa dos Estados Unidos, procurava uma maneira de interconectar esses computadores para que os pesquisadores pudessem compartilhar suas pesquisar, reduzir custos e retrabalho [Forouzan (2006)].

Em 1967, a ARPA apresentou a ideia para uma pequena rede, a ARPANET. A ideia era que computadores de diferentes fabricantes pudessem se conectar a um computador específico denominado Interface Message Processor (IMP) que, por sua vez, comunicavam entre si. Em 1969, ARPANET foi consolidada através da ligação entre quatro faculdades americanas.

Em 1973, Vint Cerf e Bob Kahn [G.Cerf and Kahn (1974)] publicaram um artigo que estabelecia um protocolo para promover a entrega e transmissão de dados (TCP). Pouco tempo depois o TCP foi dividido em dois protocolos. Um responsável roteamento de datagramas,

o Internetworking Protocol (IP) e o outro responsável pela segmentação, reagrupamento e detecção de erros (TCP). A partir de então o modelo passou a ser conhecido com o TCP/IP.

Modelo de referência OSI/ISO

Para reduzir a complexidade do projeto, a maioria das redes é organizada como uma pilha de camadas, onde as mais baixas oferecem serviços às mais altas e se comunicam diretamente apenas com as camadas adjacentes. Um serviço é um conjunto de primitivas que a camada oferece a camada localizada acima dela [Tanenbaum (2000)]. Essa subdivisão é utilizada para ocultar os detalhes de implementação e assim permitir que as camadas possam ser implementadas independentemente uma das outras, sendo necessário manter apenas a interface comum. Um protocolo é um conjunto de regras que controla o formato e o significado dos pacotes ou mensagens que são trocadas pelas entidades pares contidas em uma camada [Tanenbaum (2000)]. Um conjunto de protocolos e camadas é denominado de arquitetura de rede e um o conjunto de protocolos, um por camada, é chamado de pilha de protocolos.

Para que fosse possível a comunicação entre redes diferentes era necessária adotação de um padrão internacional. O primeiro passo para a padronização foi dado pela International Standards Organization (ISO) ao propor o modelo de referência chamado de Open Systens Interconnection (OSI). A Figura 2.1 exibe a disposição em camadas do modelo OSI.

O modelo OSI contém sete camadas:

1. Camada Física: trata da transmissão de bits em meio físico.
2. Camada de enlace de dados: trata da abstração de um canal de dados ruidoso em um canal livre de erros definindo a forma de organização de dados em quadros e como transmiti-los
3. Camada de rede: trata da transmissão de pacotes fim a fim determinando como são atribuídos os endereços e como os pacotes são enviados até o destino final.
4. Camada de transporte: responsável por tratar os detalhes de transferência confiável.
5. Camada de sessão: especifica como estabelecer um sessão de comunicação em sistema remoto levando em consideração detalhes de segurança, como autenticação, e exclusão mútua no caso de recursos compartilhados.
6. Camada de apresentação: responsável por manter a compatibilidade da representação binária de máquinas diferentes.

7. Camada de aplicação: especifica como um aplicativo interage com a rede bem como a resposta esperada do host conectado na outra ponta.

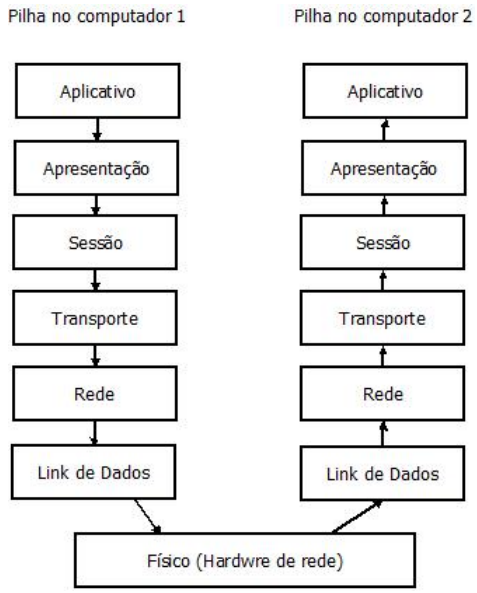

Figura 2.1: Modelo de referência OSI/ISO

No mesmo momento em que o modelo OSI era padronizado o uso do modelo TCP/IP crescia drasticamente e muitas empresas passaram a investir em hardware compatível com esse. Além disso, o modelo OSI, segundo Tanenbaum [Tanenbaum (2000)], foi padronizado prematuramente, antes de alcançar um número suficiente de pesquisas que pudessem consolidá-lo, o que resultou em implementações ruins deviso sua complexidade.

Modelo de camadas da Internet

O modelo OSI foi desenvolvido antes da ligação inter-redes ser inventada e consequentemente não contem uma camada para essa finalidade. Possui também uma camada de sessão a qual perdeu importância com a mudança dos grandes sistemas de tempo compartilhado para workstations pessoais [Comer (2007)]. Apesar dos problemas do modelo OSI, é indiscutível que ele foi fundamental para o desenvolvimento das atuais arquiteturas de redes. Em resposta aos itens citados acima, um novo modelo foi desenvolvido, o Modelo de Camadas Inter-redes ou simplesmente TCP/IP. Esse modelo possui cinco camadas. A seguir essas camadas são definidas bem como suas funcionalidades. A Figura 2.2 exibe as diferenças entre as camadas do modelo OSI e TCP/IP.

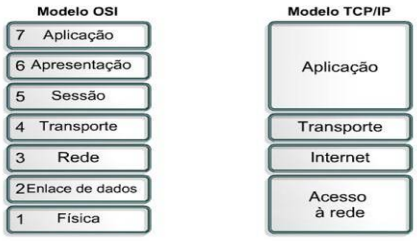

Figura 2.2: Modelo ISO x TCP IP

Camada Física

Sua principal função é fazer a interface do modelo TCP/IP com os diversos tipos de redes como: Ethernet, X.25, ATM, Frame Relay entre outras. Devido a grande variedades de tecnologias para redes essa camada não é normatizada pelo modelo. Isso permite que redes heterogêneas possam se comunicar, sendo uma das razões da flexibilidade do modelo TCP/IP.

Camada de Enlace de Dados

Como no modelo OSI, a camada de enlace de dados é responsável por movimentar datagramas da camada de rede nó a nó. Essa camada define o protocolo de enlace de dados que pode definir uma entrega confiável; com políticas de detecção e correção de erros, confirmação e retransmissão; ou não confiável, sem tais políticas [Kurose and Ross (2006)]. Normalmente os protocolos de enlace de dados que trabalham sobre redes de baixa taxa de erros; como enlaces de fibra, cabo coaxial e par trançado; não definem uma entrega confiável para não incluir mais overhead a rede. Isso porque a camada de transporte define políticas de entrega confiável, como no caso do TCP. Se considerarmos o tipo de fluxo transportado na rede, como voz em tempo real, tipicamente contínuo, as estratégias de correção de erros e retransmissão causam atrasos, que, por sua vez, causam mais problemas do que simplesmente descartar o quadro. Porém a política de entrega confiável na camada de enlace é essencial em enlaces de alta taxa de erros, como nos enlaces sem fio. Outra funcionalidade da camada de transporte que pode ser implementada na camada de enlace de dados é o controle de fluxo usado para evitar a sobrecarga do receptor pelo nó remetente. Nesse nível, o controle de fluxo é se torna mais eficiente. Outra característica da camada de enlace de dados é a possibilidade de que cada enlace use um protocolo diferente para manipular o datagrama, como Ethernet no primeiro e outro no enlace final. A Figura 2.3 retirada de [Kurose and Ross (2006)] ilustra a função da camada de Enlace de Dados.

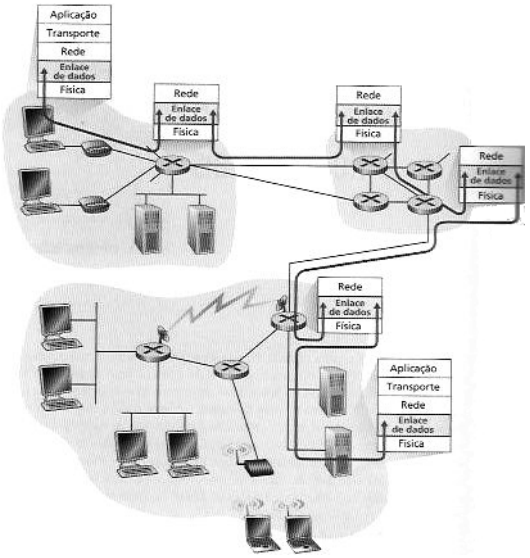

Figura 2.3: Comunicação entre camadas - Camada de enlace de dados

Camada de Rede

A camada de rede é responsável pelo repasse e roteamento de datagramas. Entende-se por repasse o encaminhamento do pacote do enlace de entrada para o enlace de saída, ou seja, envolve apenas um roteador. Já o roteamento envolve todos os roteadores, onde o protocolo de roteamento define as rotas tomadas por cada pacote a fim de que esse alcança seu destino final. Cabe a camada de rede escolher os caminhos mais apropriados para rotear os pacotes a fim de não sobrecarregar os enlaces enquanto deixa outras linhas ociosas. Os serviços oferecidos à camada de transporte levam em consideração os seguintes objetivos, segundo [Tanenbaum (2000)]:

1. Independência da tecnologia de roteadores.
2. Isolar a camada de transporte do número, tipo e topologia dos roteadores.
3. Os endereços de rede devem usar um sistema de numeração uniforme para LANs e WANs.

As premissas citadas acima não restringem a forma de roteamento, sendo possível utilizar um serviço orientado a conexão e sem conexão. Os defensores do serviço orientado a conexão argumentam que a Qualidade de Serviço é um fator dominante para aplicações em tempo real, e que um serviço sem conexão tornaria muito difícil provê-la. Os defensores do serviço sem conexão argumentam que já existem mecanismos de ordenação, tratamento de

Tabela 2.1: Comparação entre sub-redes de datagrama e circuitos virtuais

Questão	**Sub-rede de datagramas**	**Sub-rede de circuitos virtuais**
Configuração de circuitos	Desnecessária	Obrigatória
Endereçamento	Cada pacote contém os endereços de origem e de destino completos	Cada pacote contém um número de circuito virtual curto
Informações sobre o estado	Os roteadores não armazenam informações sobre o estado das conexões	Cada circuito virtual requer espaço em tabelas de roteadores por conexão
Roteamento	Cada pacote é roteado independentemente	A rota é escolhida quando o circuito virtual é estabelecido; todos os pacotes seguem essa rota
Efeito de falhas no roteador	Nenhum, com exceção dos pacotes perdidos durante a falha	Todos os circuitos virtuais que tiverem passado pelo roteador que apresentou o defeito serão encerrados
Qualidade de serviço	Difícil	Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual
Controle de congestionamento	Difícil	Fácil, se for possível alocar recursos suficientes com antecedência para cada circuito virtual

erros e controle de fluxo, tornando desnecessário criar um serviço orientado a conexão. Embora a maioria das redes não sejam orientadas a conexão, essas utilizam alguns conceitos dessas, como o estabelecimento de sessões. A Tabela 2.1 exibe uma comparação entre as sub-redes de datagrama e de circuitos virtuais retirada do livro de [Tanenbaum (2000)].

A Figura 2.4 ilustra o funcionamento do roteamento de um fluxo orientado a conexão e sem conexão.

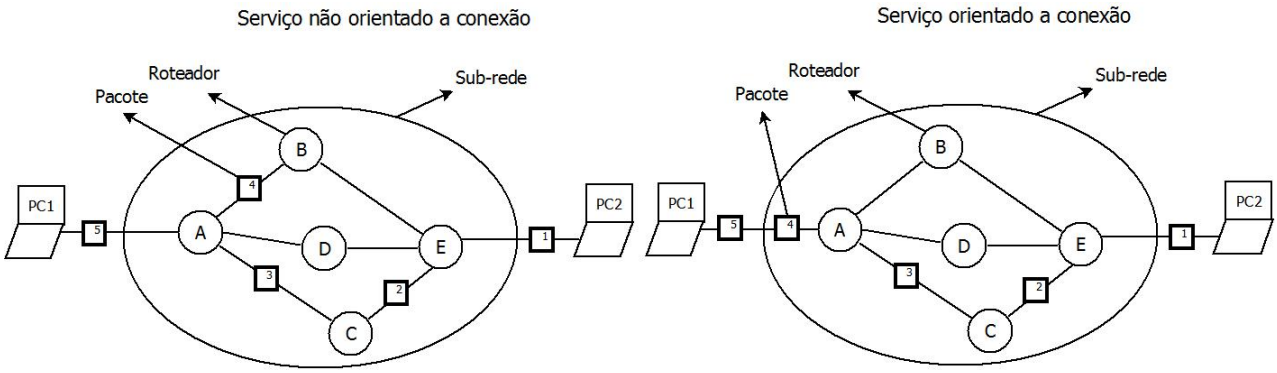

Figura 2.4: Roteamento orientado a conexão e sem conexão

Camada de Transporte

A finalidade da camada de transporte é permitir a conversação entre os hosts de origem e destino permitindo o transporte e a regulação do fluxo. Entre os protocolos desta camada destacam-se o TCP(Transmission Control Protocol) e o UDP(User Datagram Protocol). O TCP busca garantir que os segmentos entregues sejam confirmados e que sejam retransmitidos caso não haja confirmação. Além de prover mecanismos de prevenção, controle de congestionamento e reordenação. O UDP presta serviço fim a fim sem os mecanismos de retransmissão e confirmação citados acima.

Camada de Aplicação

Esta camada contém os protocolos de mais alto nível que interagem diretamente com as aplicações de usuário. O dado é passado do programa de rede no formato interno da aplicação e codificado dentro do padrão do protocolo. Alguns exemplos são: HTTP(Hypertext Transfer Protocol) [Fielding et al. (1999)], FTP(File Transfer Protocol) [Postel and Reynolds (1985)],DNS(Domain Name System) [Mockapetris (1987)] entre outros.

2.2 Qualidade de Serviço em rede IP para fluxos multimídia

A Internet é caracterizada por um conjunto de sistemas autônomos (SA) interconectados. A comunicação entres esses é possível pela estabelecimento de um protocolo de comunicação, no caso, o Internet Protocol (IP) [Postel (1981a)]. O IP é um protocolo da camada de rede e é responsável pela comunicação fim-a-fim. Esse realiza o transporte de datagramas sem garantias (best effort).

As aplicações multimídia, comumente áudio e vídeo, necessitam de maior largura de faixa e possuem restrições quanto ao atraso máximo, variação de atraso e percentual de perda de pacotes, demonstrando, então, maior dificuldade no transporte do que aplicações de dados. As aplicações que possuem tais restrições são chamadas de aplicações em tempo real [Andrade (2008)]. O protocolo IP não diferencia o tipo de informação transportado, dando o mesmo tratamento a fluxos com características diferentes. Em aplicações em tempo real o sucesso da transmissão está relacionado ao intervalo de tempo de entrega do pacote ao destinatário. Caso as restrições não sejam atendidas o pacote pode ser considerado perdido. As aplicações de tempo real podem ser classificadas em hard e soft real time. As hard real-time não toleram falhas e as soft real-time toleram até determinada taxa de erro. Este trabalho aborda aplicações soft real-time.

A maioria das aplicações multimídia é de taxa constante. Entre as aplicações de fluxo contínuo de áudio e vídeo (tempo real ou armazenado) destacam-se: rádios na Internet, palestras gravadas, telefonia IP e videoconferência. Cada tipo de fluxo possui requisitos específicos e devem ser atendidos de modo particular.

Fluxo pode ser definido como uma sequência de pacotes desde uma origem até um destino [Andrade (2008)]. As necessidades de cada fluxo podem ser definidas por quatro parâmetros: confiabilidade, atraso, flutuação (variação do atraso) e largura de faixa [Tanenbaum (2000)]. Esses parâmetros definem Qualidade de Serviço (QoS).

- A confiabilidade é definida como a taxa máxima de perda de pacotes tolerada pela aplicação. Esta pode ser alcançada pelo uso do protocolo TCP, que possui mecanismos para garantir a entrega dos pacotes, como retransmissão e reordenação. Porém, esses mecanismo introduzem atraso adicional, impossibilitando a garantia de limites mínimos de taxa de transmissão. Para transporte multimídia o protocolo UDP (User Datagrama Protocol) [Postel (1980)] é mais adequado. Esse não possui mecanismos de retransmissão, controle de congestionamento e ordenação de pacotes, prestando um serviço de datagrama não confiável. Devido a tolerância das aplicações multimídia de até 20% de taxa de erro [Andrade (2008)], torna-se viável a utilização desse protocolo. Além disso, técnicas de redundância podem ser utilizadas para diminuir a taxa de perda de pacotes.
- O atraso é definido como tempo gasto até que o pacote chegue ao destino. Ele é formado pela soma do atraso de enfileiramento dos pacotes no roteador, propagação e processamento.
- A flutuação ou variação do atraso é definida como a variação do atraso em transmissões sucessivas.

- Largura de faixa é a capacidade de vazão do meio por segundo.

Conclui-se, então, que as redes não estão preparadas para o transporte de aplicações multimídia, já que essas disponibilizam o serviço de melhor esforço, (best effort), ou seja, não a quaisquer garantias, como: de entrega, ordem dos pacotes, taxa de transmissão, atraso máximo ou variação de atraso.

2.3 Algoritmos de Roteamento

Um algoritmo de roteamento tem por finalidade descobrir uma rota otimizada entre o roteador fonte e o roteador de destino. Rota otimizada significa normalmente menor custo, mas a qualidade da rota depende muito dos objetivos do administrador da rede [Kurose and Ross (2006)]. Essas rotas deveriam seguir os princípios de: corretude, simplicidade, robustez, estabilidade, clareza e otimalidade. Implementar todos os princípios pode não ser possível devido a complexidade.

Os algoritmos de roteamento podem ser classificados em adaptativos e não-adaptativos. Os primeiros usam informações da rede, como tráfego e topologia para indicar o melhor roteamento. Já os não-adaptativos usam noções pré-definidas da rede. Os algoritmos de roteamento ainda podem ser classificados em estáticos e dinâmicos. Os primeiros pressupõem que todas as requisições são conhecidas e não haverá mudança no estado da rede. Já os dinâmicos tratam requisições que podem surgir a qualquer momento e suportam alterações na topologia. Podem ainda ser classificados em algoritmos centralizados (globais) ou descentralizados. Os primeiros necessitam do conhecimento de todas as características da rede e então as rotas são calculadas em um determinado nó (centralizado), com a possibilidade de duplicação (outros nós de processamento). Nos algoritmos decentralizados o cálculo do menor caminho é feito de maneira iterativa e descentralizada. Cada nó possui informações incompletas e a troca de informações entre nós vizinhos continuamente determina a visão global da rede [Kurose and Ross (2006)].

Para a definição das rotas em si, a estratégia mais difundida é a do caminho mínimo. Nessa, a rede é interpretada como um grafo onde os roteadores são vértices e as linhas de comunicação são as arestas. Os algoritmos mais conhecidos para implementá-la são o de Dijkstra [Dijkstra (1959)], que calcula o caminho mínimo entre dois vértices e o de Floyd [Floyd (1962)] que calcula o caminho para todos todos os vértices utilizando programação dinâmica. Embora escolher o caminho mínimo pareça simples, a escolha dos custos das arestas não é. Algumas das métricas utilizadas para calcular os custos são:

- número de saltos,
- distância euclediana,
- atraso médio de cada link,
- banda disponível,
- combinação dessas métricas.

Outra estratégia é o algoritmo de inundação. Esse encaminha o pacote para todos os demais roteadores, menos para o que enviou o pacote. Isso garante que o destino será alcançado, caso exista um caminho até esse. Essa estratégia privilegia a propriedade de robustez ao custo de sobrecarregar a rede, além de criar muitas cópias, exigindo um sistema para descarte de pacotes duplicados.

Um dos algoritmos utilizados para roteamento na Internet é o Vetor de Distâncias [Tanenbaum (2000)]. Esse calcula o menor caminho por meio da troca de informações entre os roteadores vizinhos através da tabela de encaminhamento. A métrica mais utilizada para esse calculo é o número de saltos, alternativamente também se usa o atraso, tamanho da fila de encaminhamento e distância física entre roteadores. A taxa de atualização das rotas deve ser pequena o suficiente para refletir as mudanças de estado da rede e grande o suficiente para não causar um overhead devido ao envio de mensagens de controle. Um problema desse algoritmo é a baixa taxa de convergência, que pode deixar a representação da rede diferente da realidade, prejudicando o roteamento. Para solucionar o problema da baixa taxa de convergência foi proposto o algoritmo chamado Estado de Enlace [Tanenbaum (2000)]. Cada roteador aprende sobre o endereçamento e custo para alcançar os vizinhos. A informação é então interpretada e enviada a todos os roteadores da rede por broadcast. Esse passo acelera a convergência. Com informações disponíveis de todos os roteadores é feito o calculo do menor caminho para todos os roteadores. Normalmente o algoritmo de [Dijkstra (1959)] é usado para essa finalidade.

2.4 Abordagens para o oferecimento de QoS em redes IP

Os mecanismos de roteamento do protocolo IP provem a comunicação baseados no melhor aproveitamento dos recursos da rede, utilizando, como uma das políticas, o menor caminho entre origem e destino. Nesse cenário, melhor esforço, métricas como perda de pacotes e atraso de comunicação não são priorizadas. Como resultado, algumas aplicações sensíveis

a essas métricas podem não funcionar como deveriam. Devido a popularidade, estrutura e escalabilidade do protocolo IP são propostos mecanismos para adicionar garantias quanto a qualidade de serviço. Várias técnicas podem ser utilizadas para melhorar a adequação das redes à aplicações multimídia. Podem ser estruturais, de modelagem de tráfego ou diferenciação de dados bem como uma mescla dessas.

A solução mais simples, porém, provavelmente a mais onerosa é superdimensionar os componentes da rede, aumentando a largura de faixa. Porém, essa solução não garante que as aplicações não consumirão todos os recursos, causando novamente o problema. Uma outra abordagem é aumentar o tamanho dos buffers nos roteadores. Isto diminui a variação de atraso, porém, aumenta o atraso total. As duas técnicas anteriores não resolvem o problema da Qualidade de Serviço, apenas melhoram o projeto.

Uma outra técnica baseia-se em estabelecer o nível de serviço. Isto é, modelar o tráfego pelo ajuste de tráfego médio de saída através de um acordo entre cliente e provedor. Para garantir que o contrato seja cumprido são necessárias políticas de policiamento. Duas técnicas podem ser usadas para tanto: balde furado e balde de símbolos [Tanenbaum (2000)].

O algoritmo do balde furado faz com que a taxa de saída seja constante independente do fluxo de entrada. Caso a rajada de entrada seja maior que a capacidade de armazenamento do buffer os pacotes são descartados. A Figura 2.5 exibe a analogia de um balde furado e a regulação do fluxo de pacotes.

A técnica do balde de símbolos permite rajadas controladas de saída. A cada unidade de tempo são geradas fichas, permissões, até um determinado limite. Quando o fluxo chega, os pacotes são liberados de acordo com o número de fichas. A Figura 2.6 exibe a lógica do funcionamento do balde de símbolos.

Essas técnicas permitem o controle de fluxo, porém não fazem a distinção entre aplicações de dados e multimídia.

Políticas de prioridade no encaminhamento podem ser utilizadas para diferenciar classes de fluxo. O roteador possui filas de entrada e saída, e a alocação dessas é feita por algoritmos de enfileiramento que definem quando os pacotes serão transmitidos e quais serão descartados, afetando a banda utilizada e a latência na transmissão. A combinação desses fatores pode ser utilizada como forma de obter Qualidade de Serviço. Uma fila pode ser definida por uma política de enfileiramento, que diz respeito a forma de armazenamento dos pacotes; um algoritmo de escalonamento, que define a ordem de transmissão dos pacotes e uma política de descarte, que determina quais pacotes serão descartados quando o buffer estiver cheio.

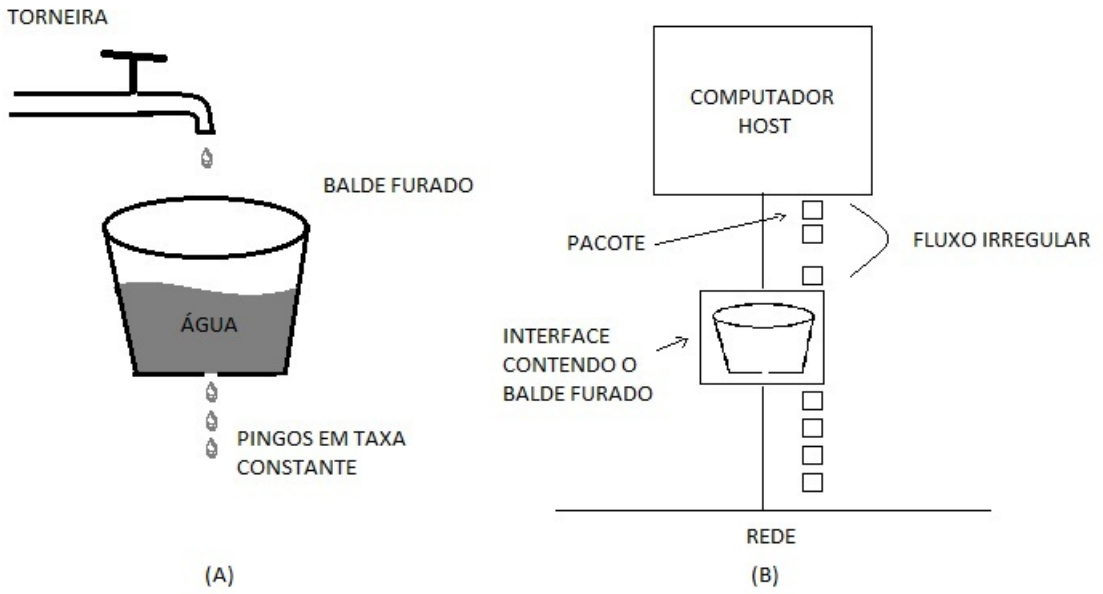

Figura 2.5: Funcionamento do algoritmo do balde furado

A presença de justiça e priorização também é importante. A justiça define que todos as fontes serão atendidas em algum momento e a prioridade define filas que terão mais pacotes transmitidos que outras. Algumas políticas de enfileiramento são: FIFO, FIFO com prioridade, Enfileiramento Justo e Enfileiramento Justo Ponderado.

Na FIFO (First In First Out), os pacotes são atendidos de acordo com a ordem de chegada. Quando o buffer enche, são descartados os pacotes do fim da fila (tail drop). Não existe mecanismo de priorização de pacotes. Na FIFO com prioridade o roteador implementa várias filas FIFO, uma para cada classe de prioridade. Enquanto houver pacotes na fila de maior prioridade esses serão transmitidos. A prioridade do pacote é marcada no cabeçalho IP do pacote, no campo Type Of Service (ToS). Pacotes não classificados são tratados como pacotes normais. O problema dessa abordagem é que filas de maior prioridade podem impedir que filas de menor prioridade transmitam. No Enfileiramento Justo (fair queuing) existe uma fila separada, sem prioridade, para cada fluxo. As filas são atendidas em sequencia (round-robin). Caso a aplicação aumente o fluxo além da capacidade do buffer especifico, os pacotes adicionais serão descartados. Essa estratégia não resolve o problema dos fluxos multimídia, já que não existe uma priorização. O Enfileiramento Justo Ponderado (Weighted Fair Queuing - WFQ) garante que todas filas serão atendidas ao mesmo tempo que prioriza algumas. Isto é, algumas filas podem transmitir mais bits que outras. Essa estratégia ajuda, mas não garante QoS. A Figura 2.7 ilustra o funcionamento dos mecanismos citados acima.

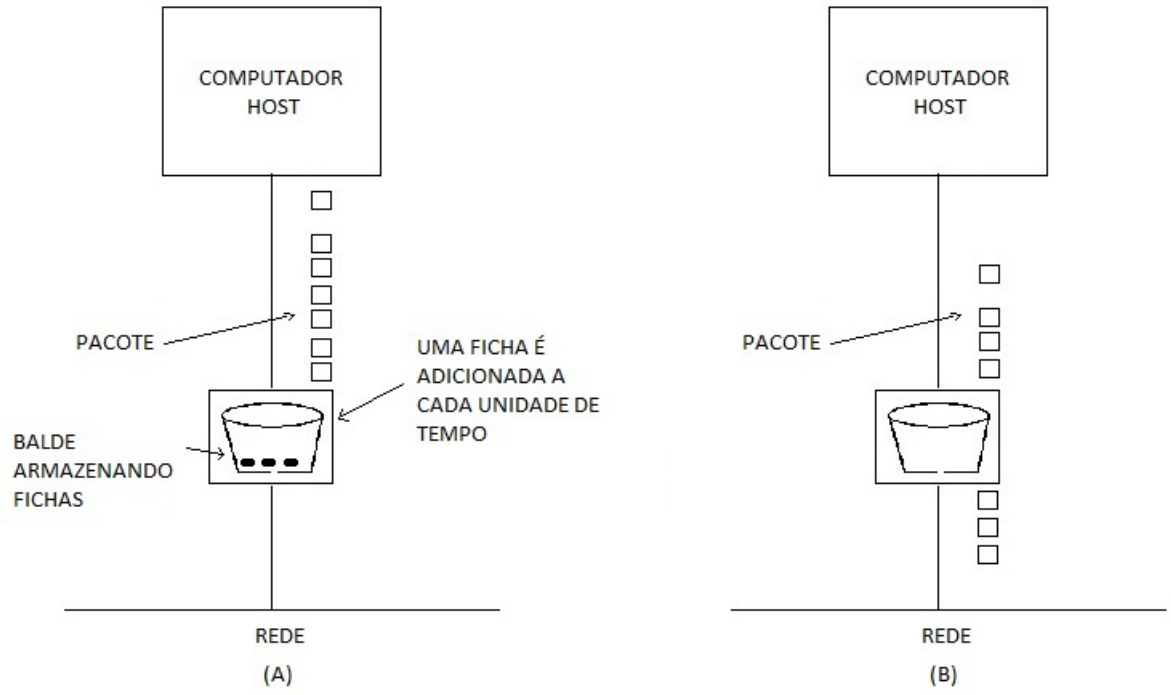

Figura 2.6: Funcionamento do algoritmo balde de símbolos

O gerenciador de Qualidade de Serviço possibilita o gerenciamento e a alocação de recursos [Andrade (2008)]. Esse basicamente faz o controle de admissão de novos fluxos e negocia os requisitos de QoS necessários. Quando a aplicação deseja utilizar a rede, essa envia para o gerenciador sua necessidade de QoS. O gerenciador verifica os recursos disponíveis na rede e permite ou nega a admissão do fluxo. Essa técnica busca evitar a sobrecarga da rede. Caso as necessidades da aplicação mudem, o processo se repete. O conjunto de parâmetros que especificam a Qualidade de Serviço é conhecido como especificação de fluxo [Andrade (2008)]. Os parâmetros são: largura de faixa, atraso e perda. A largura de faixa é composta pelos atributos: unidade máxima e taxa máxima de transmissão, que determinam a largura máxima do fluxo; taxa e tamanho das rajadas do fluxo, que determina o tamanho do balde de fichas. O atraso especificado pelo valor mínimo suportado e máxima taxa de flutuação. As perdas são definidas pelo número máximo de perdas por período e consecutivas.

Uma das arquiteturas para o provimento de QoS é a IntServ (Serviços Integrados) [Braden (1994)] . Essa permite o controle de admissão e reserva de recursos, garantindo que exista banda suficiente para o fluxo. A IntServ é uma estrutura criada pela IETF (Internet Engineering Task Force) para garantir Qualidade de Serviço para sessões individualizadas de aplicativos. Ele trabalha com reserva de recursos e estabelecimento de sessão. O termo serviços

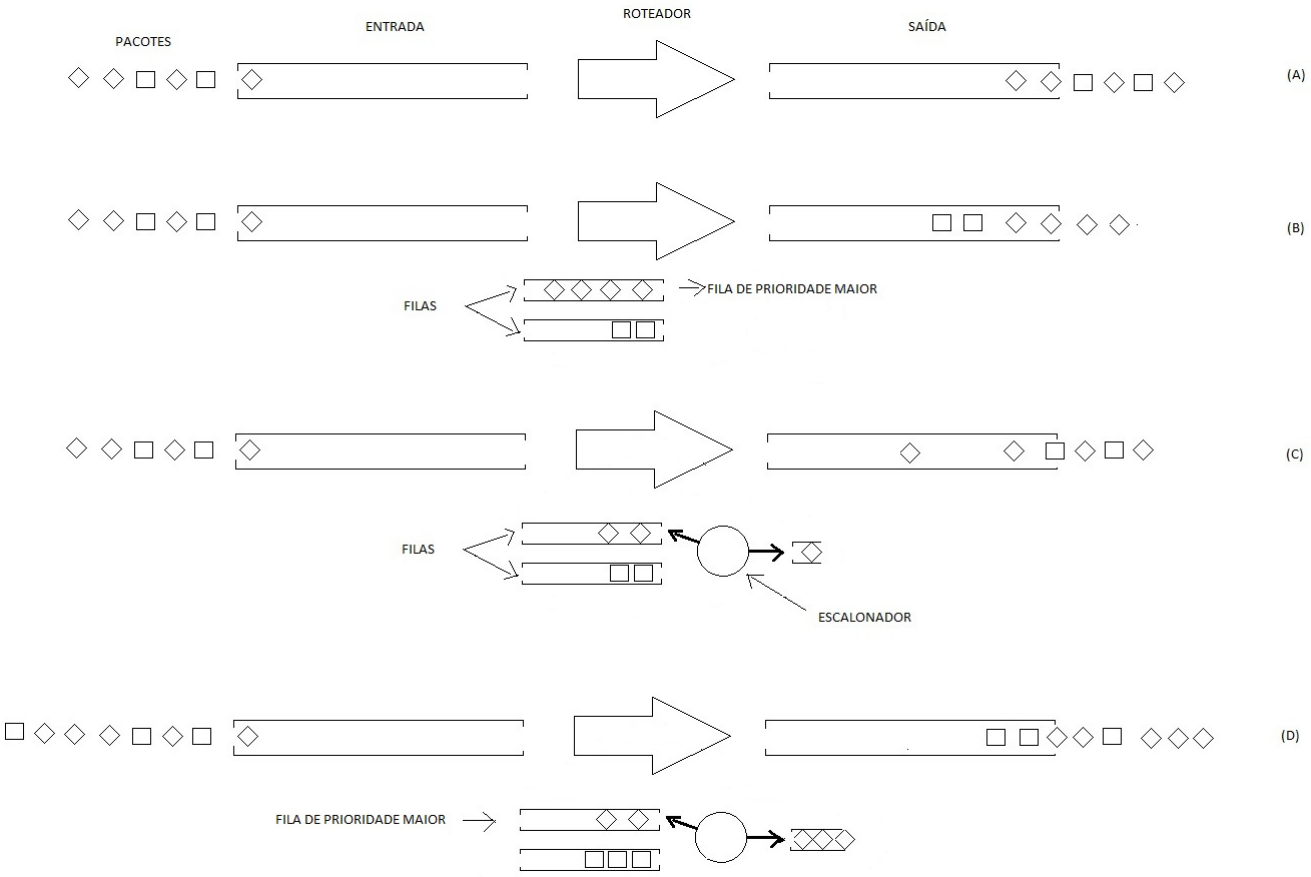

Figura 2.7: Políticas de Encaminhamento: (A) FIFO, (B) FIFO com prioridade, (C) Enfileiramento Justo e (D) Enfileiramento Justo Ponderado

integrados é empregado para designar serviços que incluem: serviço de melhor esforço, serviços de tempo real e serviços de compartilhamento controlado de enlace [Braden (1994)].

O protocolo principal do IntServ é o RSVP (ReSource reserVation Protocol) [Braden et al. (1997)], usado para reserva de recursos. O IntServ possui um overhead na reserva de recursos o que leva a problemas de escalabilidade.

Esse modelo é composto por quatro componentes:

- Escalonador de pacotes: gerencia o encaminhamento dos vários fluxos de pacotes. É composto também pelo avaliador, que mede características de tráfego na rede para auxiliar o escalonador no escalonamento de pacotes e controle de admissão.
- Classificador: mapeia os pacotes que chegam em determinadas classes. Os pacotes de uma mesma classe recebem o mesmo tratamento.

- Controle de admissão: implementa o algoritmo que possibilita ao roteador saber se aceita ou não o novo fluxo baseado em critérios de QoS. Alguns fluxos podem ser rejeitados por falta de recursos na rede.
- Controle de reserva: protocolo responsável por reservar os recursos necessários por todo o caminho do fluxo. Pode ser qualquer protocolo compatível com IntServ, na prática é utilizado o protocolo RSVP (Resource Reservation Protocol). O RSVP é utilizado pelos roteadores para repassar requisições de QoS para os demais roteadores no caminho do fluxo para manter informações de estado e garantir o serviço desejado.

O modelo de serviços integrados propõem mais duas classes de serviço além do serviço de melhor esforço: Serviço Garantido e o Serviço de Carga Controlada [Maia (2006)].

- Serviço Garantido: define um limite para atraso de enfileiramento nos roteadores. Ele garante o atraso e a taxa de bits.
- Serviço de Carga Controlada: receberá uma qualidade de serviço muito próxima da qualidade oferecida por uma rede não sobrecarregada.

A arquitetura de Serviços Integrados define que as informações sobre estado dos fluxos devem estar nos sistemas finais. Isso apresenta alguns problemas, como [Maia (2006)]:

- O montante de informações aumenta proporcionalmente ao número de fluxos causando sobrecarga de armazenamento e processamento nos roteadores, logo a arquitetura não é escalável.
- Todos os roteadores devem implementar RSVP, controle de admissão, classificação e escalonamento de pacotes, encarecendo o custo do roteador.
- Para serviço garantido, toda a rede deve suportar IntServ.
- IntServ/RSVP não é adequado a aplicações que apresentam fluxos de curta duração devido o overhead em sinalizar a reserva de recursos na rede, deteriorando o desempenho da aplicação.

A arquitetura DiffServ (Differenciated Services) [Blake (1998)], por sua vez, além das características mencionadas no IntServ, possibilita também o tratamento diferenciado dos fluxos através da definição de Classes de Serviço (Class of Service - CoS),possibilitando incluir métricas como atraso, perda de pacotes e disponibilidade de serviço.

Devido a dificuldade em implementar o IntServ/ RSVP, criou-se uma nova abordagem, os Serviços Diferenciados (DiffServ). A arquiterura DiffServ faz um agrupamento de fluxos que recebem o mesmo tratamento e não um fluxo separado. São criadas várias classes de serviços, onde cada classe possui um tipo de encaminhamento. Os pacotes que não pertencem a nenhum grupo (classe) recebem tratamento padrão, ou seja, um equivalente ao serviço de melhor esforço.

Na arquitetura DiffServ, os roteadores são agrupados em domínios. Os roteadores são classificados em borda ou núcleo. Normalmente um domínio DiffServ é composto por uma ou mais redes sob uma mesma administração [Maia (2006)]. A classificação dos pacotes ocorre nas bordas do domínio, criando várias classes de agregação de fluxo. O encaminhamento das agregações é definido nó a nó (Per-Hop-Behavior, PHBs).

Os PHPs são identificados pela marcação no campo DSCP (Differentiated Services Code Point). Esse campo é obtido renomeando o campo ToS (Type of Service), no caso do IPV4, ou Traffic Class, no IPV6. A diferenciação dos fluxos no núcleo do domínio é obtida através da análise do campo DSCP, escrito nos roteadores de borda, analisado a cada salto. Existem duas propostas para o PHB para DiffServ: PHB EF (Expedited Forwarding) e PHB AF (Assured Forwarding).

O PHB EF oferece um serviço fim-a-fim com baixa perda, baixo atraso, baixo jitter e largura de banda suficiente. O PHB AF garante um desempenho melhor que o serviço melhor esforço. Existe também o PHB default, usado para manter a compatibilidade com o fluxo melhor esforço de outros roteadores. Desta forma, os pacotes de aplicações multimídia devem encaminhados utilizando o PHP do tipo *Expedited Forwarding*.

O algoritmo do DiffServ possui basicamente três etapas: classificação dos pacotes, marcação dos pacotes e filtragem. O filtro é responsável por regular o fluxo de saída a fim de que o fluxo de saída tenha formas aceitáveis. Ainda assim essa estratégia não fornece garantias uma vez que depende do protocolo IP, não confiável. São necessários mecanismos que permitam melhor controle dos recursos da rede.

A arquitetura MPLS (MultiProtocol Label Switching) é a mais recente para alcançar qualidade de serviço em redes TCP/IP. A arquitetura MPLS engloba os avanços obtidos pelas arquiteturas anteriores e agrega o roteamento explícito como estratégia para alcançar os níveis desejados de QoS. Os pacotes que entram no domínio MPLS recebem um rótulo analisado exclusivamente, dispensando a leitura dos demais dados do cabeçalho do pacote. Ao sair do domínio o rótulo é retirado e então o pacote é encaminhado da maneira habitual. A arquitetura é formada por software e hardware capazes de utilizar suas funcionalidades. A parte de software

é formada pelos protocolos de comunicação e o hardware por roteadores capazes de identificar e tratar os rótulos de uma rede MPLS.

2.5 A Arquitetura MPLS

Multiprotocol Label Switching (MPLS) é um framework da Internet Engineering Task Force (IETF) que provê o encaminhamento de fluxos de tráfego através de uma rede [Trillium (2005)]. Esse pode ser definido como uma técnica de marcação de pacotes com rótulos curtos de tamanho fixo, os quais definem o tratamento que esses receberão na rede [Andrade (2008)]. Esses rótulos definem classes que estão associadas a determinadas rotas na rede. A característica mais importante do MPLS é a capacidade de encaminhar pacotes por rotas criadas explicitamente, possibilitando o uso de Engenharia de Tráfego, reduzindo, também, a complexidade do encaminhamento. A complexidade é reduzida uma vez que o encaminhamento dos pacotes é baseado em rótulos (labels) curtos e de tamanho fixo. Uma rede MPLS é capaz de oferecer garantias de Qualidade de Serviço sem a necessidade de enlaces dedicados [Andrade (2008)], uma vez que com o uso de TE é possível gerar rotas que respeitem as necessidades de QoS de cada fluxo.

Quando um pacote chega a um roteador, esse faz uma busca em sua tabela de roteamento. Essa busca pode demorar muito, dependendo do tamanho da tabela. O MPLS, por sua vez, encaminha os pacotes por meio de rótulos (labels) ao invés de endereços. Isso inclui a noção de encaminhamento de pacotes orientado a conexão nas redes IP, permitindo a criação e uso de caminhos de tráfego. Logo as redes MPLS possuem a flexibilidade das redes sem conexão e as vantagens das redes orientadas a conexão [Girish et al. (2000)].

O rótulo MPLS é inserido entre a camada de rede e enlace. Isso retira o processamento da camada de rede o qual passa ser realizado pela camada MPLS. O encaminhamento de pacotes com labels só é realizado dentro do domínio MPLS, onde os roteadores são capazes de entender os labels. Neste contexto, os roteadores são chamados de Label Switching Router (LSR). Os roteadores de borda responsáveis por inserir o rótulo no pacote são chamados Ingress Label Edge Router (ingress LER). Os roteadores de borda responsáveis por retirar o rótulo são chamados Egress Label Edge Router (egress LER).

Os caminhos aos quais os pacotes percorrem, neste contexto, são chamados LSPs (Label Switching Paths). O MPLS atribui uma Classe de Equivalência (FEC, Forwarding Equivalence Class) ao pacote quando ele entra na rede. Esta informação da FEC é então codificada em um label e inserida no pacote. A partir de então o cabeçalho não é mais analisado e não a mais busca na tabela de roteamento dos roteadores seguintes. A cada LSR

verifica-se o label e esse é substituído para indicar qual o próximo LSR. Quando o pacote sai do domínio MPLS o rótulo é retirado e o pacote passa a ser encaminhado novamente pela rede IP. Nesse processo, o LER, verifica a informação do tempo de vida (TTL), recalcula o checksum e remonta o pacote IP o qual é encaminhado segundo as informações de seu cabeçalho.

Existem duas formas de construir uma LSP: hop-by-hop e com roteamento explícito:

- No primeiro, o roteador escolhe o próximo salto de forma independente.
- Já com LSP de roteamento explícito, o roteador de borda (LSR), define quais os demais LSRs serão usados para o fluxo. Isto leva em consideração restrições como largura de banda suficiente e requisitos de QoS, além de políticas administrativas da rede. O roteamento explícito utiliza o CR-LDP (Constraint Routed Label Distribuition Protocol) ou o RSVP-TE (Resource Reservation Protocol for Traffic Engineering Extensions) como protocolo de sinalização [Maia (2006)]. A Figura 2.8 ilustra um LSP.

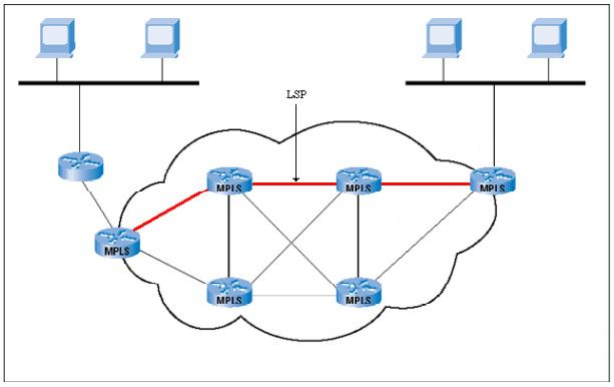

Figura 2.8: LSP (Label Switch Path)

Os LSR's usam um protocolo de comunicação para manter atualizadas as tabelas de encaminhamento, chamadas LIB (Label Information Base).

A lista a seguir resume os principais conceitos em uma rede MPLS.

- LSR (Label Switching Router) -É o roteador responsável por gerenciar os rótulos. Caso seja responsável por incluir os rótulos recebe o nome de Ingress LSR (LSR de Ingresso), caso seja por retirar os rótulos, ao sair do domínio MPLS é chamado Egress LSR (LSR Egresso).
- FEC (Forwarding Equivalence Class) - Na arquitetura MPLS, quando um pacote entra no domínio é definida ou verificada uma FEC para então ser associada a um rótulo. Esse

rótulo define o caminho que seguirá o pacote. Nos roteadores seguintes apenas o rótulo é verificado, e nele estão contidas as informações necessárias para o roteamento.

- LSP (Label Switched Path) -É o caminho qual o pacote deverá seguir até deixar o domínio MPLS. Pode ser implícito ou explícito. Há um interesse especial no explícito, já que definindo a rota a ser percorrida é possível atender os requisitos de QoS esperados na rede.
- LDP (Label Distribution Protocol) -É o protocolo de distribuição das informações de roteamento da rede. Ao se definir uma FEC e associá-la a um rótulo esse protocolo é usado para informar aos demais LSRs qual encaminhamento deverá ser usado para esse rótulo. Normalmente usa-se a estratégia de inundação para esse fim.
- TTL (Time-To-Life) -É o tempo de vida do pacote. Ao entrar no domínio MPLS o pacote recebe o número máximo de saltos. A cada salto um contador é incrementado e comparado a esse tempo de vida. Caso ultrapasse, esse pacote é descartado. Isso é utilizado para prevenir que um pacote circule indefinidamente na rede no caso de algum erro.

2.5.1 Protocolos de Roteamento versus MPLS

No roteamento IP, cada roteador presente no caminho, analisa o pacote IP e o encaminha de acordo com sua tabela de roteamento, isso faz com que o tempo de processamento seja proporcional ao tamanho da tabela. Os pacotes roteados utilizando protocolos de roteamento, como vetor de distância e estado de enlace, então, seguem o menor caminho entre os nós [Andrade (2008)]. Já numa rede MPLS, somente os roteadores de borda analisam o pacote, criando um caminho para este e atribuindo um *label*. Desta forma, o processamento é feito apenas nas bordas da rede, reduzindo o tempo de processamento. Os roteadores de núcleo apenas substituem o *label*, identificando o próximo salto do pacote. Além disso, a arquitetura MPLS é capaz de realizar roteamento explícito, criando circuitos virtuais, e em conjunto com a Engenharia de Tráfego é possível realizar o balanceamento e a otimização dos recursos da rede. Segundo [Trillium (2005)], o MPLS possui as seguintes características:

- especifica mecanismos para gerenciar fluxos de tráfego de várias granularidades;
- é independente dos protocolos das camadas 2 e 3;
- realiza mapeamento de endereços IP para rótulos de tamanho fixo;
- faz a interface com os protocolos de roteamento existentes.

2.5.2 Descrição do Rótulo

O cabeçalho do protocolo MPLS é chamado shim header, este é inserido entre o cabeçalho IP e o cabeçalho *Ethernet* e possui quatro campos. O primeiro campo é chamado *Label*, possuindo 20 bits. Este pode ser considerado uma forma abreviada para o cabeçalho do pacote, os roteadores o analisam para encaminhar o pacote. O segundo campo é chamado EXP, de 3 bits. Esse define a classe de serviço a qual o pacote pertence, definindo sua prioridade. O terceiro campo é chamado *stack* (S), de 1 bit. Este indica se o rótulo é o último da pilha e é utilizado devido a possibilidade do empilhamento de rótulos. O Quarto campo é chamado *Time-To-Life* (TTL), de 8 bits. Este indica o número de saltos restantes antes do descarte do pacote. O número máximo de saltos é 255, após esse número o pacote é descartado a fim de evitar *loops*. A Figura 2.9 retirada de [de Assis et al. (2002)] ilustra o cabeçalho MPLS.

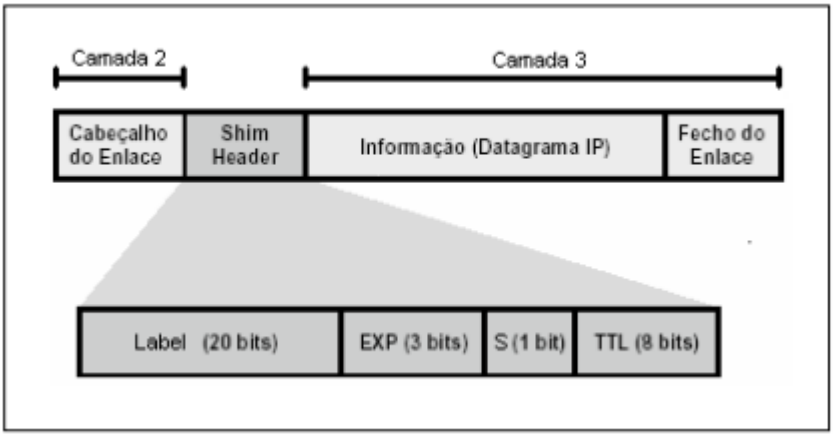

Figura 2.9: Rótulo MPLS

2.5.3 Estrutura de Dados MPLS

Cada LSR guarda informações sobre rótulos e instruções de encaminhamento. Esses trabalham de acordo com a tabela de encaminhamento Forwarding Information Base (FIB). Essa tabela possui os atributos:

- Next Hop Label Forwarding Entry (NHLFE) com as informações necessárias para o encaminhamento. Esse é composto pelos campos de endereço do próximo salto e um flag indicando se o label deve ser trocado, retirado ou colocado. O NHLFE insere rótulos de saída.
- O Incoming Label Map (ILM) é usado para interpretar os rótulos de entrada.

- FEC-to-NHLFE (FTN) é responsável por mapear pacotes em labels no LSR de ingresso.

A Figura 2.10 ilustra a utilização da estrutura de dados no encaminhamento dos pacotes MPLS.

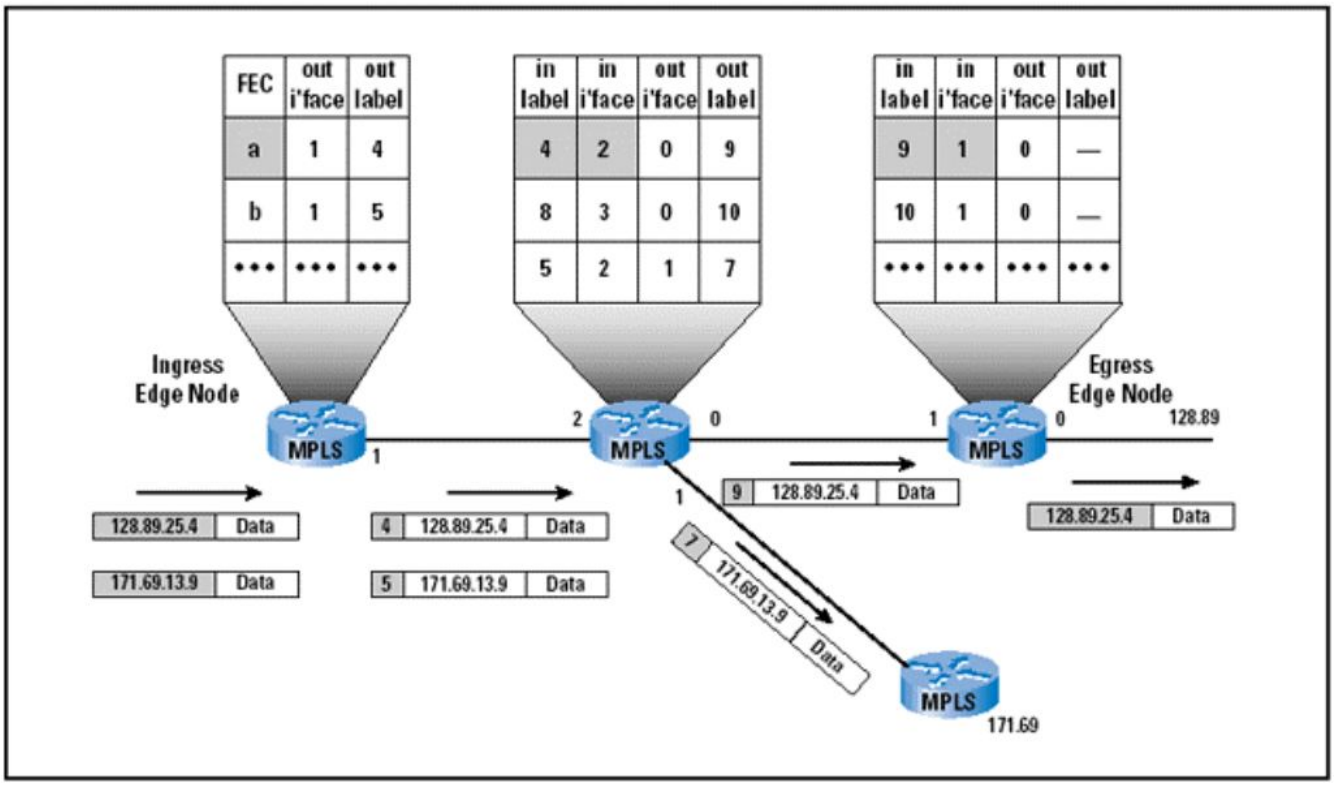

Figura 2.10: Tabela de Encaminhamento MPLS

Forward Equivalence Class (FEC)

O MPLS atribui a mesma classe de encaminhamento (FEC) a grupos de pacotes que possuam os mesmos requisitos de QoS. O rótulo possui um identificador ID usado para representar a FEC. Basicamente uma FEC poderá ser atribuída a um fluxo de três formas: utilizando do prefixo do IP de destino, associando o egress LER ao fluxo ou associando uma FEC a cada fluxo individualmente. A terceira forma é mais indicada para maior aproveitamento dos recursos. A Figura 2.11 ilustra a associação de um fluxo a uma FEC.

Distribuição de Rótulos

O protocolo MPLS utiliza um sistema de distribuição de rótulos (Label Distribuition Protocol - LDP). Esse inclui uma série de mensagens e processos para gerenciamento dos rótulos a fim

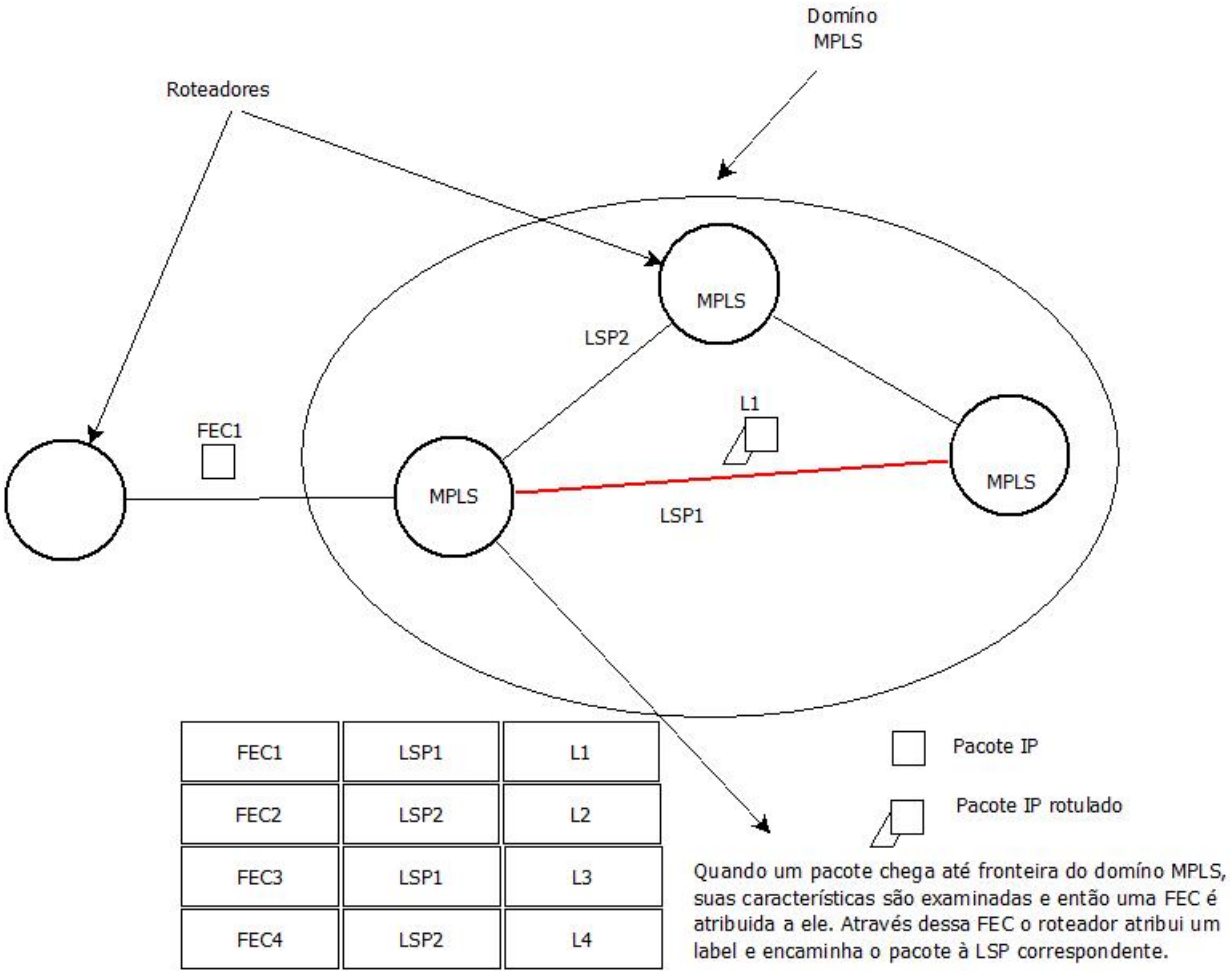

Figura 2.11: Classe de Encaminhamento

que os roteadores descubram uns aos outros permitindo a comunicação e o estabelecimento de LSPs. Os protocolos BGP e RSVP também podem ser usados para essa função.

Tabela de Rótulos

A tabela de rótulos, LIP (*Label Information Base*) é responsável por armazenar os rótulos de de entrada e saída de cada LSR. Estes possuem mecanismos que permitem compartilhar informações com outros LSR. As informações podem ser compartilhadas por demanda, quando alguma entidade solicita a informação ou quando acontece a associação FEC-rótulo [Andrade (2008)].

2.6 Roteamento baseado em QoS (QoSR)

Roteamento baseado em QoS pode ser definido como um roteamento onde o caminho a ser percorrido é traçado levando em consideração a disponibilidade de recursos na rede para determinado fluxo, bem como os requisitos de QoS desse, como largura de banda e atraso [Ash (2001)].

O Roteamento Baseado em Restrições (CBR, Constraint Based Routing) é o processo de computar rotas sujeitas a mais de uma restrição [Wroclawski (1997)], como custo e políticas de segurança. O roteamento baseado em QoS pode ser encarado como um CBR onde as restrições são critérios de qualidade de serviço (QoSR).

Diferente do roteamento convencional, o QoSR mantém o estado da capacidade dos links a fim de possibilitar o atendimento das restrições de QoS. Isso possibilita traçar rotas dinamicamente, além de possibilitar o re-roteamento. A otimização trata da utilização eficiente dos recursos da rede, como, por exemplo, buscar caminhos ociosos enquanto evita outros momentaneamente sobrecarregados. Isso é obtido utilizando técnicas de TE (Engenharia de Tráfego).

O QoSR é normalmente orientado a conexão utilizando reserva de recursos, fornecendo garantias de QoS. Normalmente a reserva de recursos é realizada através do protocolo RSVP [Braden et al. (1997)]. Esse não implementa um método para encontrar as rotas com recursos suficientes para os fluxos, implementada pelo mecanismo de QoSR. O Controle de Admissão de Conexão, por sua vez, determina se um fluxo deve ser aceito ou rejeitado verificando o sucesso ou fracasso da tentativa de reserva de recursos na rota determinada.

2.7 Engenharia de Tráfego

A Engenharia de Tráfego (TE) é uma técnica que busca otimizar o fluxo na rede para evitar a sobrecarga dos enlaces. A TE utiliza princípios tecnológicos e científicos para medir, modelar e controlar o tráfego com objetivo de otimizar o desempenho da rede [Maia (2006)], maximizando a utilização da largura de banda e realocando os fluxos em caso de sobrecarga e/ou falhas nos enlaces.

Segundo [Maia (2006)], a TE engloba quatro problemas: controle de admissão, roteamento baseado em restrições, re-roteamento e planejamento de recursos:

- Controle de admissão - determina se uma requisição pode ser ou não admitida, caso possa, essa é inserida na rota determinada.

- Roteamento baseado em restrições - responsável por encontrar caminhos que satisfaçam as restrições impostas, como, por exemplo: número de saltos, taxa de transmissão, atraso fim-a-fim e variação do atraso. O roteamento é realizado sobre a banda residual obtida após a retirada da banda necessária para atender a requisição.

- Re-roteamento - trata-se da realocação de um fluxo devido a congestionamento, falha no enlace ou para melhorar a eficiência da rede. Existe a ressalva, porém, que o re-roteamento maciço das rotas pode deterior o desempenho da rede devido o overhead causado pela troca de mensagens de controle entre os roteadores, quando feito frequentemente.

- Planejamento de recursos - objetiva prever o comportamento das próximas demandas a fim de reservar recursos suficiente para atende-las. Pode ser estimado por meio do histórico de tráfego na rede [Girish et al. (2000)]. Esse planejamento pode ser realizado por meio de quatro processos básicos: medição, modelagem, análise e otimização [Awduche et al. (2002)].

  - Medição - Utilizada para mensurar a QoS na rede e validar as política de TE. Os dados obtidos são utilizadas a fim de manter a rede otimizada. É mais eficiente quando segue um padrão para obtenção e aplicação.
  - Modelagem - Construção de um modelo que descreva as características de tráfego e atributos da rede [Maia (2006)]. O modelo exibe as informações mais importantes, como restrições, nós e enlaces, facilitando a tarefa de análise e simulação. A simulação é utilizada a fim de prever o comportamento da rede, tornando possível o uso de estratégias para prevenir a deterioração do desempenho da rede.
  - Análise - trata-se da verificação do estado da rede e caracterização da carga de tráfego representativa [Maia (2006)]. Os dados obtidos na etapa de medição são tratados a fim de encontrar enlaces sobrecarregados e subutilizados bem como potenciais gargalos e pontos sujeitos a falhas. Pode ser dividida em reativa ou proativa. A reativa busca identificar o problema por meio de diagnósticos e tenta resolvê-los. Nesse caso o problema já está estabelecido. A proativa por sua vez tenta prever o problema para tomar iniciativas antes que de fato eles aconteçam.
  - Otimização - trata-se da potencialização do desempenho da rede. A otimização de desempenho da rede pode ser corretiva ou preventiva [Awduche et al. (2002)]. Na primeira o problema já existe e a meta é resolvê-lo. Na segunda o objetivo é melhorar o desempenho da rede mesmo quando não existem problemas e/ou não são previstos [Maia (2006)]. A otimização deve ser feita de forma continua e pode

ser em tempo real ou não. Na fase de planejamento o otimização não é feita em tempo real.

A capacidade do protocolo MPLS de gerar rotas explícitas somadas a mecanismos de medição e criação de LSPs permitem a implementação da TE. A seleção das LSPs pode ser classificada de acordo com o abordagem para a resolução de congestionamentos, disponibilidade de informações sobre o estado da rede e das demandas a serem alocadas. Quanto a abordagem para a solução de congestionamentos, a seleção das LSPs podem ser classificadas em preventivas e reativas. Em abordagens preventivas a alocação de caminhos previne o congestionamento. Nas reativas as medidas são tomadas após o estabelecimento do congestionamento.

Quanto a disponibilidade de informações sobre o estado da rede o processo pode ser classificado em estático ou dinâmico. No estático o estado da rede é verificado e as informações são utilizadas para gerar as rotas, que não serão mais alteradas. Na abordagem dinâmica, o estado da rede é verificado de tempos em tempos e as rotas são alteradas a fim de refletir o estado atual da rede.

Quanto a disponibilidade de informações sobre as demandas o processo de seleção de LSPs pode ser classificado em on-line ou off-line. No processo on-line as demandas surgem aleatoriamente. No off-line todas as demandas são conhecidas e a seleção das LSPs consiste em organiza-las da melhor forma simultaneamente.

O cálculo para seleção de rotas ainda pode ser classificado em centralizado e distribuído. No centralizado as informações são reunidas em um roteador ou servidor e o cálculo é efetuado. Na descentralizada o calculo é feito nos roteadores da rede [Andrade (2008)].

As LSPs são determinadas por algoritmos de roteamento baseados em QoS, e embora existam várias técnicas, essas se enquadram em dois grandes grupos: algoritmos baseados em caminho mínimo e balanceamento de carga. Os algoritmos de caminho mínima buscam encontrar o caminho ótimo segundo algum um critério, sendo que nem sempre se trata do caminho com menor número de nós intermediários. Algoritmos que visam o balanceamento de carga buscam enviar os fluxos de maneira equilibrada, não sobrecarregando alguns enlaces enquanto outros ficam ociosos.

2.8 Otimização mono e multi-objetivo

Na otimização mono-objetivo existe apenas uma função objetivo enquanto as demais são modeladas como restrições. A comparação das soluções obtidas é direta uma vez que essas

só contemplam uma dimensão. Dentro da área de otimização existem problemas os quais a solução depende da adequação de vários objetivos ao invés de apenas um. Normalmente os objetivos são conflitantes entre si e a melhora do resultado de um objetivo acarreta a deterioração do resultado de outros. Essas características definem um problema de Otimização Multiobjetivo (POM), também chamado de Otimização Multicritério ou Multiatributo [Sampaio (2011)].

Um problema de otimização sujeito a dois ou mais objetivos é caracterizado como um problema multiobjetivo. Nesse tipo de problema é possível definir dois tipos de solução: soluções que comparadas as outras apresentam simultaneamente desempenho inferior (soluções dominadas) e soluções as quais apresentam resultados superiores para um ou mais objetivos. Esse segundo grupo, formado por soluções não-dominadas, também chamado de soluções Pareto-ótimas, é denominado conjunto Pareto-ótimo. O segundo grupo citado anteriormente é construído por meio do conceito de dominância e de Solução Pareto-Ótima definidos em [Santos (2009)] como:

- O ponto x domina o ponto y se x é diferente de y e para função f, f(x) $<=$ f(y).

A solução Pareto-Ótima pode ser definida como:

- A solução é Pareto-Ótima se não existe qualquer outra solução que domine-a.

Considerando todas as soluções factíveis para o problema multi-objetivo, todas soluções não-dominadas compõem o conjunto Pareto-ótimo.

As abordagens para o tratamento de problemas multi-objetivos se dividem em duas correntes segundo [de Castro (2001)]:

- definir prioridades ou pesos para as várias funções objetivo de interesse aglutiná-las em uma única função objetivo;
- sem nenhuma informação adicional encontra-se o conjunto das soluções ótimas de Pareto para escolher uma dentre estas posteriormente.

A primeira alternativa considera a existência de informações suficientes para definir os pesos com exatidão. Porém, na maioria dos problemas multi-objetivo, as informações são escarças, tornando a segunda abordagem mais interessante.

Uma solução utópica também pode ser definida para fins de comparação da qualidade das soluções em problemas multi-objetivo. Essa é definida como a solução que apresenta

valores ótimos para todos os objetivos simultaneamente. Essa definição pode ser usada para fazer estimativas da distância da solução ou ser usada como métrica para tomada de decisões.

A diferença principal entre otimização mono e multi-objetivo está no número de dimensões. A mono-objetivo é unidimensional e a multi-objetivo é multi-dimensional, chamado espaço Z, o que aumenta a complexidade do problema. No caso do problema em questão, cada indicador de QoS considerado é modelado como uma função objetivo.

Devido as abordagens multi-objetivo acarretarem um conjunto de soluções não-dominadas, faz-se necessário um mecanismo para escolher qual a solução será de fato utilizada. Em [Maia (2006)] um rede neural é proposta a fim de escolher automaticamente uma das soluções do conjunto Pareto-Ótimo. Uma outra possibilidade é a utilização do método de ponderação dos objetivos baseado no método AHP (*Analytic Hierarchy Process*) proposto por [Saaty (1980)], utilizado neste trabalho. O método de ponderação dos objetivos atribui pesos entre 0 e 1 às funções objetivo de forma que o somatório desses pesos seja igual a 1. Através de operações de harmonização e normalização aplicadas a um conjunto de soluções, é possível definir uma nota para cada solução, tornando possível a criação de um *ranking*. Desta forma, alterando os pesos é possível escolher uma nova solução sem a necessidade de executar todo o método de otimização novamente.

2.9 Alternativas exatas para a solução de problemas multiobjetivo

Entres as alternativas exatas, duas técnicas se destacam pela simplicidade e por serem mais difundidas. Ambas reduzem o problema multi-objetivo em um mono-objetivo. Essas são:

- Ponderação de objetivos - Consiste em definir um multiplicador (peso) para cada função objetivo, sendo que a soma desses multiplicadores deve ser igual a 1 (100%). As funções objetivo são então aglutinadas, reduzindo-as a apenas uma. Ou seja, a espaço de busca passa a ser unidimensional e a comparação das soluções é direta. A solução desse resultará em um ponto no conjunto Pareto-ótimo.

- Problema e-Restrito - Consiste em converter as funções objetivo em restrições, exceto uma delas. Para cada nova restrição é definido um limite. A variação desses valores limitantes resulta em novos pontos no conjunto Pareto-ótimo. Aleatoriamente as restrições são modificadas a fim de explorar outros pontos no espaço de busca. Como desvantagem, esse método apresenta um aumento na complexidade ao aumentar o número de

restrições. Na definição dos limites das novas restrições, existe a possibilidade de atribuir valores inviáveis.

2.10 Heurísticas - Computação Evolutiva

Uma vez que o problema da alocação de rotas em redes MPLS sujeitas a mais de duas restrições de QoS é NP-Completo [Shao et al. (2006)], uma solução exata não é viável. Este trabalho utiliza um algoritmo baseado nos conceitos de computação evolutiva a fim de encontrar uma solução aproximada do problema em tempo computacional viável.

A computação evolutiva é formada por algoritmos inspirados na teoria da evolução de Darwin. Essa afirma que o princípio da seleção natural privilegia indivíduos mais aptos para reprodução. Os indivíduos com mais descendentes têm maior probabilidade de perpetuarem seu código genético nas gerações seguintes. Esse tipo de algoritmo é usado, principalmente, na solução de problemas os quais o método exato é inviável. Um dos representantes da computação evolutiva são os Algoritmos Genéticos (AGs).

Os AGs são algoritmos de otimização global desenvolvidos por [Holland (1975)] baseados nos mecanismos de genética e seleção natural. A população inicial evolui com sucessivas operações de seleção, cruzamento e mutação. Uma das vantagens dos AGs é trabalhar com um conjunto de soluções simultaneamente, cobrindo um maior espaço de busca, aumentando a possibilidade de encontrar a solução global.

Segundo [Goldberg (1989)] as principais diferenças são:

- codificação de parâmetros em detrimento a utilização direta de parâmetros;
- população de soluções candidatas e não uma única solução;
- utilizam informações de custo e recompensa, e não derivadas de funções;
- utilizam regras de transição probabilísticas e não determinísticas.

De acordo com a nomenclatura utilizada nos AGs, define-se uma solução candidata como individuo. Um conjunto de soluções candidatas, ou indivíduos, avaliadas simultaneamente é dado o nome de população. Cada indivíduo possui um grau de adaptação ou aptidão, também chamado de fitness. Esta mede a qualidade da solução dada pelo indivíduo. Os AGs não garantem soluções ótimas, mas aproximam-se bastante dessas em um tempo computacional razoável. Embora tenha um componente aleatório, usado para garantir a diversidade da população e evitar a estagnação em pontos mínimos ou máximos locais, a busca é guiada

e não totalmente aleatória. Segundo [Maia (2006)], um algoritmo genético é composto pelos passos descritos na Figura 2.12.

```
1 Escolher a população inicial;
2 while individuo adequado não for obtido ou número máximo de gerações não
  alcançado do
3 |   Definir a nota de cada individuo;
4 |   Aplicar operadores de reprodução sobre os indivíduos selecionados;
5 |   Aplicar operadores de mutação sobre os filhos gerados pelo cruzamento;
6 end while
```

Figura 2.12: Algoritmo Genético

Os indivíduos mais aptos devem ter maior probabilidade de reprodução, mas ainda assim os demais indivíduos devem ter possibilidade de realizarem o cruzamento. Alguns parâmetros, como: taxa de cruzamento e mutação, tamanho da população afetam diretamente a eficiência do algoritmo e devem ser escolhidos cuidadosamente de acordo com o problema a ser resolvido. O elitismo também deve ser avaliado a fim de que bons indivíduos sejam assegurados nas gerações seguintes. O cruzamento possibilita explorar novos pontos no espaço de busca pela combinação de indivíduos afim de gerar outros diferentes. Normalmente a taxa de cruzamento varia de 50% a 100%. Entre as técnicas de cruzamentos destacam-se: o cruzamento de um ponto, dois pontos, ponto variável, cruzamento uniforme e cruzamento ortogonal [Andrade (2008)].

Um dos métodos mais comuns utilizados para a seleção dos pais é o método da roleta. Neste, o indivíduo ocupa uma área na roleta proporcional a sua aptidão. Assim os melhores indivíduos ocupam uma área maior e têm maior probabilidade de serem escolhidos. Existe o problema, porém, da monopolização da roleta, caso um individuo seja muito superior aos demais. Caso os indivíduos tenham aptidões próximas, a escolha pode não privilegiar os indivíduos mais adequados. Uma solução para esses problemas é a técnica do ranking [Maia (2006)], onde a área ocupada pelo individuo na roleta é baseada em sua qualificação (ranking) perante a população. Os indivíduos selecionados participam da fase de reprodução onde podem ser combinados ou modificados através de operadores genéticos.

O crossover é responsável por recombinar as características genéticas dos pais, permitindo que elas sejam herdadas nas próximas gerações [Maia (2006)]. Esse operador é dominante e portanto tem maior probabilidade de ser aplicado que a mutação. Taxas muita altas de cruzamento acarretam numa substituição acelerada de indivíduos, aumentando a possibilidade de

descartar bons indivíduos antes que outros melhores sejam gerados. Taxas baixas acarretam na estagnação da população.

A mutação é o operador responsável por manter a diversidade genética. Este aleatoriamente modifica os genes do individuo, fazendo com que a possibilidade de exploração de todo o espaço de busca não chegue a zero, assegurando, também, que a solução não fique confinada em mínimos ou máximos locais. Normalmente a taxa de mutação é baixa, uma vez que uma taxa alta levaria a um comportamento muito aleatório. Taxas muito baixas, por sua vez, restringem o espaço de busca.

Um outro parâmetro utilizado em AGs é o intervalo de geração. Esse controla a taxa de indivíduos que serão substituídos na geração seguinte. Altas taxas fazem com que indivíduos com boa aptidão sejam retirados da população antes que outros melhores sejam gerados. Taxas baixas tornam o algoritmo muito lento devido a necessidade de um maior número de gerações.

2.10.1 Representações e conceitos de AGs

Um AG parte de uma população inicial aplicando-se a cada geração os operadores genéticos de seleção, cruzamento e mutação até que seja alcançado o número máximo de gerações ou seja encontrada a solução adequada. No AG simples (Simple Genetic Algorithm - SGA), todos os indivíduos são substituídos a cada geração. No Algoritmo de Reposição (Replacement Genetic Algorithm -RGA), apenas uma parte dos indivíduos é substituída. Já no AG de Estado Estável (Steady State Genetic Algorithm -SSGA) apenas um percentual de indivíduos é substituído para geração seguinte. Normalmente a decisão de que um indivíduo deverá ou não permanecer na população é verificada de acordo com a aptidão do mesmo. Essa, por sua vez, é calculada com base na função objetivo.

A função objetivo deve ser capaz de estabelecer uma relação entre o problema a ser resolvido e uma equação matemática que o caracterize[Andrade (2008)]. Funções objetivo que geram valores negativos podem invalidar algoritmos de seleção. Para tanto abordagens como a soma de uma constante suficientemente grande a função, mapeamento linear e penalizações podem contornar esse problema. A representação utilizada neste trabalho é apresentada na (Seção 2.11).

Os parâmetros de entrada também têm forte influencia sobre o desempenho do AG. Os principais parâmetros são: tamanho da população, taxa de mutação, taxa de cruzamento e critério de parada [Andrade (2008)]. O tamanho da população depende da complexidade do problema. Quanto mais indivíduos, maior a probabilidade de encontrar boas soluções, por

aumentar o espaço de busca explorado ao custo de aumentar o tempo de processamento. A taxa de cruzamento controla a porcentagem da população que seguirá para o cruzamento. Quanto maior a taxa, maior a velocidade de convergência e maior possibilidade de convergência prematura [Andrade (2008)]. Baixas taxas de cruzamento levam a um tempo de convergência mais longo devido a baixas trocas de material genético. O operador de mutação tem como objetivo aumentar a variabilidade da população, assim os indivíduos podem explorar outras regiões do espaço de soluções. Altas taxas de mutação podem tornar a busca aleatória e baixas taxas podem prender a solução em mínimos/máximos locais. O critério de parada do AG pode ser o número de gerações, número consecutivo de gerações sem melhora de resultados ou tempo de execução.

2.11 Representação do Problema

Foi utilizado um algoritmo genético multi-objetivo inspirado no algoritmo NSGA-II para resolver o problema da alocação de rotas. Optou-se por não utilizar representação binária na representação do individuo uma vez que o algoritmo implementado usa engenharia de tráfego tanto na mutação quando no cruzamento. No trabalho de Andrade (2008) utiliza-se representação binária e a mutação faz com que o cromossomo passe a representar uma outra rota já definida na inicialização do algoritmo por meio do algoritmo KShortest-Paths. A cada mutação o cromossomo gerado (conjunto de bits) é consultado na estrutura de rotas, criadas inicialmente, e o custo do caminho é recalculado para o individuo em questão. Ou seja, as rotas possíveis estão restritas ao número de rotas geradas na inicialização do algoritmo. Este trabalho usa a mesma estratégia para criar rotas viáveis inicialmente, porém, o conjunto de rotas iniciais é abandonado no inicio do AG e cada indivíduo passa a se modificar parte por movimentos aleatórios, parte por busca guiada. Essa notação torna necessário que cada indivíduo carregue consigo as rotas utilizadas.

Os critérios para a definição do fitness do indivíduo utilizados nesse trabalho são:

1. Número de rejeições (Minimização)
2. Número de enlaces utilizados (Minimização)
3. Desvio padrão da banda residual na rede (Minimização)

Verifica-se contradições entre as funções objetivo, comuns em algoritmos multiobjetivo. O aumento no número de rejeições faz com que o número de enlaces utilizados caia. Já privilegiar a minimização do número de enlaces utilizados pode desbalancear a carga na rede,

prejudicando a minimização do desvio padrão da banda residual. De uma forma geral o algoritmo busca atender o maior número de demandas possíveis buscando a otimização do uso de enlaces sem deixar a carga da rede desbalanceada, e obviamente, sem exceder a capacidade de escoamento do enlace.

A rede foi representada como uma matriz de adjacências, porém, optou-se por cortá-la na diagonal uma vez que este trabalho considera que um enlace é bidirecional e as operações de preenchimento e verificação de matrizes seriam mais eficientes diminuindo o tamanho da matriz. A topologia da rede é representada pelas matrizes:

- Adjacência - Representa os enlaces existentes na interligação dos nós. 0 representa a ausência do enlace e 1 a presença do enlace.
- Capacidade - Representa a banda suportada pelo enlace. Vai de 0 a 1024.
- Atraso - Representa o tempo necessário para o pacote sair do nó transmissor e chegar o nó receptor. Optou-se pelo valor de 3 milésimos de segundo.
- Jitter - Representa a variação de atraso em transmissões sucessivas. Optou-se pelo valor de 3 milésimos de segundo.
- Custo - Representa o custo atrelado a transmissão. Optou-se pelo valor 3.

A estrutura básica da solução é representada por um indivíduo. A Figura 2.13 define o esquema de um indivíduo.

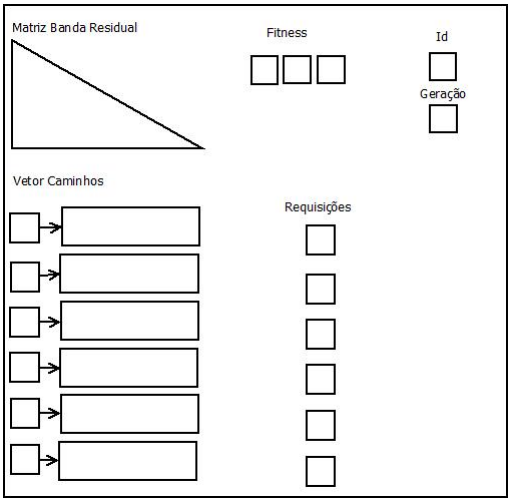

Figura 2.13: Esquema de Indivíduo

A matriz banda residual representa a banda remanescente após a alocação das demandas. Mais uma vez, como todos os enlaces são considerados bidirecionais, optou-se por cortar a matriz na diagonal. Vetor Caminhos representa as rotas definidas para as respectivas demandas. Cada caminho pode ser considerado um cromossomo. Requisições representa um conjunto de demandas com seus respectivos status, atendida ou não atendida. Fitness é um vetor que representa o valor dos três objetivos escolhidos como função objetivo desse problema. Id é um número inteiro positivo que representa unicamente um individuo durante a execução do algoritmo. Geração é um número inteiro positivo que identifica a qual geração o indivíduo pertence.

Capítulo 3

Algoritmo

Este capítulo apresenta a solução adotada para resolução do problema de alocação de rotas. Apresenta-se a modelagem do problema bem como a representação das classes e uma breve explicação da função de cada uma.

3.1 Diagrama de Classes

A Figura 3.1 representa o diagrama de classes simplificado do algoritmo de alocação de rotas MPLS.

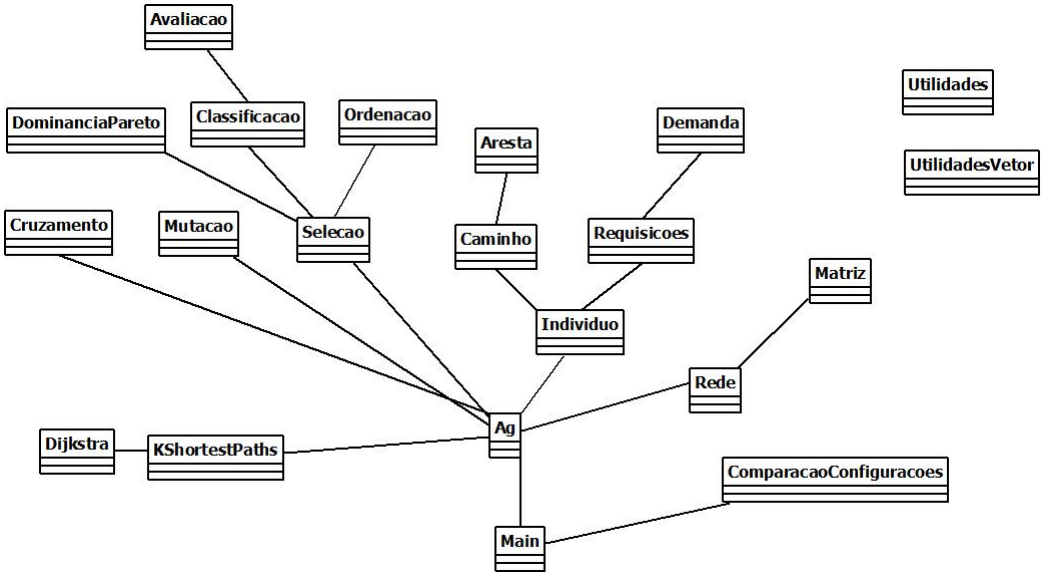

Figura 3.1: Diagrama de Classes do problema de alocação de rotas

3.1.1 Matriz

A classe matriz é a principal estrutura de armazenamento de dados deste algoritmo. Essa pode ser carregada a partir de um arquivo texto ou ser digitada. Possui métodos para calculo de média, variância e desvio padrão. Além disso é capaz de fazer cálculos sobre um escopo específico, mais especificamente sobre um vetor de índices passados como parâmetro, para a realização de somas e subtrações. Ainda existem métodos para soma, subtração e multiplicação de constantes e utilitários para gravação em arquivo texto e exibição para testes.

3.1.2 Rede

Aglutina as informações sobre a rede citadas anteriormente sob a forma de matrizes. As informações são carregadas através de um arquivo texto. Possui métodos para carregar e editar o valor dos atributos e utilitários para exibição e geração de um rede aleatória para testes.

3.1.3 Aresta

Representa um enlace com informações sobre os vértices que o formam e o custo envolvido. Possui métodos para carregar e editar o valor dos atributos.

3.1.4 Caminho

Representa as rotas. Possui um vetor inteiro representando os nós que formam o caminho e o custo para percorrê-lo. Além dos métodos para carregar e editar atributos, possui métodos para verificação da existência do caminho, tamanho e exibição desse.

3.1.5 Dijkstra

Essa classe representa o algoritmo de Dijkstra utilizado para encontrar o menor trajeto entre dois pontos em em grafo com valores não negativos. Seu construtor recebe uma instância da classe Matriz, nó inicial e nó final. O método 'caminhoEPossivel', de retorno booleano, é utilizado para verificar se o algoritmo obteve sucesso. Em caso positivo o método 'getMenor-Caminho' é utilizado para obter a rota encontrada.

3.1.6 KShortestPaths

Essa classe faz uso do algoritmo de Dijkstra sucessivamente retirando a aresta de menor custo do menor trajeto encontrado a fim de construir um conjunto com os menores caminhos

possíveis. A condição de parada é a impossibilidade de encontrar um caminho válido. Isso acarreta na possibilidade de gerar números distintos de rotas para uma ou outra requisição. Para equilibrar o número de rotas foi criado o método 'completeAte'. Esse método deixou de fazer sentido quando decidiu-se mudar a representação do problema, uma vez que não era mais necessário garantir que a conversão de um número binário para decimal resultasse em um número fora dos índices da tabela de rotas.

3.1.7 Demanda

Essa classe representa uma solicitação para a rede podendo ser ou não atendida. Juntamente com os nós emissor e receptor é passado a banda necessária para atender as necessidades do fluxo. Além dos métodos para carregar e editar parâmetros o método 'isAtendida' de retorno booleano é utilizado para verificar o status da demanda.

3.1.8 Requisições

Essa classe representa o conjunto de todas as demandas que devem ser otimizadas. Para gerar as requisições são passados como parâmetros o número de requisições, banda mínima e banda máxima. Cada requisição terá banda entre os limites de banda mínima e máxima. Caso banda mínima e máxima sejam iguais, todas as requisições geradas requisitarão a mesma quantidade de banda de rede. As requisições também poderiam ser carregadas a partir de um arquivo texto, desde de que no formato necessário.

3.1.9 Indivíduo

Essa classe representa uma solução viável para o problema. Seu construtor recebe uma instância da classe Rede e uma instância da classe 'Requisicoes'. O método estático 'geraCaminhosPorDemanda' é responsável por criar rotas viáveis para as requisições no momento da instanciação. Então durante o ciclo de vida do algoritmo a cada nova instanciação de Individuo um conjunto de rotas é escolhido aleatoriamente e um ID único é atribuído a nova instância.

A classe 'Individuo' é capaz de se comparar com outras instâncias através dos operadores: menor ($<$), maior($>$) e igual ($==$). Exceto o operador 'igual', onde verifica-se o ID apenas, os outros operadores comparam seus valores de fitness em pares, com a ordem de precedência: número de requisições não atendidas, número de enlaces utilizados e desvio padrão da banda residual. De tal forma, se A possui 5 requisições não atendidas e B possui apenas 2, B será classificado a frente de A. Caso a primeira comparação seja idêntica, parte-se para o segundo critério e assim sucessivamente.

O método 'semelhanca' usa outra abordagem. Esse retorna um valor entre 0 e 1 representando quanto as rotas se assemelham de um indivíduo para outro. 1 significa 100% semelhante enquanto que 0 representa 0% semelhante. A menor parte de comparação neste caso é o caminho. Ou seja, para 3 demandas, se duas das respectivas rotas forem iguais para ambos indivíduos, a semelhança entre eles será de 66%. A Figura 3.2 ilustra o método de comparação de semelhança.

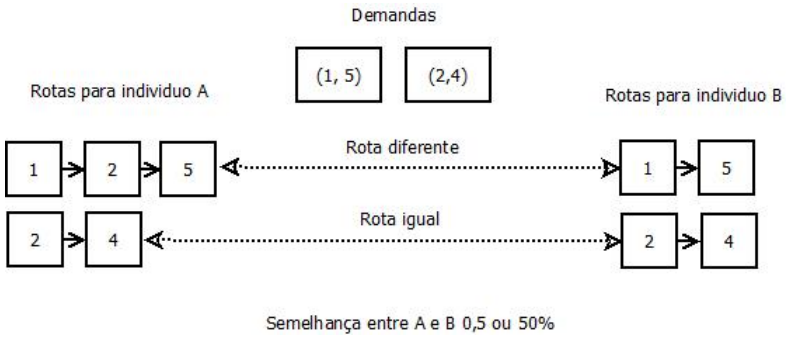

Figura 3.2: Método 'Semelhanca'

A classe 'Individuo' também possui a capacidade de recalcular seu fitness. Para isso são utilizados os métodos 'reload', 'reloadOrdemAleatoria' e 'reloadForPath'. Estes são utilizados após os operadores de cruzamento e seleção. Esses métodos buscam garantir que nunca uma rota será alocada sem que existe capacidade suficiente na rede para tanto. Em operações como a mutação, a ordem de reavaliação das rotas de um indivíduo pode definir uma prioridade de alocação de um fluxo. Para tornar a reavaliação justa, o indivíduo é reavaliado de maneira aleatória. Assim as primeiras demandas não são privilegiadas em relação às últimas.

3.1.10 Utilidades

Essa classe aglutina alguns métodos que não se encaixaram em outras classes por serem genéricos. Possui métodos para geração de números aleatórios inteiros entre determinados índices, fatorial, geração aleatória de valor booleano e conversão de inteiros em string.

3.1.11 UtilidadesVetor

Representa a mesma ideia de Utilidades. Possui métodos utilizados para operações sobre conjuntos, como: união e interseção. Além de métodos para concatenação de vetores inteiros, preenchimento linear e conversão para string.

3.1.12 Seleção

Esta classe contém os métodos de seleção usados tanto para selecionar o indivíduos que passarão para a próxima geração, quanto os pais selecionados para o cruzamento. Esta classe recebe um array de Indivíduos e um número máximo de indivíduos a serem selecionados, além de alguns parâmetros opcionais. As seleções implementadas nessa classe são: aleatória, por torneiro de Pareto, por distinção de rotas, fronteira de Pareto, torneio, roleta, roleta-ranking e ranking. Em 3.4.1 esses métodos serão abordados com mais detalhes.

3.1.13 Ordenação

Esta classe faz a ordenação estável de um array de Indivíduos por meio de quatro métodos de comparação possíveis: número de rejeições, desvio padrão da matriz residual de Indivíduo, número de enlaces utilizados e nota de fitness. Um array de Indivíduos é passado por referência e esse é devolvido ordenado segundo o tipo de comparação utilizada. Esta classe é utilizada por Seleção afim de criar um ranking.

3.1.14 Classificação

Esta classe atribui valores aos parâmetros de Indivíduo: número de indivíduos semelhantes, nota representativa do fitness e número de indivíduos que o dominam. Esses valores são obtidos por meio de uma amostra, ou seja, o valor é relativo a comparação com os outros indivíduos da amostra. Esta classe é utilizada pelo Algoritmo Genético a fim de gerar informações para a avaliação de desempenho do algoritmo.

3.1.15 Avaliação

Esta classe é utilizada para atribuir um peso que represente a prioridade de cada item de fitness. Um peso é atribuído a cada função objetivo, sendo que seu somatório deve ser igual a 1. Os pesos então são utilizados pelo método de avaliação da classe Classificação a fim de gerar um nota que represente a aptidão de um indivíduo numa amostra.

3.1.16 Dominância de Pareto

Esta classe implementa as regras utilizadas pela definição de dominância de Pareto citadas em 2.8. Retorna verdadeiro caso o primeiro array domine o segundo e falso caso não domine.

3.1.17 Mutação

Esta classe realiza a mutação nos caminhos por meio de TE. Esta implementa os seguintes métodos de mutação: Caminho passando por um ponto fora do caminho original, menor caminho viável, menor caminho que proporcione balanceamento de carga e caminho que evite o enlace mais sobrecarregado do caminho original. Em 3.4.3 os métodos serão explicados com mais detalhes.

3.1.18 Cruzamento

Esta classe implementa quatro tipos de cruzamento: um ponto, dois pontos, uniforme e ortogonal. Neste caso a menor porção do individuo é a rota específica para a demanda. Os filhos resultantes do cruzamento são uma combinação das rotas usadas pelos pais. Em 3.4.2 os métodos serão explicados com mais detalhes.

3.1.19 Comparação de Configurações

Esta classe é usada para criar relatórios que possibilitem examinar o ranking das melhores configurações para cada topologia e número de demandas. A tabela exibe o ranking dos melhores resultados e suas respectivas configurações de seleção, cruzamento e mutação. Foi criada uma versão usando HTML (Hyper Text Markup Language) para exibir uma tabela formatada em browsers para melhor visualização e devido a capacidade de softwares de planilha, como Excel, exibirem os resultados formatados para edição. A Figura 3.3 exibe a tabela em formato HTML para a topologia Mesh com 50 requisições.

Temp_Exec	Geracao	Num_Enlaces	DP_Banda_Res	Rejeicoes	Tipo_Selecao	Tipo_Selecao_Cruzamento	Tipo_Cruzamento	Tipo_Mutacao
0.202	2	82	0.159344	0	1	6	2	3
0.109	1	82	0.199487	0	2	2	3	3
0.187	2	82	0.199487	0	1	5	1	3
0.187	2	82	0.199487	0	5	1	1	3
0.187	2	82	0.199487	0	6	8	2	3
0.188	2	82	0.199487	0	6	8	3	3
0.202	2	82	0.199487	0	4	7	1	3
0.202	2	82	0.199487	0	4	2	3	3
0.203	2	82	0.199487	0	4	2	1	3
0.203	2	82	0.199487	0	6	6	2	3
0.203	2	82	0.199487	0	4	2	2	3
0.218	2	82	0.199487	0	2	2	2	3
0.219	2	82	0.199487	0	5	7	2	3
0.234	2	82	0.199487	0	5	6	1	3

Figura 3.3: Relatório de Configurações por Topologia

Tabela 3.1: Tipos de Seleção

Código	**Tipo de Seleção**
1	Torneio Pareto
2	Fronteira Pareto
3	Torneio
4	Roleta
5	Roleta Ranking
6	Ranking

Tabela 3.2: Tipos de Seleção Cruzamento

Código	**Tipo de Seleção para Cruzamento**
1	Aleatória
2	Torneio Pareto
3	Limite Semelhança
4	Torneio
5	Roleta
6	Roleta Ranking
7	Ranking

A primeira coluna representa o tempo de execução do método de otimização. A segunda coluna representa a geração do melhor indivíduo para a configuração usada. A terceira coluna exibe o somatório do tamanho do caminhos (número de enlaces) para cada demanda. A quarta coluna exibe o desvio padrão da banda residual após o uso do algoritmo de otimização. Estes valores estão entre 0 e 1 representando um valor percentual. A quinta coluna exibe o número de requisições não atendidas. Da sexta à oitava coluna o número representa o tipo de método usado para, respectivamente, a seleção para a próxima geração, a seleção para o cruzamento, o cruzamento e a mutação. As tabelas 3.1, 3.2, 3.3 e 3.4 exibem os códigos com suas respectivas descrições referindo-se aos códigos usados no relatório exibido em 3.3.

3.2 Funcionamento Geral

Esta seção explica o fluxo principal do algoritmo. A fase inicial do algoritmo é o configuração dos parâmetros utilizados na otimização. A Seção 3.3 apresenta uma explicação detalhada

Tabela 3.3: Tipos de Cruzamento

Código	**Tipo de Cruzamento**
1	Dois Pontos
2	Um Ponto
3	Uniforme
4	Ortogonal

Tabela 3.4: Tipos de Mutação

Código	**Tipo de Mutação**
1	Ponto Fora do Caminho
2	Menor Caminho Viável
3	Balanceamento de Carga
4	Desvio de Enlace Crítico

desses parâmetros. Após a fase de configuração, o local onde encontra-se o arquivo de representação da rede é conhecido e carregado assim que necessário.

As requisições são geradas aleatoriamente, porém, apenas no início da otimização. Ou seja, todas as requisições são previamente conhecidas. Com o cenário de simulação definido, estrutura de rede e demandas em memória, parte-se para a geração da população inicial. Ao instanciar a classe Indivíduo pela primeira vez, o método estático 'geraCaminhosPorDemanda' é acionado. Destaca-se aqui que ele é chamado automaticamente apenas na primeira vez que a classe Indivíduo é instanciada. O objetivo desse método é criar várias rotas para cada demanda utilizando o algoritmo de Dijkstra (1959). Outra ressalva necessária é que nesse ponto a semente utilizada para geração de números aleatórias já havia sido inicializada e salva. A partir de então, com várias rotas para cada demanda, um número aleatório entre zero e o número de rotas menos um é escolhido para cada demanda. A rota então é examinada para esse indivíduo. Se a rota não extrapola nenhuma restrição ela é avaliada e inserida para o indivíduo. Caso contrário, soma-se uma rejeição e a rota para a demanda deste individuo permanece vazia. Repete-se o processo até que o número máximo de indivíduos para a população seja alcançado.

Com a população inicial disponível tem início o processo de otimização até que um boa solução seja alcançada ou se atinja o número máximo de gerações. Para este problema, uma boa solução é aquela que não possui rejeições, ou seja, todas as demandas foram atendidas. Até que os critérios de parada sejam atendidos são realizadas as operações de seleção, cruzamento

e mutação. Ao fim da otimização a melhor solução é salva a fim de compará-la a outras soluções que utilizarão tipos de seleção, cruzamento e mutação diferentes, porém para a mesma topologia e requisições. A Figura 3.4 ilustra as etapas da simulação.

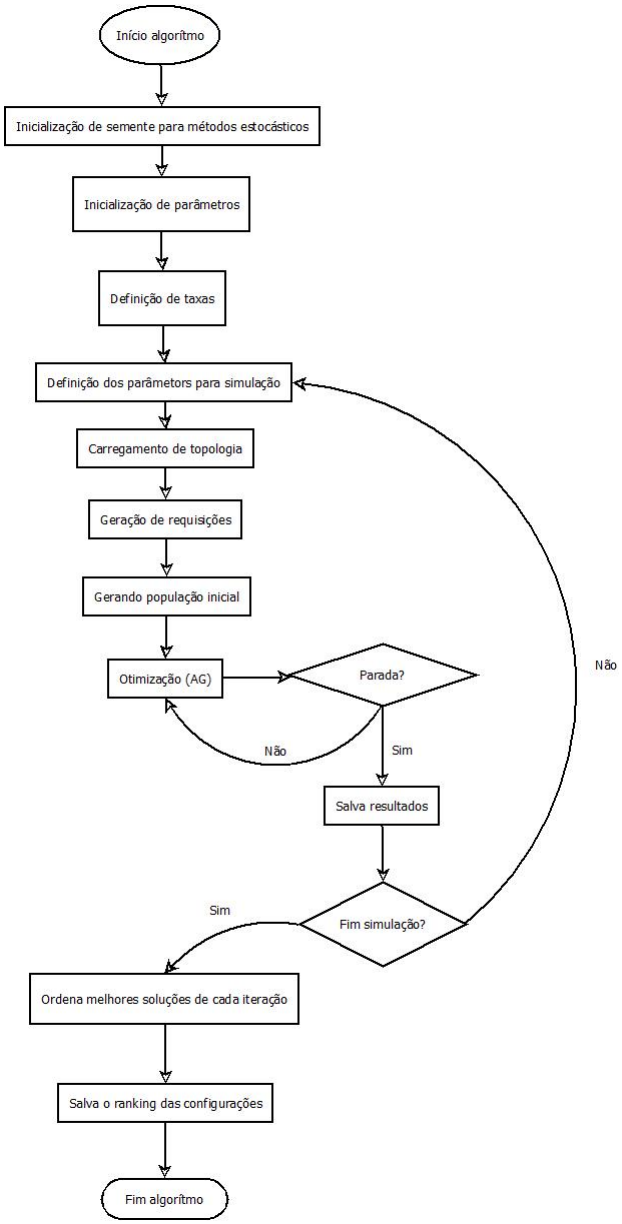

Figura 3.4: Fluxograma da Simulação

Tabela 3.5: Peso das Funções Objetivo

Função objetivo	**Peso (%)**
Número de rejeições (Minimização)	70
Número de enlaces utilizados (Minimização)	20
Desvio padrão da banda residual na rede (Minimização)	10

3.3 Parâmetros de configuração

Essa seção esclarece as possíveis configurações para os tipos de seleção, cruzamento, mutação e suas respectivas taxas utilizadas nas simulações. Inicialmente é necessário definir os pesos das funções objetivo a fim de tornar possível a ordenação das soluções viáveis. A Tabela 3.5 exibe o peso das funções objetivo utilizados nas simulações.

Como todas as funções objetivo são de minimização foi necessário harmonizar e depois normalizar os valores a fim de gerar um ranking.

Em seguida foi definida a banda de cada requisição. Nas simulações utilizou-se o valor 64 e a capacidade do link de todas as redes como 1024. O número de gerações máximo do algoritmo genético foi definido como 10. Definiu-se também uma variável chamada 'limiar-Semelhança' com valor de 60%. Essa variável é utilizada como critério para afirmar se um indivíduos é semelhante a outro. Ou seja, caso a semelhança de dois indivíduos seja maior ou igual a 60%, esses são considerados semelhantes. A Figura 3.2 ilustra um caso de semelhança entre indivíduos.

A próxima etapa é a definição das topologias utilizadas nas simulações, essas são: Carrier, Dora, Mesh, NFS, Ring e Sul. As simulações trabalharam com 50, 100 e 150 requisições.

As taxas de cruzamento e mutação utilizadas no AG foram de respectivamente 50% e 10%. Utilizou-se, também, taxas de mutação em caso de rejeição, com valor de 40% e taxa de rejeição de 5%. Como a mutação utiliza engenharia de tráfego, ela é capaz de criar um rota viável a partir de uma rota vazia e isso possivelmente diminuiria o número de rejeições. A taxa de rejeição é utilizada nos casos em que as mutações não surgem efeito. Uma vez que não a banda disponível para a nova rota, esta é descartada e a rota anterior é restaurada. Utilizando a taxa de rejeição é possível rejeitar um rota que não melhora, liberando banda na rede.

Os parâmetros seguintes tratam do tipo de seleção, cruzamento e mutação utilizados. Optou-se por testar a combinação entre esses de forma a avaliar as melhores combinações. A seção 3.4 aborda detalhadamente os operadores genéticos.

3.4 Operadores Genéticos

Esta seção aborda detalhadamente os operadores genéticos de seleção, cruzamento e mutação implementados na resolução do problema de alocação de rotas em uma rede MPLS.

3.4.1 Seleção

Foram implementadas oito formas de seleção: aleatória, por torneio de Pareto, por distinção, fronteira de Pareto, torneio, roleta, roleta por ranking e ranking. Todos os métodos de seleção recebem um array de indivíduos e um número máximo a ser selecionado.

A seleção aleatória seleciona o número de indivíduos sem elitismo. Logo os indivíduos são escolhidos de forma totalmente aleatória. A Figura 3.5 exibe a lógica do algoritmo.

Data: população, número máximo de indivíduos a ser selecionados
Result: vetor de indivíduos selecionados

```
1 while número de indivíduos selecionados é menor que o máximo de indivíduos a
  ser selecionado do
2 |   escolha um indivíduo da população;
3 |   coloque-o em um vetor auxiliar;
4 |   retire o indivíduo selecionado da população;
5 end while
6 return vetor com os indivíduos selecionados
```

Figura 3.5: Seleção Aleatória

A seleção por torneiro de Pareto seleciona o indivíduo levando em consideração se este é dominante em relação ao outro indivíduo. Um indivíduo A e B são escolhidos aleatoriamente. Se A domina B, então A é selecionado. Se B domina A, então B é selecionado. Caso não haja relação de dominância entre A e B, um deles será escolhido aleatoriamente. A Figura 3.6 exibe a lógica do algoritmo.

A seleção por distinção, chamada de 'limiteSemelhanca', busca selecionar indivíduos diferentes entre si. A Figura 3.2 ilustra a forma de como a comparação é feita. Um limiar é passado como parâmetro e um indivíduo é selecionado aleatoriamente. Esse será o indivíduo

```
Data: população, número máximo de indivíduos a ser selecionados
Result: vetor de indivíduos selecionados
1  while número de indivíduos selecionados é menor que o máximo de indivíduos a
   ser selecionado do
2      escolha o primeiro indivíduo da população;
3      escolha o segundo indivíduo da população;
4      while o primeiro indivíduo é igual ao segundo do
5          escolha novamente o segundo indivíduo
6      end while
7      if o primeiro indivíduo é dominante em relação ao segundo then
8          selecione o primeiro
9      else
10         if o segundo indivíduo é dominante em relação ao primeiro then
11             selecione o segundo
12         else
13             escolha aleatoriamente entre o primeiro e o segundo indivíduo
               selecionado
14         end if
15     end if
16     coloque o indivíduo selecionado em um vetor auxiliar;
17     retire o indivíduo selecionado da população;
18 end while
19 return vetor com os indivíduos selecionados
```

Figura 3.6: Seleção por torneio de Pareto

base da comparação. Este indivíduo será chamado de 'Ib' para evitar ambiguidades. Então, percorre-se o vetor de indivíduos comparando cada indivíduo, i', com o indivíduo Ib. Caso a semelhança entre Ib e i' seja menor que o limiar de semelhança, este será selecionado, se o número máximo de indivíduos selecionados ainda não foi atingido. Pode existir o caso em que não exitam indivíduos distintos suficientes para completar o número máximo de indivíduos a ser selecionados. Neste caso, se o parâmetro 'completaAteMaxInd' for verdadeiro. O limiar de semelhança será aumentado em 10% e o método será novamente executado até completar o número máximo de indivíduos selecionados. A Figura 3.7 exibe a lógica do algoritmo.

A seleção por fronteira de Pareto busca alcançar o número de indivíduos a ser selecionados utilizando a comparação da dominância entre indivíduos, citada em 2.8. Primeiramente é realizada a classificação de dominância. Os indivíduos não dominados são incluídos em um vetor auxiliar. Esses então são retirados da população e novamente é obtido um conjunto de indivíduos não dominados. O processo é repetido até alcançar o número máximo de indivíduos

Data: população, número máximo de indivíduos a ser selecionados, limiar de semelhança, flag para forçar o aumento do limiar de semelhança
Result: vetor de indivíduos selecionados

```
escolha o indivíduo base;
exclua esse indivíduo da população;
for cada indivíduo da população do
    if semelhança entre indivíduo base e indivíduo da população selecionado é menor que o limiar then
        coloque o indivíduo selecionado no vetor auxiliar;
        retire o indivíduo selecionado da população;
    end if
end for
if número máximo de indivíduos não foi atingido e o flag para forçar o aumento do limiar de semelhança é verdadeiro then
    while número máximo de indivíduos selecionados não for atingido do
        execute novamente o algoritmo aumentando o limiar em 10%;
        concatene o resultado ao vetor auxiliar;
    end while
end if
return vetor com os indivíduos selecionados
```

Figura 3.7: Seleção por Distinção

a ser selecionados ou até que não seja possível definir novas fronteiras. Caso a segunda opção ocorra, o método de seleção por distinção é utilizado para completar o número de indivíduos a ser selecionado. Esse processo gera fronteiras de indivíduos as quais mais internas dominam as mais externas. A Figura 3.8 exibe a lógica do algoritmo.

Na seleção por torneio os indivíduos são escolhidos aleatoriamente aos pares e têm seu fitness confrontado segundo uma ordem de prioridade. Primeiro número de rejeições, depois número de enlaces utilizados e por último o desvio padrão da banda residual. Se o indivíduo A possui menor número de rejeições que o indivíduo B, A é selecionado. Se A e B empatam no critério número de rejeições, o segundo critério é avaliado e assim por diante. Caso o empate persista perante os três critérios, um indivíduo do par é selecionado aleatoriamente. A Figura 3.9 exibe a lógica do algoritmo.

Na seleção por roleta, cada indivíduo ocupa um setor da roleta de acordo com o nível de aptidão. Quanto mais bem adaptado, maior será o setor ocupado na roleta e maior a probabilidade de ser selecionado. Esse algoritmo garante que todo indivíduo tem possibilidade de ser selecionado, porém, privilegia os mais bem adaptados. No entanto, quando a aptidão

Data: população, número máximo de indivíduos a ser selecionados
Result: vetor de indivíduos selecionados

```
1  inicializar contador de fronteiras e indivíduos com zero;
2  instanciar estrutura para armazenar índices de indivíduos dominados para cada
   indivíduo (Dominados^i_j);
3  instanciar estrutura para armazenar número de indivíduos que dominam cada
   indivíduo e inicializar com zero(Dominam^i_n);
4  for cada indivíduo da população i^0 do
5      for próximo indivíduo da população i^1 do
6          if i^0 domina i^1 then
7              Dominados^{i^0}_{i^1} ← i^1;
8              Dominam^{i1}_{n←n+1};
9          else
10             if i^1 domina i^0 then
11                 Dominados^{i^1}_{i^0} ← i^0;
12                 Dominam^{i0}_{n←n+1};
13             end if
14         end if
15     end for
16 end for
17 while número máximo de indivíduos selecionados não for atingido do
18     for cada posição no vetor Dominam do
19         if Dominam^i_n, n é igual a zero, ou seja, não é dominado por outro indivíduo
           then
20             insira o indivíduo da posição i no vetor auxiliar de indivíduos
               selecionados;
21             atribua a n o valor (-1) para indicar que o indivíduo na posição i foi
               verificado;
22             if número o máximo de indivíduos foi atingido then
23                 saia dos loops;
24             end if
25         end if
26     end for
27     for cada posição no vetor Dominam do
28         if Dominam^i_n, n é igual a (-1) then
29             atribua a n o valor (-2) para indicar que ele já foi selecionado;
30             for cada posição no vetor Dominados^j do
31                 Dominam^j_{n←n+−1};
32             end for
33         end if
34     end for
35     if o número de indivíduos a ser selecionados não foi atingido then
36         utilize o método de seleção por distinção para alcança-lo;
37         concatene o retorno ao vetor auxiliar de indivíduos selecionados;
38     end if
39 end while
40 return vetor com os indivíduos selecionados
```

Figura 3.8: Seleção por Fronteira de Pareto

Data: população, número máximo de indivíduos a ser selecionados
Result: vetor de indivíduos selecionados

```
1  while número máximo de indivíduos selecionados não for atingido do
2      escolha aleatoriamente um indivíduo i^0 ;
3      escolha outro indivíduo i^1;
4      while i^0 for igual a i^1 do
5          escolha aleatoriamente outro indivíduo i^1 ;
6      end while
7      if i^0 é melhor que i^1 then
8          insira i^0 ao vetor auxiliar de indivíduos selecionados;
9          retire o indivíduo i^0 da população;
10     else
11         if i^1 é melhor que i^0 then
12             insira i^1 ao vetor auxiliar de indivíduos selecionados;
13             retire o indivíduo i^1 da população;
14         else
15             flag ← aleatoriamente verdadeiro ou falso;
16             if flag for verdadeiro then
17                 insira i^0 ao vetor auxiliar de indivíduos selecionados;
18                 retire o indivíduo i^0 da população;
19             else
20                 insira i^1 ao vetor auxiliar de indivíduos selecionados;
21                 retire o indivíduo i^1 da população;
22             end if
23         end if
24     end if
25 end while
26 return vetor com os indivíduos selecionados
```

Figura 3.9: Seleção por Torneio

dos indivíduos é semelhante, ocorre o problema da seleção aleatória. As porções na roleta são praticamente idênticas e isso faz com que a seleção passe a ser aleatória. A roleta é montada utilizando o somatório das notas dos indivíduos. Essas, por sua vez, são obtidas através da harmonização e normalização do vetor de fitness, utilizando pesos para privilegiar algum critério de aptidão. As operações realizadas para criar as notas garantem que a amostra de indivíduos avaliada terá o somatório das notas sempre igual a 100. Este valor será o tamanho da roleta.

Os indivíduos ocupam uma posição relativa na roleta de acordo com a ordenação decrescente da sua aptidão. Um número de 1 a 100, representando a posição de parada da roleta, é sorteado. A precisão do número sorteado é de 2 casas decimais. Ou seja, pode ser sorteado o número como, por exemplo, 56,29. A partir de então soma-se as notas dos indivíduos até encontrar o índice do indivíduo que alcança o valor sorteado para a roleta. O indivíduo selecionado é excluído da roleta e o valor de sua aptidão é subtraído do tamanho dessa. O processo continua até alcançar o número máximo de indivíduos a ser selecionado. A Figura 3.10 exibe a lógica do algoritmo.

Na seleção por roleta baseada em ranking, a proporção ocupada por cada indivíduo diz respeito a sua posição relativa aos outros indivíduos da população, ou seja, seu ranking. De tal forma, o melhor indivíduo ocupará metade da roleta. O segundo ocupará a metade da área restante deixada pelo primeiro no ranking, ou seja, um quarto. O terceiro ocupará a metade da área deixada pelo primeiro e segundo indivíduos no ranking, um oitavo, e assim por diante.

O tamanho da roleta é proporcional ao número de indivíduos a serem alocados. Por exemplo, caso sejam 20 indivíduos, a roleta terá tamanho igual a 1048575, que é igual a $2^{20}-1$, ou seja, dois elevado ao número de indivíduos menos um. Dessa maneira o último colocado no ranking ocupara 1 de espaço na roleta. Assim o número sorteado que representa a posição de parada da roleta pode ser inteiro. Seguindo o exemplo acima, o número sorteado na primeira iteração será de 1 a 1048575. Após o indivíduo ser selecionado na roleta, é necessário retirá-lo da população, e portanto da roleta, para que ele não seja selecionado novamente. Isto é feito retirando o indivíduo selecionado da população e recalculando o tamanho da roleta. Ainda seguindo o exemplo citado anteriormente, após a retirada do indivíduo a roleta terá tamanho igual a 524287, $2^{19}-1$. A Figura 3.11 exibe a lógica do algoritmo.

Na seleção por ranking, a população é ordenada de forma decrescente de acordo com sua aptidão. Os n primeiros classificados são selecionados. A Figura 3.12 exibe a lógica do algoritmo.

Data: população, número máximo de indivíduos a ser selecionados, pesos das aptidões
Result: vetor de indivíduos selecionados

```
1  inicialize o tamanho da roleta T com tamanho 100;
2  inicialize um acumulador acc com 0;
3  ordene a população P;
4  while número máximo de indivíduos selecionados não for alcançado do
5      escolha aleatoriamente um número n entre 1 e o tamanho da roleta T com duas
       casas decimais de precisão;
6      for i ← 0 até T do
7          acc ← acc + nota de P^i;
8          if acc acumulado é maior ou igual ao número n then
9              adicione o indivíduo selecionado P^i ao vetor auxiliar de indivíduos
               selecionados;
10             T ← T - a nota de P^i;
11             T ← o teto de T;
12             exclua P^i da população P;
13             pare o loop;
14         end if
15     end for
16     acc ← 0;
17 end while
18 return vetor com os indivíduos selecionados
```

Figura 3.10: Seleção por método Roleta

3.4.2 Cruzamento

O cruzamento consiste em gerar filhos dados dois ou mais indivíduos como pais. Os filhos possuem características de ambos os pais usados no cruzamento. Neste trabalho, as características passadas aos filhos são as rotas para uma determinada requisição.

Cada pai respeita a restrição de não exceder a banda disponível para um determinado enlace. Porém, ao mesclar rotas de pais diferentes, os filhos gerados podem desrespeitar essa restrição. Portanto é necessário reavaliar a quantidade de banda utilizada nos enlaces e rejeitar requisições que desrespeitem essa restrição.

Caso a avaliação das rotas para cada requisição seja linear, da primeira para a última rota, as últimas demandas serão prejudicadas. Quando a primeira alocação de rota é feita, a rede possui 100% dos recursos. Na segunda alocação, a rede possui menos recursos, uma vez que já existe um rota alocada. Então a ordem de alocação interfere claramente na possibilidade

Data: população, número máximo de indivíduos a ser selecionados
Result: vetor de indivíduos selecionados

```
1  ordene a população P de forma decrescente;
2  inicialize o tamanho da roleta T com valor de 2^P_tamanho − 1;
3  while número máximo de indivíduos selecionados não for alcançado do
4  |  acumulador acc ← 0;
5  |  sorteie um número N entre 1 e o tamanho da roleta T;
6  |  for i ← 0 até o tamanho da população P do
7  |  |  acc ← acc − 2^(i+1);
8  |  |  if acc é maior ou igual a n then
9  |  |  |  selecione o indivíduo P^i e insira-o no vetor de indivíduos selecionados;
10 |  |  |  exclua P^i diminuindo em um o tamanho da população;
11 |  |  |  redefina o tamanho da roleta T com valor de 2^P_tamanho − 1;
12 |  |  |  saia do loop;
13 |  |  end if
14 |  end for
15 end while
16 return vetor com os indivíduos selecionados
```

Figura 3.11: Seleção por método Roleta Ranking

Data: população, número máximo de indivíduos a ser selecionados
Result: vetor de indivíduos selecionados

```
1  ordene a população P de forma decrescente;
2  insira os n primeiros indivíduos no vetor de indivíduos selecionados, onde n é o
   número de indivíduos a ser selecionados;
3  return vetor com os indivíduos selecionados
```

Figura 3.12: Seleção por método Ranking

de rejeição da requisição. Para resolver esse problema, a ordem de avaliação das rotas é aleatória. Assim a prioridade de avaliação das rotas e igual para todas as requisições.

Quatro formas de cruzamento foram implementadas: um ponto, dois pontos, uniforme e ortogonal.

No cruzamento de um ponto, um número n, entre um e a quantidade de requisições, R, menos um é sorteado. O zero não pode ser sorteado, uma vez que isso acarretaria na geração de clones. Então n será o ponto de interseção.

O $filho^0$ é formado pelas rotas do pai^0 de 0 a n, excluindo n, concatenado com as rotas do pai^1, de n, inclusive, até $R-1$, onde R é o total de requisições. O $filho^1$ é formado pelas rotas do pai^1 de 0 a n, excluindo n, concatenado com as rotas do pai^0, de n, inclusive, até $R-1$. A Figura 3.13 exibe a lógica do algoritmo.

Data: pai^0, pai^1, rede, requisições
Result: vetor de indivíduos

instancie o ponto de cruzamento n com um número entre 1 e o número de requisições menos um;
for cada requisição^x **do**
  if $x < n$ **then**
    $filho^0 \leftarrow rota^x$ do pai^0;
  else
    $filho^0 \leftarrow rota^x$ do pai^1;
  end if
end for
reavalie as rotas do $filho^0$ para garantir que não desrespeitem as restrições da rede;
insira o $filho^0$ no vetor auxiliar $filhos$;
for cada requisição^x **do**
  if $x < n$ **then**
    $filho^1 \leftarrow rota^x$ do pai^1;
  else
    $filho^1 \leftarrow rota^x$ do pai^0;
  end if
end for
reavalie as rotas do $filho^1$ para garantir que essas não desrespeitem as restrições da rede;
insira o $filho^1$ no vetor auxiliar $filhos$;
return vetor com indivíduos filhos

Figura 3.13: Cruzamento de um ponto

No cruzamento de dois pontos, o conjunto de rotas é repartido em três partes. Da posição inicial ao primeiro ponto de cruzamento, do primeiro ponto de cruzamento ao segundo e do segundo ponto em diante. O primeiro filho recebe a primeira e última parte do pai^0, e a segunda parte do pai^1. O segundo filho recebe a primeira e última parte do pai^1, e a segunda parte do pai^0. Para evitar que os filhos sejam clones dos pais e que o cruzamento de dois pontos seja reduzido ao cruzamento de um ponto, algumas restrições devem ser respeitadas.

Dado que R^t é o número total de requisições, $R_q = \{r_0, r_1, r_2, ..., r_n\}$ o conjunto de índices para as requisições, n^0 o primeiro ponto de cruzamento e n^1 o segundo:

- $n^0 \in [r_1, r_{n-1}]$;
- $n^1 \in (n^0, r_n]$.

Respeitando as restrições acima sempre haverão três seguimentos para formar o conjunto de rotas de cada indivíduo. A Figura 3.14 exibe a lógica do algoritmo.

No cruzamento uniforme, uma máscara de bits aleatória, m, é definida medindo o tamanho referente ao número de requisições. O $filho^0$ é obtido considerando que bits 0 da máscara m representam rotas do pai^0 e bits 1 representam rotas do pai^1. O $filho^1$ é obtido considerando que bits 0 representam rotas do pai^1 e bits 1 representam rotas do pai^0. A Figura 3.15 exibe a lógica do algoritmo.

No cruzamento ortogonal o número de pais determina o número de segmentos de cruzamento. Por exemplo, caso o cruzamento ortogonal seja realizado com 3 pais, existirão 3 segmentos de cruzamento. Então são geradas as permutações na qual cada pai ocupa apenas um segmento por vez. Desta forma, para três pais, a, b e c; e segmentos s^1, s^2, e s^3; as combinações seriam conforme a Figura 3.16.

Como o número de possibilidades de filhos é igual ao fatorial do número de pais, este trabalho se detêm a utilizar três pais a fim de evitar a explosão do número de filhos, o que tornaria o método extremamente lento. Além disso, dos seis indivíduos gerados, apenas os três melhores são retornados.

O intervalo dos segmentos é baseado no quociente da divisão inteira de número de requisições pelo número de segmentos. Por exemplo, para 20 requisições e 3 segmentos, o quociente é 6. Logo, os intervalos são: $[0, 6)$, $[6, 12)$ e $[12, 19)$. A Figura 3.17 exibe a lógica do algoritmo.

Data: pai^0, pai^1, rede, requisições
Result: vetor de indivíduos

instancie o ponto de cruzamento n^0 com um número entre 1 e o número de requisições menos dois;
instancie o ponto de cruzamento n^1 com um número entre n^0 e o número de requisições menos um;
for cada requisição^x **do**
    if $x < n^0$ **then**
        $filho^0 \leftarrow rota^x$ do pai^0;
    else
        if $x < n^1$ **then**
            $filho^0 \leftarrow rota^x$ do pai^1;
        else
            $filho^0 \leftarrow rota^x$ do pai^0;
        end if
    end if
end for
reavalie as rotas do $filho^0$ para garantir que não desrespeitem as restrições da rede;
insira o $filho^0$ no vetor auxiliar $filhos$;
for cada requisição^x **do**
    if $x < n^0$ **then**
        $filho^1 \leftarrow rota^x$ do pai^1;
    else
        if $x < n^1$ **then**
            $filho^1 \leftarrow rota^x$ do pai^0;
        else
            $filho^1 \leftarrow rota^x$ do pai^1;
        end if
    end if
end for
reavalie as rotas do $filho^1$ para garantir que não desrespeitem as restrições da rede;
insira o $filho^1$ no vetor auxiliar $filhos$;
return vetor com indivíduos filhos

Figura 3.14: Cruzamento de dois pontos

Data: pai^0, pai^1, rede, requisições
Result: vetor de indivíduos

instancie a máscara de bits m;
for cada bit da máscara m^x **do**
  if m^x é igual a 0 **then**
    $filho^0 \leftarrow rota^x$ do pai^0;
  else
    $filho^0 \leftarrow rota^x$ do pai^1;
  end if
end for
reavalie as rotas do $filho^0$ para garantir que não desrespeitem as restrições da rede;
insira o $filho^0$ no vetor auxiliar $filhos$;
for cada bit da máscara m^x **do**
  if m^x é igual a 0 **then**
    $filho^1 \leftarrow rota^x$ do pai^1;
  else
    $filho^1 \leftarrow rota^x$ do pai^0;
  end if
end for
reavalie as rotas do $filho^1$ para garantir que não desrespeitem as restrições da rede;
insira o $filho^1$ no vetor auxiliar $filhos$;
return vetor com indivíduos filhos

Figura 3.15: Cruzamento de dois pontos

s^1	s^2	s^3		
a	b	c	→	$filho^1$
a	c	b	→	$filho^2$
b	a	c	→	$filho^3$
b	c	a	→	$filho^4$
c	a	b	→	$filho^5$
c	b	a	→	$filho^6$

Figura 3.16: Possibilidades do cruzamento ortogonal com três pais

3.4.3 Mutação

A mutação consiste em gerar perturbações nas soluções afim de explorar melhor o espaço de busca. Porém, movimentos completamente aleatórios tendem a deteriorar as soluções. Partindo desse princípio, a mutação neste trabalho também age como uma busca local, melhorando as soluções ao mesmo tempo que explora outros pontos no espaço de busca.

Data: vetor de indivíduos pais, rede, requisições, número máximo de filhos n
Result: vetor de indivíduos

```
1 gera segmentos de cruzamento utilizando o número de requisições e o tamanho do
  vetor de pais;
2 possibilidades de cruzamento ← permutação do número de elementos do vetor de
  pais;
3 for cada possibilidade^i de cruzamento do
4 |   vetor de indivíduos filhos ← individuo gerado com possibilidades^i utilizando os
  |   segmentos de cruzamento;
5 end for
6 vetor indivíduos filhos ← n melhores filhos gerados;
7 return vetor com indivíduos filhos
```

Figura 3.17: Cruzamento Ortogonal

Quatro tipos de mutação foram implementadas: rota passando por um ponto aleatório, menor rota viável, menor rota considerando o balanceamento de carga e rota que evita o enlace mais sobrecarregado da rota original.

Em todos os casos, antes da alteração da rota original, os recursos utilizados por esta são reintegrados à rede, caso a rota não seja vazia. Outro passo em comum a todos os métodos é a retirada dos recursos gastos pela demanda a ser alocada em toda a rede. Desta forma garante-se que caso o grafo representativo da rede continue conexo, haverá banda suficiente para atender a requisição.

Na primeira estratégia de mutação, após os passos comuns citados acima, um nó fora da rota original é escolhido. A matriz representativa da banda residual é então transformada em uma matriz de adjacência. Essa transformação faz com que enlaces com banda residual maiores que 0 passem a ser 1. Então cada posição da matriz é multiplicada por um número inteiro suficientemente grande. Os enlaces adjacentes ao nó escolhido são então reduzidos ao valor 1, enquanto os demais enlaces permanecem com custo elevado. Na sequencia o algoritmo de Dijkstra é submetido a matriz resultante. Caso o nó escolhido esteja a uma distancia razoável, onde distância se refere ao número de saltos, do no emissor ao receptor, haverá um chance considerável que ele passe a integrar a nova rota. Caso contrário a mutação se comportará buscando a menor distância entre dois nós. A Figura 3.18 exibe a lógica do algoritmo.

```
Data: caminho original, banda gasta pela requisição, matriz de banda residual,
      matriz de atrasos, taxa de rejeição
Result: novo caminho
1  if caminho original não é vazio then
2  |  recomponha a banda utilizada pelo caminho original à matriz de banda residual;
3  |  conjunto viável ← (nós da rede) - (nós do caminho original);
4  |  if o conjunto viável for vazio then
5  |  |  ponto ← um nó qualquer da rede;
6  |  else
7  |  |  ponto ← um nó do conjunto viável;
8  |  end if
9  end if
10 retire a banda gasta pela requisição de toda a rede para encontrar os enlaces
   viáveis;
11 matriz aux ← transformação da matriz de banda residual em matriz de adjacências;
12 matriz aux ← matriz aux * 10000;
13 for cada enlace da matriz aux do
14 |  if o enlace é adjacente ao ponto escolhido then
15 |  |  custo do enlace ← 1;
16 |  end if
17 end for
18 utilize Dijkstra na matriz aux, e os nós inicial e final do caminho original;
19 if Dijkstra obteve sucesso then
20 |  novo caminho ← caminho gerado por Dijkstra;
21 |  novo caminho ← custo calculado usando a matriz de atrasos;
22 else
23 |  num ← número aleatório de 0 a 99;
24 |  if num > taxa de rejeição then
25 |  |  novo caminho ← vazio;
26 |  end if
27 end if
28 return novo caminho
```

Figura 3.18: Mutação: rota passando por um ponto aleatório

Na segunda estratégia de mutação, após as operações para garantir a viabilidade do caminho, é aplicado o algoritmo de Dijkstra na matriz adjacente resultante da transformação da matriz de banda residual. A Figura 3.19 exibe a lógica do algoritmo.

Data: caminho original, banda gasta pela requisição, matriz de banda residual, matriz de atrasos, taxa de rejeição
Result: novo caminho

```
1  if caminho original não é vazio then
2  |  recomponha a banda utilizada pelo caminho original à matriz de banda residual;
3  end if
4  retire a banda gasta pela requisição de toda a rede para encontrar os enlaces
   viáveis;
5  transforme a matriz de banda residual em uma matriz de adjacências;
6  utilize Dijkstra na matriz resultante da transformação da matriz de banda residual,
   e os nós inicial e final do caminho original;
7  if Dijkstra obteve sucesso then
8  |  novo caminho ← caminho gerado por Dijkstra;
9  |  novo caminho ← custo calculado usando a matriz de atrasos;
10 else
11 |  num ← número aleatório de 0 a 99;
12 |  if num > taxa de rejeição then
13 |  |  novo caminho ← vazio;
14 |  end if
15 end if
16 return novo caminho
```

Figura 3.19: Mutação: menor rota viável

Na terceira estratégia, a matriz representativa da capacidade da rede é subtraída da matriz de banda residual, resultando na matriz da banda utilizada pelas rotas. Dessa forma, os menores valores indicam enlaces os quais possuem maior banda disponível, enquanto os valores mais altos indicam que a banda restante é escassa. O algoritmo de Dijkstra é então aplicado a matriz banda utilizada e as rotas geradas tendem a passar por enlaces com maior banda disponível. A Figura 3.20 exibe a lógica do algoritmo.

A quarta estratégia assemelha-se a primeira e a terceira estratégias. O enlace mais sobrecarregado do caminho original será o escolhido para ser punido. Para verificar o enlace mais sobrecarregado, basta verificar a proporção entre banda residual e capacidade total de cada enlace e então subtrair o resultado da matriz de adjacências que representa a interligação da topologia. Então percorre-se a matriz considerando o caminho original em busca do enlace mais utilizado. A Figura 3.21 ilustra a operação citada acima.

```
Data: caminho original, banda gasta pela requisição, matriz de banda residual,
      matriz de atrasos, taxa de rejeição
Result: novo caminho
if caminho original não é vazio then
    recomponha a banda utilizada pelo caminho original à matriz de banda residual;
end if
retire a banda gasta pela requisição de toda a rede para encontrar os enlaces
viáveis;
matriz banda utilizada leftarrow matriz capacidade - matriz de banda residual;
utilize Dijkstra na matriz de banda utilizada, e os nós inicial e final do caminho
original;
if Dijkstra obteve sucesso then
    novo caminho ← caminho gerado por Dijkstra;
    novo caminho ← custo calculado usando a matriz de atrasos;
else
    num ← número aleatório de 0 a 99;
    if num > taxa de rejeição then
        novo caminho ← vazio;
    end if
end if
return novo caminho
```

Figura 3.20: Mutação: balanceamento de carga

A matriz de banda residual transformada em matriz de adjacências também é utilizada neste método. Está é multiplicada por um número grande, punindo todos os enlaces. Os enlaces utilizados na rota original são poupados da punição e seu custo é rebaixado a 1. O enlace mais utilizado, então, sofre um punição. Seu custo é configurado com um valor suficientemente grande para que ele não seja atrativo, se tornando o enlace com maior custo da rede. Após todas essas operações o algoritmo de Dijkstra é aplicado. O resultado são rotas semelhantes a rotas originais, porém, excluindo o enlace crítico da rota original. A Figura 3.22 exibe a lógica do algoritmo.

3.5 Testes

As topologias utilizadas neste trabalho foram Carrier, Dora, Mesh, NFS, Ring e Sul. Sua escolha foi motivada pela ampla utilização em trabalhos correlatos. A Figura 3.23 exibe as topologias citadas acima.

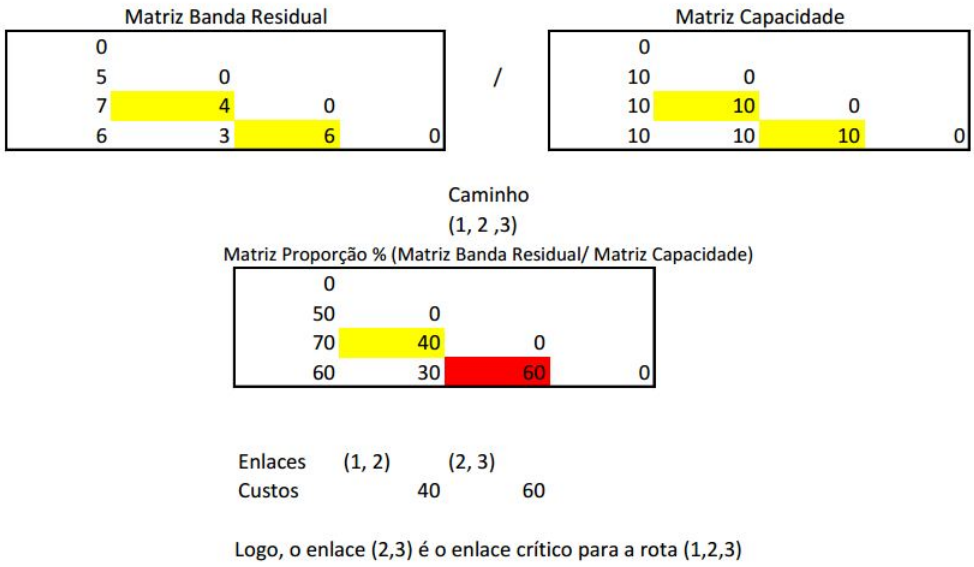

Figura 3.21: Cálculo de enlace mais sobrecarregado para uma rota

Os testes consistiram em verificar quais as melhores combinações de seleção, cruzamento e mutação. Os parâmetros avaliados em ordem de prioridade foram:

1. Número de rejeições
2. Número de enlaces utilizados
3. Desvio padrão da matriz de banda residual
4. Tempo de execução

Cada simulação consistia em executar o algoritmo genético 8064 vezes. Este número é resultado do produto dos parâmetros utilizados nos cenários avaliados. Os parâmetro e valores adotados foram:

- Pesos das funções objetivo
  * Minimização das rejeições (70%)
  * Minimização do número de enlaces (20%)
  * Minimização do desvio padrão da banda residual (10%)

- Número de gerações para algoritmo genético
  * 10

```
Data: caminho original, banda gasta pela requisição, matriz de banda residual,
      matriz de atrasos, taxa de rejeição
Result: novo caminho
1  if caminho original não é vazio then
2  |  enlace critico ← enlace mais utilizado do caminho original;
3  |  recomponha a banda utilizada pelo caminho original à matriz de banda residual;
4  end if
5  retire a banda gasta pela requisição de toda a rede para encontrar os enlaces
   viáveis;
6  matriz aux ← transformação da matriz de banda residual em matriz de adjacências;
7  matriz aux ← matriz aux * 100;
8  if caminho original não é vazio then
9  |  matriz aux ← subtração de 99 para cada enlace presente o caminho original;
10 |  matriz aux ← punição para o enlace crítico no valor de 9999;
11 end if
12 utilize Dijkstra na matriz aux, e os nós inicial e final do caminho original;
13 if Dijkstra obteve sucesso then
14 |  novo caminho ← caminho gerado por Dijkstra;
15 |  novo caminho ← custo calculado usando a matriz de atrasos;
16 else
17 |  num ← número aleatório de 0 a 99;
18 |  if num > taxa de rejeição then
19 |  |  novo caminho ← vazio;
20 |  end if
21 end if
22 return novo caminho
```

Figura 3.22: Mutação: desvio de enlace crítico da rota original

- Número máximo de indivíduos
  * 20
- Limiar de semelhança
  * 60%
- Taxa de cruzamento
  * 50%
- Taxa de mutação
  * 10%

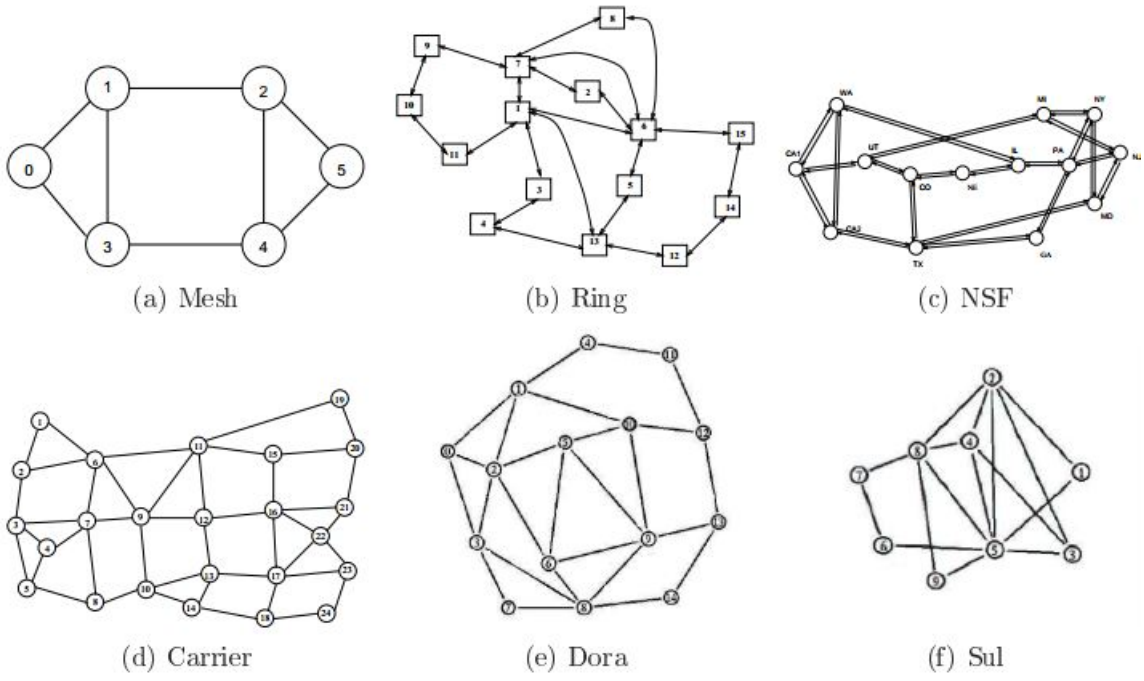

Figura 3.23: Topologias utilizadas em testes

- Taxa de mutação para rejeições
  * 40%
- Taxa de rejeição
  * 5%
- Topologias - (capacidade do enlace 1024)
  * Carrier
  * Dora
  * Mesh
  * Nfs
  * Ring
  * Sul
- Banda utilizada nas requisições
  * 64
- Número de requisições
  * 100
  * 150

- Tipos de seleção
  * Torneio Pareto
  * Fronteira Pareto
  * Torneio
  * Roleta
  * Roleta Ranking
  * Ranking

- Tipos de seleção para cruzamento
  * Aleatória
  * Torneio Pareto
  * Limite Semelhança
  * Torneio
  * Roleta
  * Roleta Ranking
  * Ranking

- Tipos de cruzamento
  * Dois Pontos
  * Um Ponto
  * Uniforme
  * Ortogonal

- Tipos de mutação
  * Ponto Fora do Caminho Original
  * Menor Caminho Viável
  * Balanceamento de Carga
  * Desvio do Enlace Crítico

As simulações foram realizadas em um notebook com 6gb de memória ram, processador core i5 com 4 threads e hd de 750 gb de 7500 rpm. O tempo da simulação foi medido em milissegundos. O contador de tempo foi inicializado antes da entrada na rotina de otimização e finalizado depois dessa. Assim, as rotinas de configuração de parâmetros iniciais, carregamento da topologia e tratamento dos resultados não foram contabilizadas.

A cada iteração dos testes, um arquivo com as melhores soluções resultante da rotina de otimização para o conjunto de parâmetros é salvo. O arquivo contêm informações referentes aos parâmetros utilizados na otimização, grupo de requisições utilizado, semente geradora para

números aleatórios e o conjunto de soluções não-dominadas classificadas de forma decrescente. Ou seja, da melhor para a pior.

Findo os testes para o par topologia e número de requisições, uma tabela é salva contendo o melhor resultado para cada configuração utilizada, formando um ranking. Dois arquivos de ranking são criados para cada par topologia e número de requisições. Um em formato texto livre e o outro em formato HTML. Desta forma é possível visualizar o ranking em formato HTML no browser e manuseá-lo no Excel para tratar os dados e obter informações mais facilmente.

3.6 Resultados

Esta seção aborda os resultados obtidos nos testes. Primeiramente é abordado a metodologia utilizada para tratar os resultados. Em seguida serão comparados os métodos de seleção, seleção para cruzamento, cruzamento e mutação para cada par topologia e número de requisições. Então serão abordadas as comparações entre os métodos considerando todos os pares topologia e número de requisições.

As soluções foram classificadas seguindo a ordem de prioridade: menor número de rejeições, menor número de enlaces utilizados, menor desvio padrão de banda residual e menor tempo de execução; gerando um ranking. A cada posição do ranking foi atribuída uma pontuação, onde o primeiro colocado recebe 672 pontos e o último recebe algo próximo de 1 ponto. A Figura 3.24 ilustra a distribuição de pontos. Um ranking foi construído para cada par topologia número de requisições.

Cada posição no ranking representa uma combinação única de métodos. Esses recebem os pontos previstos para cada posição no ranking. O somatório dos pontos para cada método é então normalizado, tornado possível a comparação entre os grupos de métodos de seleção, seleção para cruzamento, cruzamento e mutação.

Nos testes realizados com a topologia Carrier e 100 requisições não houve rejeição para nenhum agrupamento de métodos. A melhor solução ocupou 43% dos recursos da rede. Essa foi construída utilizando a seleção por roleta, seleção para cruzamento por roleta ranking, cruzamento de um ponto e mutação por menor caminho viável.

Nos testes realizados com a topologia Carrier e 150 requisições, as rejeições variaram de 14 a 28. A melhor solução ocupou 58% dos recursos da rede. Essa foi construída utilizando a seleção por fronteira de Pareto, seleção para cruzamento aleatória, cruzamento ortogonal e mutação por menor caminho viável.

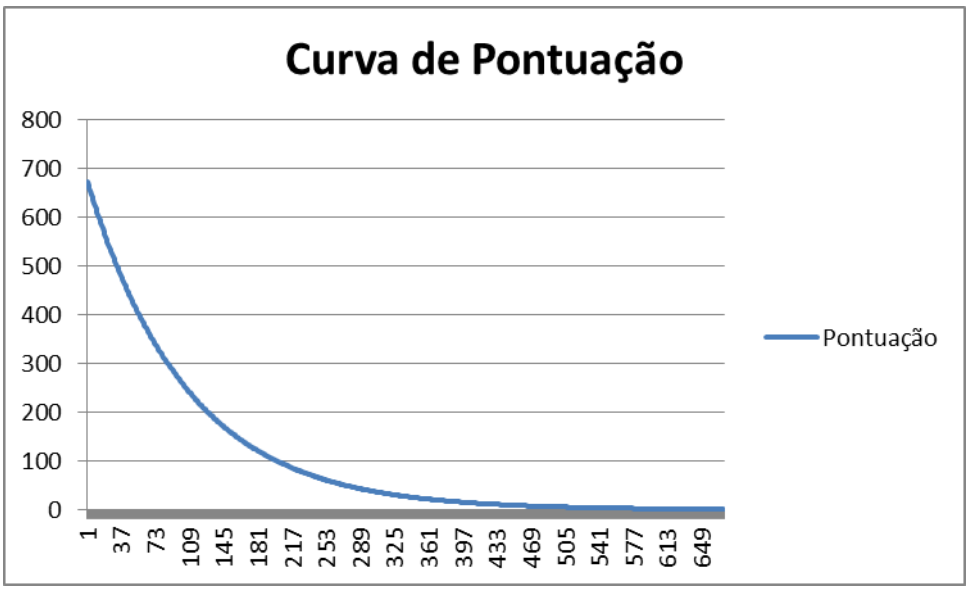

Figura 3.24: Curva de Pontuação

Considerando os métodos de seleção para 100 requisições, torneio de Pareto, torneio e fronteira de Pareto ficaram muito próximos. Nos métodos de seleção para cruzamento, roleta ranking foi superior, seguido de ranking. Para os métodos de cruzamento a melhor pontuação foi do método de dois pontos seguido do uniforme. Nos métodos de mutação, menor caminho viável foi superior.

Para 150 requisições, as seleções por roleta-ranking e ranking foram superiores. Nos métodos de seleção para cruzamento, torneio foi superior seguido de ranking e roleta-ranking. Para os métodos de cruzamento a melhor pontuação foi do método ortogonal seguido do de dois pontos. Nos métodos de mutação, menor caminho viável foi superior. A Figura 3.25 exibe os resultados para a topologia Carrier.

Nos testes realizados com a topologia Dora e 100 requisições, as rejeições variaram de 0 a 3. A melhor solução ocupou 56% dos recursos da rede. Essa foi construída utilizando a seleção roleta-ranking, seleção para cruzamento torneio de Pareto, cruzamento ortogonal, e mutação por balanceamento de carga.

Nos testes realizados com a topologia Dora e 150 requisições, as rejeições variaram de 11 a 35. A melhor solução ocupou 79% dos recursos da rede. Essa foi construída utilizando a seleção roleta-ranking, seleção para cruzamento roleta-ranking, cruzamento ortogonal e mutação por menor caminho viável.

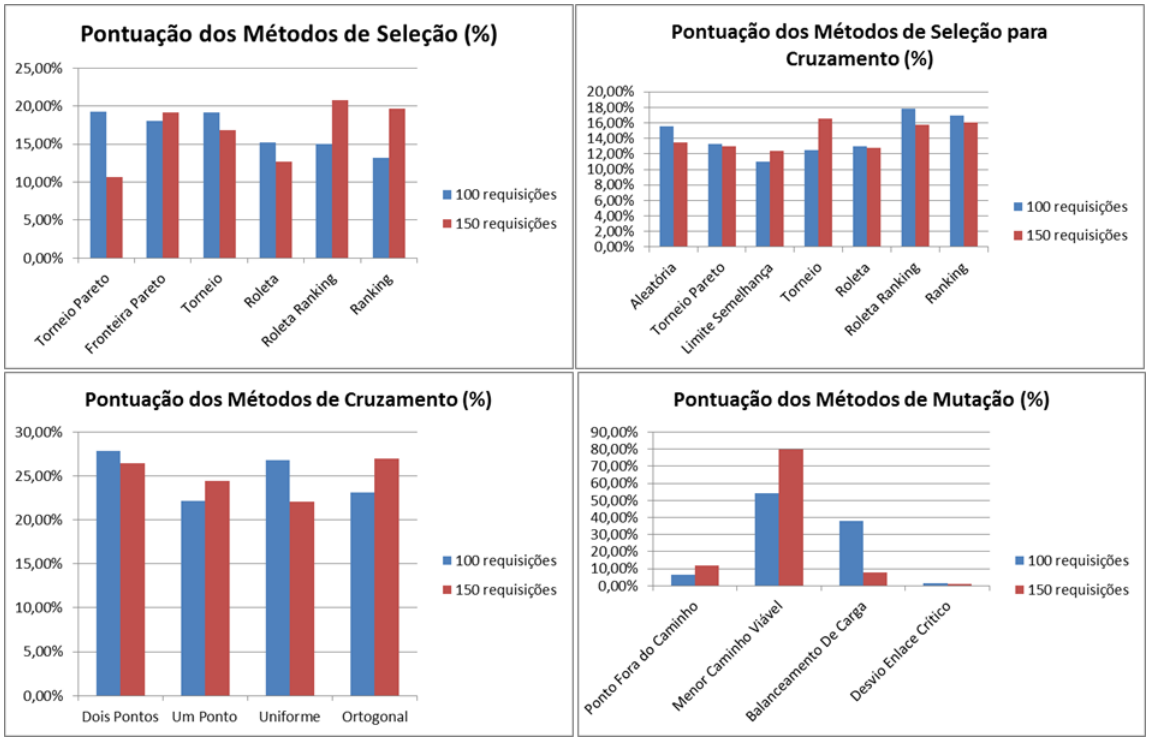

Figura 3.25: Resultados para Topologia Carrier

Considerando os métodos de seleção para 100 requisições, roleta-ranking e ranking ficaram muito próximos. Nos métodos de seleção para cruzamento, ranking e torneio foram superiores. Para os métodos de cruzamento, uniforme e dois pontos ficaram muito próximos, seguido do de um ponto. Nos métodos de mutação, balanceamento de carga se mostrou muito superior aos demais.

Para 150 requisições, a seleção por roleta-ranking foi superior. Nos métodos de seleção para cruzamento, ranking, roleta-ranking e torneio estão próximos, porém ranking é superior. Para os métodos de cruzamento, ortogonal foi superior. Para os métodos de cruzamento, balanceamento de carga foi superior. A Figura 3.26 exibe os resultados para a topologia Dora.

Nos testes realizados com a topologia Mesh e 100 requisições, as rejeições variaram de 25 a 34. A melhor solução ocupou 72% dos recursos da rede. Essa foi construída utilizando seleção por ranking, seleção para cruzamento torneio de Pareto, cruzamento ortogonal e mutação por balanceamento de carga.

Nos teste realizados com a topologia Mesh e 150 requisições, as rejeições variaram de 56 a 80. A melhor solução ocupou 97% dos recursos da rede. Essa foi construída utilizando

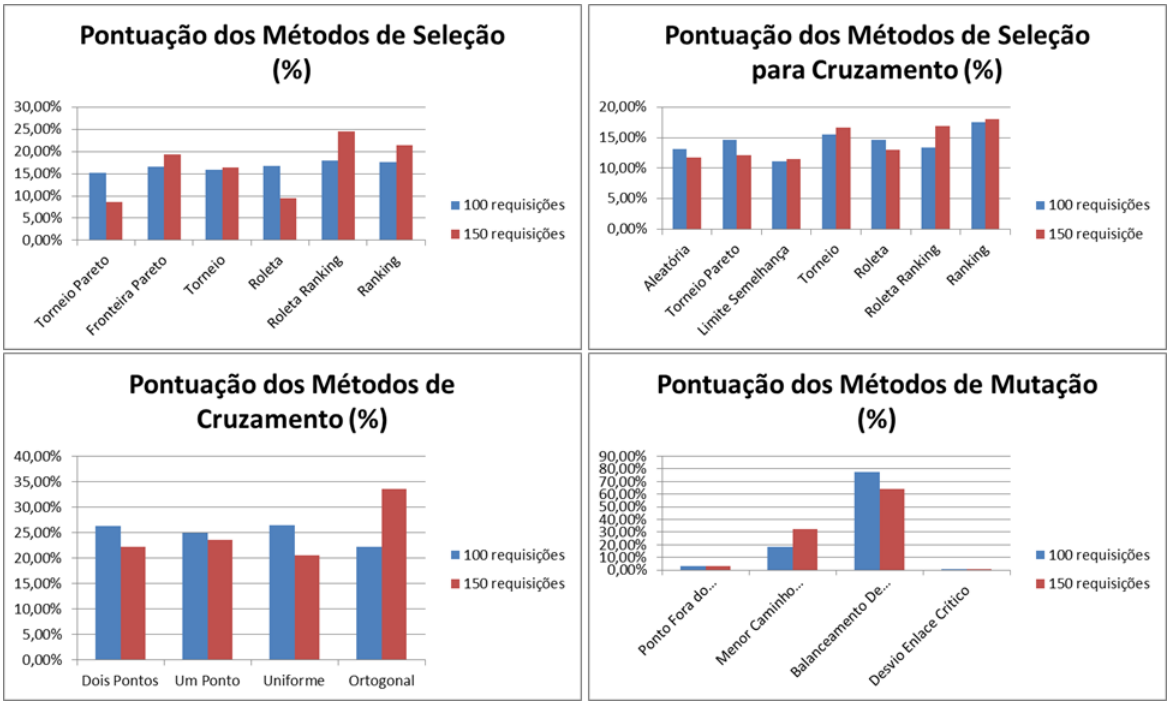

Figura 3.26: Resultados para Topologia Dora

seleção por roleta-ranking, seleção para cruzamento ranking, cruzamento uniforme e mutação por menor caminho viável.

Considerando os métodos de seleção para 100 requisições, ranking foi superior, seguido de roleta-ranking. Na seleção para cruzamento, ranking e torneio ficaram próximas, seguido de roleta-ranking. O cruzamento ortogonal foi superior, seguido do cruzamento de um ponto. A mutação por balanceamento de carga foi bem superior aos demais métodos.

Considerando os métodos de seleção para 150 requisições, ranking e roleta-ranking ficaram muito próximos, sendo ranking superior. Na seleção para cruzamento, roleta-ranking foi superior aos demais métodos. O cruzamento ortogonal foi bem superior aos demais métodos. Nas mutações, balanceamento de carga e menor caminho viáveis ficaram praticamente empatados, sendo balanceamento de carga levemente superior. A Figura 3.27 exibe os resultados para a topologia Mesh.

Nos testes realizados com a topologia Nfs e 100 requisições, as rejeições variaram de 0 a 8. A melhor solução ocupou 64% dos recursos da rede. Essa foi construída utilizando seleção torneio de Pareto, seleção para cruzamento ranking, cruzamento de dois pontos e mutação por balanceamento de carga.

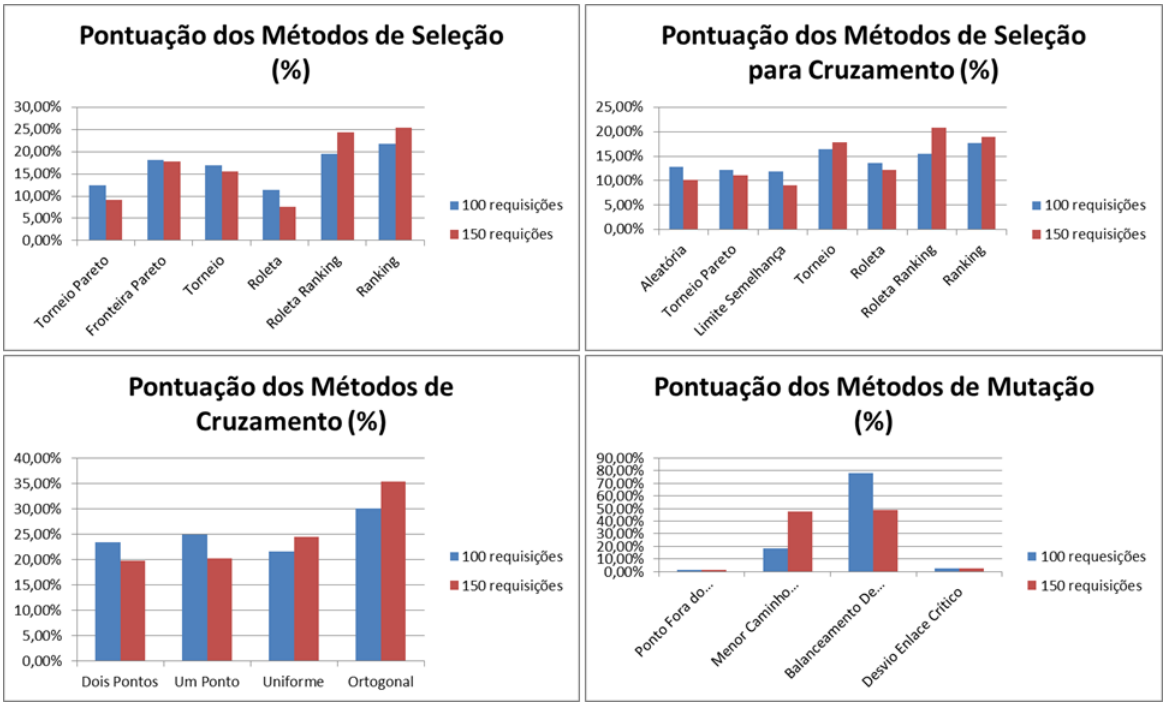

Figura 3.27: Resultados para Topologia Mesh

Nos testes realizados com a topologia Nfs e 150 requisições, as rejeições variaram de 20 a 48. A melhor solução ocupou 82% dos recursos da rede. Essa foi construída utilizando seleção por ranking, seleção para cruzamento ranking, cruzamento ortogonal e mutação por balanceamento de carga.

Considerando 100 requisições, todos os métodos de seleção ficaram muito próximos, porém, roleta-ranking foi superior. Na a seleção para cruzamento, a seleção aleatória e ranking ficaram praticamento empatados, mas ranking foi superior. Para os métodos de cruzamento, um ponto foi superior, seguido do cruzamento uniforme. Nos métodos de mutação, balanceamento de carga foi muito superior.

Considerando 150 requisições, roleta-ranking foi o melhor método de seleção e seleção para cruzamento, seguido de roleta-ranking. Para os cruzamentos, dois pontos foi superior, porém, muito próximo do cruzamento ortogonal. Para as mutação, balanceamento de carga foi muito superior. A Figura 3.28 exibe os resultados para a topologia Nfs.

Nos testes realizados com a topologia Ring e 100 requisições, as rejeições variaram entre 8 e 17. A melhor solução ocupou 66% dos recursos da rede. Essa foi construída utilizando seleção por torneio de Pareto, seleção para cruzamento por ranking, cruzamento ortogonal e mutação por balanceamento de carga.

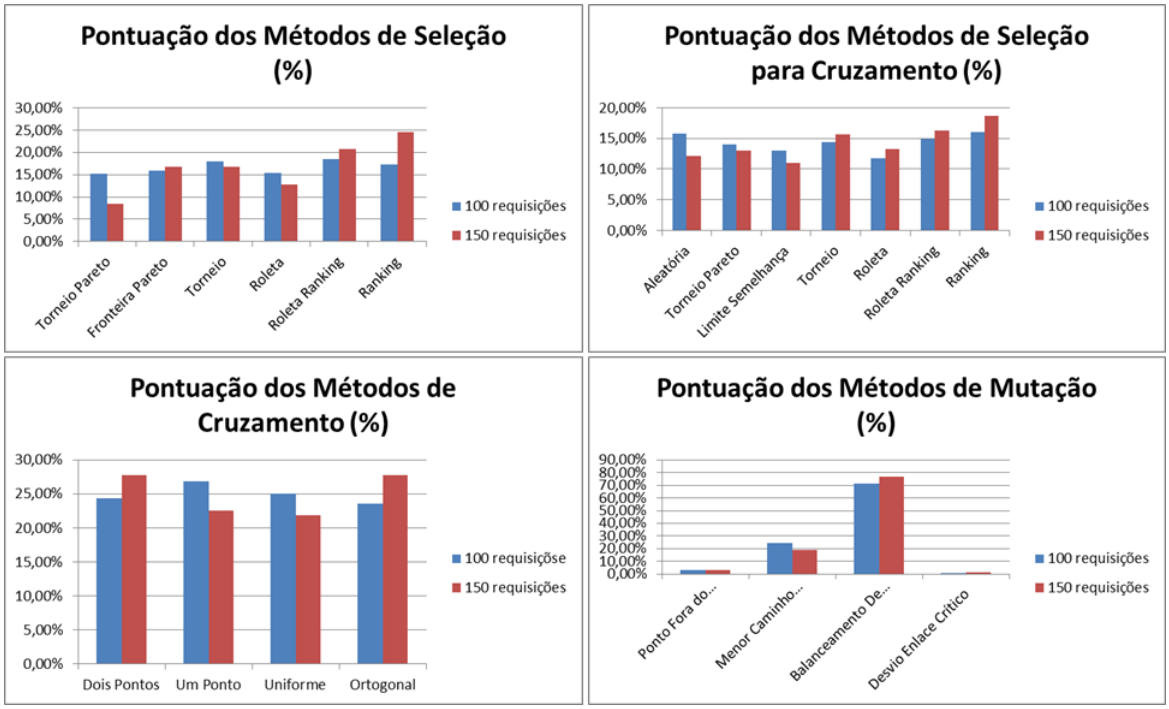

Figura 3.28: Resultados para Topologia Nfs

Nos testes realizados coma topologia Ring e 150 requisições, as rejeições variaram entre 30 e 54. A melhor solução ocupou 81% dos recursos da rede. Essa foi construída utilizando seleção por ranking, seleção para cruzamento por torneio, cruzamento ortogonal e mutação por balanceamento de carga.

Considerando 100 requisições, o melhor método de seleção foi roleta-ranking, seguido de ranking e fronteira de Pareto. Ranking foi superior nos métodos de seleção para cruzamento, seguido de torneio e roleta-ranking. O cruzamento ortogonal foi superior entre os métodos de cruzamento. Para as mutações, balanceamento de carga foi muito superior.

Considerando 150 requisições, o melhor método de seleção e seleção para cruzamento foi o ranking, seguido do método roleta-ranking. O cruzamento ortogonal foi superior para os métodos de cruzamento. Para as mutações, balanceamento de carga foi muito superior. A Figura 3.29 exibe os resultados para a topologia Ring.

Nos testes realizados com a topologia Sul e 100 requisições, as rejeições variaram de 13 a 16. A melhor solução ocupou 51% dos recursos da rede. Essa foi construída utilizando seleção por ranking, seleção para cruzamento por limite de semelhança, cruzamento de dois pontos e mutação por balanceamento de carga.

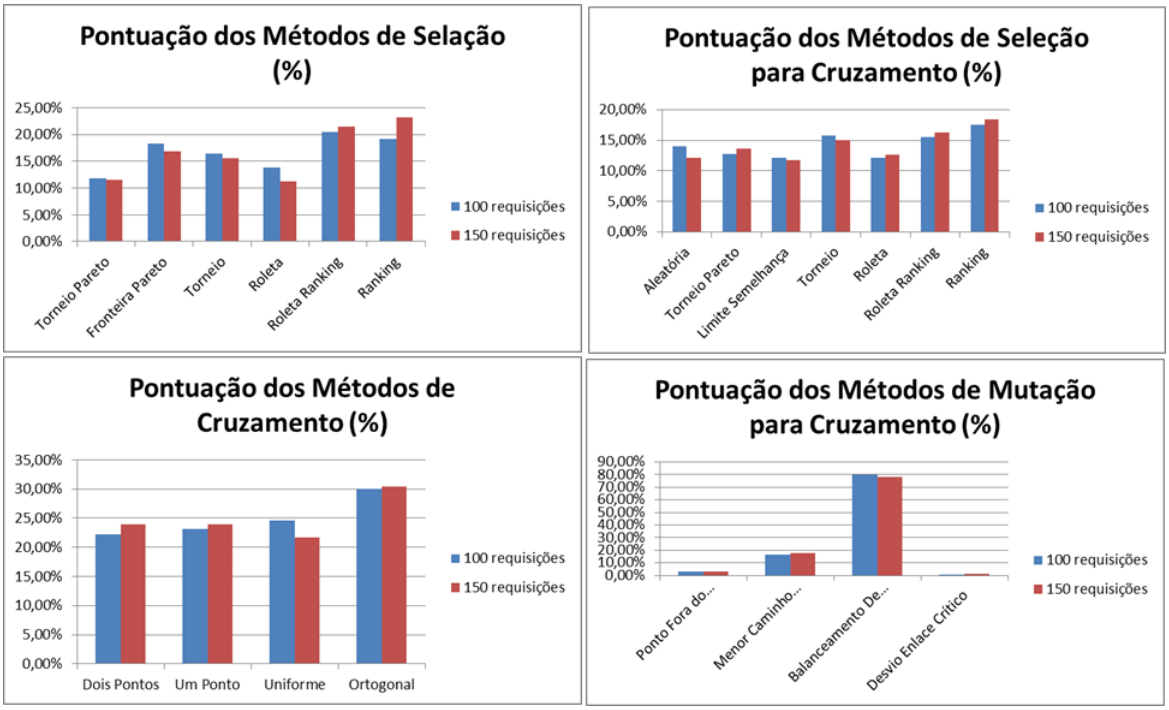

Figura 3.29: Resultados para Topologia Ring

Nos testes realizado com a topologia Sul e 150 requisições, as rejeições variaram de 26 a 40. A melhor solução ocupou 69% dos recursos da rede. Essa foi construída utilizando seleção por ranking, seleção para cruzamento por torneio, cruzamento de um ponto e mutação por balanceamento de carga.

Considerando 100 requisições, o melhor método de seleção foi o roleta-ranking, seguido do ranking. Na seleção para cruzamento, ranking é superior. Para os métodos de cruzamento, dois pontos é superior, seguido dos cruzamento uniforme e ortogonal, empatados. Para as mutações, balanceamento de carga foi superior.

Considerando 150 requisições, o melhor método de seleção foi o ranking. Na seleção para cruzamento, ranking é superior, seguido de roleta-ranking e torneio. Para os métodos de cruzamento, o cruzamento ortogonal é superior. Para as mutações, balanceamento de carga foi superior. A Figura 3.30 exibe os resultados para a topologia Sul.

Considerando o desempenho geral dos métodos de seleção para todos os pares topologia e número de requisições é possível perceber que as seleções por ranking e roleta-ranking se destacaram perante as demais. O método de seleção Fronteira de Pareto parece ser mais estável, ocupando a terceira colocação quase sempre. A seleção por torneio ocupou o quarto

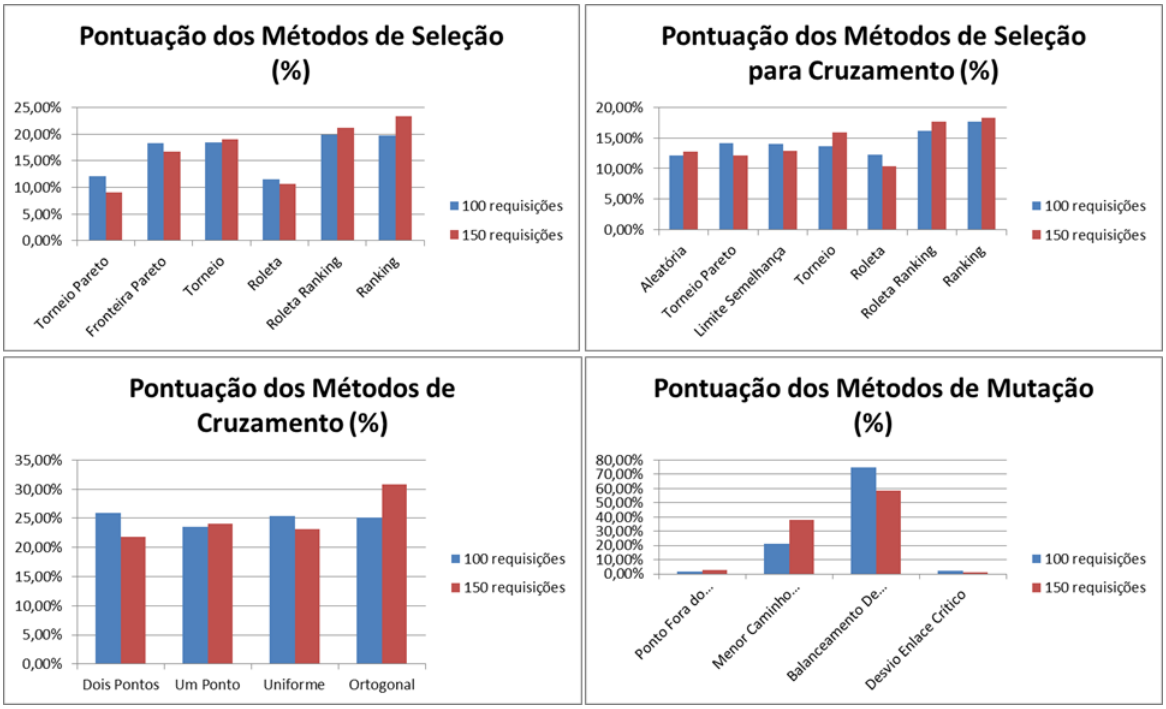

Figura 3.30: Resultados para Topologia Sul

lugar na maioria dos testes. Os métodos de seleção por roleta e torneio de Pareto dividiram a última colocação.

Nota-se que a tendência não se aplica a topologia Carrier para 100 requisições. Isso pode ser explicado pelas condições de parada do algoritmo: número máximo de gerações ou solução com zero rejeições. Devido o grande número de enlaces da topologia Carrier, em alguns casos as rotas iniciais já atendiam o critério de parada de zero rejeições, acarretando a não utilização do método de otimização. A Figura 3.31 exibe o desempenho dos métodos de seleção para todos os pares topologia e número de requisições.

Para os métodos de seleção para cruzamento, ranking foi superior na maioria dos casos testados. A segunda posição foi dividida entre as seleções por roleta-ranking e torneio. Os métodos roleta e torneio de Pareto dividem a quarta posição. A seleção aleatória se mostrou muito instável, obtendo bons resultados em alguns testes e sendo ruim em outros. A seleção por semelhança não obteve um bom resultado, ficando em último quase sempre. A mesma explicação dada para o comportamento da seleção para a topologia Carrier com 100 requisições também pode ser adotado para as seleções para cruzamento. A Figura 3.32 exibe o desempenho dos métodos de seleção para cruzamento para todos os pares topologia e número de requisições.

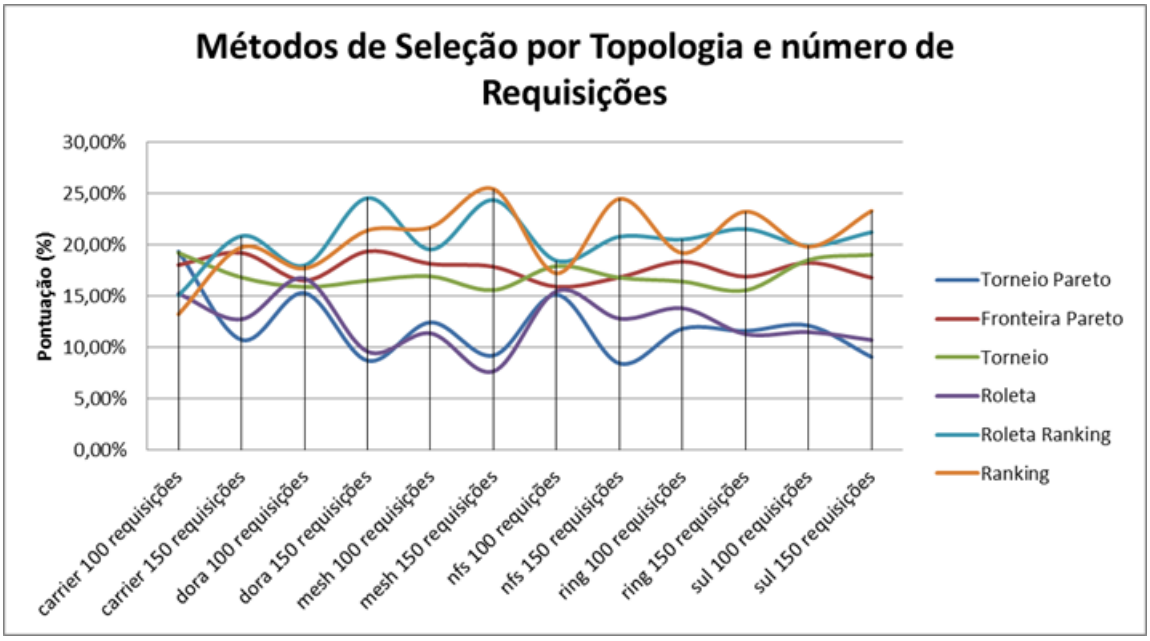

Figura 3.31: Comportamento dos métodos de seleção

Nos cruzamentos, o ortogonal foi superior na maioria dos casos testados, sendo as vezes consideravelmente melhor que os demais métodos de cruzamento. O segundo lugar é discutível devido a grande variação, porém, o cruzamento de um ponto raramente foi o último colocado. O cruzamento de dois pontos foi um pouco superior ao cruzamento uniforme. A Figura 3.33 exibe o desempenho dos métodos de seleção para cruzamento para todos os pares topologia e número de requisições.

Para os métodos de mutação, balanceamento de carga foi muito superior em relação a todos os métodos, menos para a topologia Carrier. O método de mutação por menor caminho viável ocupou a segunda colocação. O método de mutação por rota passando em um ponto aleatório ficou em terceiro lugar, sendo algumas vezes ultrapassado pelo método de mutação desvio de enlace crítico. A Figura 3.34 exibe o desempenho dos métodos de mutação todos os pares topologia e número de requisições.

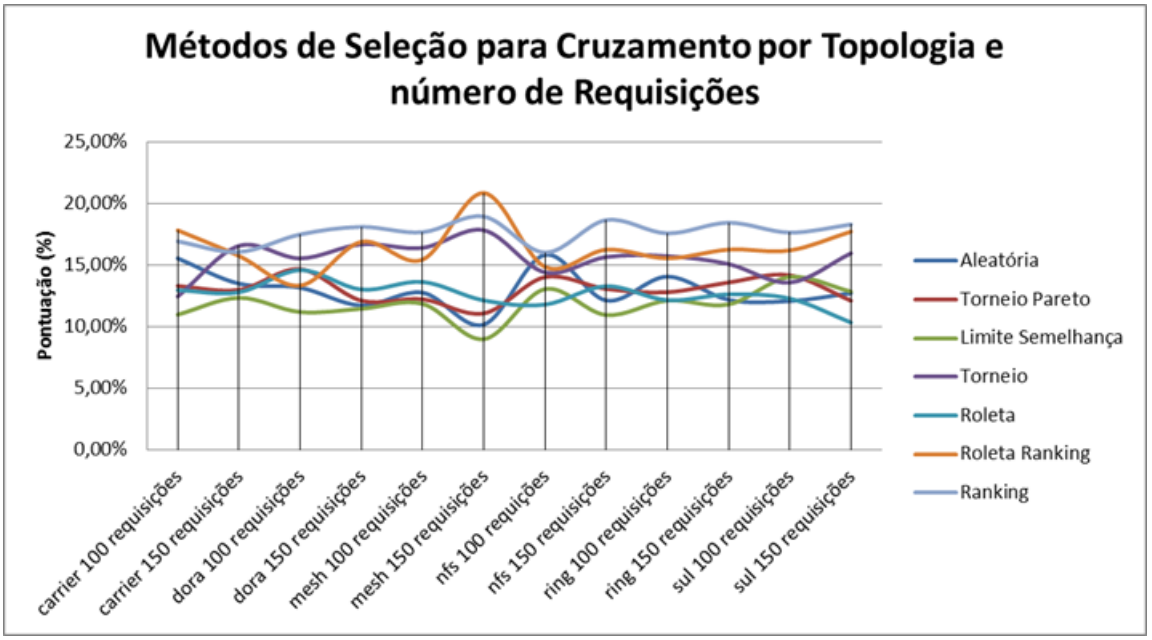

Figura 3.32: Comportamento dos métodos de seleção para cruzamento

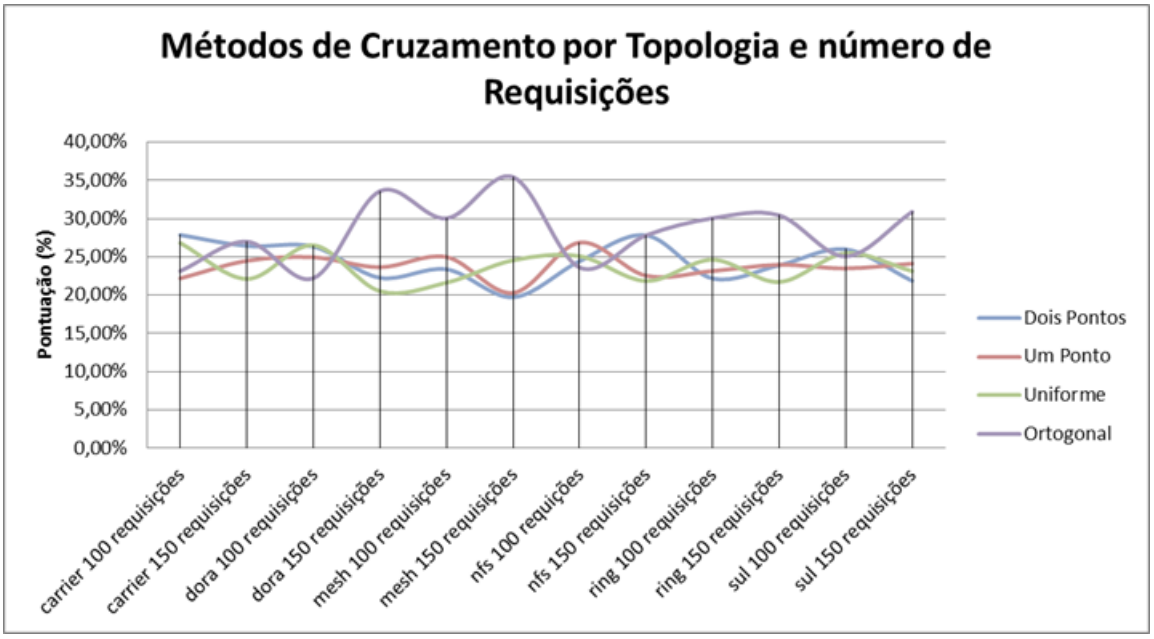

Figura 3.33: Comportamento dos métodos de cruzamento

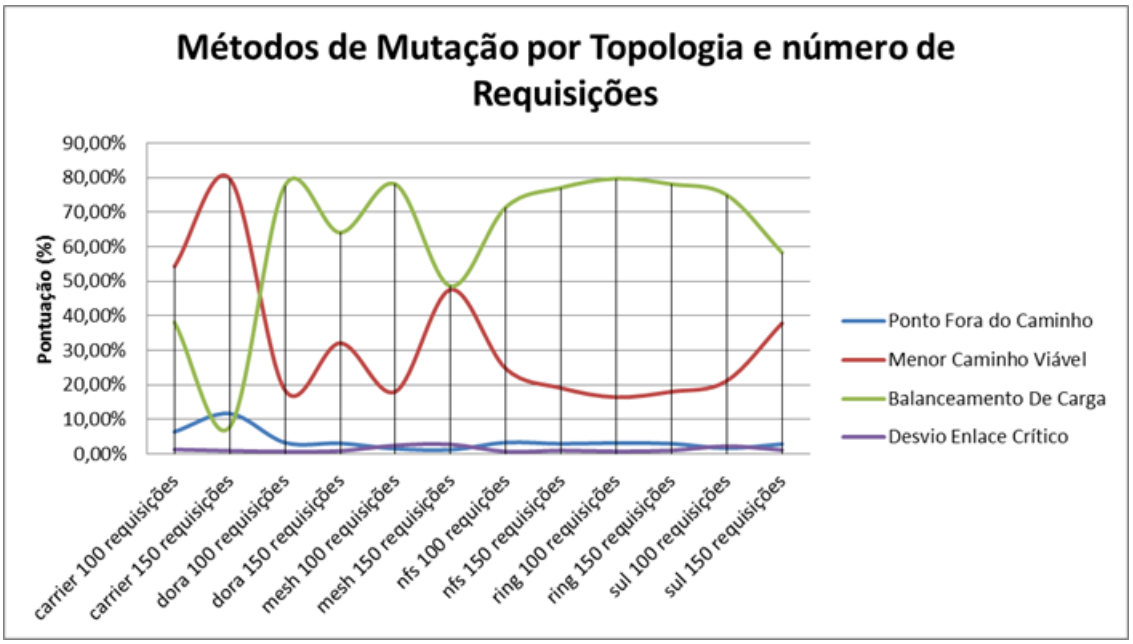

Figura 3.34: Comportamento dos métodos de mutação

Capítulo 4

Conclusão

De acordo com os testes realizados, a combinação de operadores genéticos mais adequada seria: seleção utilizando o método roleta-ranking, seleção para cruzamento por ranking, cruzamento ortogonal e mutação utilizando o método de balanceamento de carga. Porém, notou-se variações no desempenho dos operadores genéticos decorrente da execução dos testes em topologias e número de requisições diferentes. Esta constatação sugere que seria possível escolher um conjunto de métodos levando em consideração características da rede a fim de melhorar os resultados. Outro ponto a ser considerado é o sistema de avaliação da qualidade da solução. O sistema de avaliação utilizando a distribuição de pesos para as funções objetivo se mostrou eficiente, mas neste trabalho ele não é dinâmico. A adequação dos pesos seria um mecanismo útil para prevenir o aparecimento de congestionamentos em alguns enlaces ou otimizar o fluxo por caminhos mais curtos em situações de ociosidade da rede.

A construção de tabelas em formato HTML para visualização no browser e edição em softwares de planilha, como o Excel, foi de grande utilidade, facilitando o tratamento de dados. O número de relatórios poderia ser expandido a fim de evitar o tratamento manual, diminuindo o tempo gasto na análise dos resultados.

A construção de mecanismos para escolha dos operadores genéticos e adequação dinâmica dos pesos das função objetivos, além da utilização de threads para realização de tarefas simultâneas, principalmente para reavaliação de matrizes, são algumas das melhorias para trabalhos futuros.

Referências Bibliográficas

Andrade, A. V. (2008). Provisionamento de Qualidade de ServiÃ§o em Redes MPLS utilizando Algoritmos Bio-inspirados em um Ambiente de Tráfego Auto-Similar. PhD thesis, UFMG. [citado na(s) páginas(s) 11, 16, 20, 22, 25, 28, 32, 33, 34]

Ash, G. (2001). Traffic engineering & qos methods for ip-, atm-, & tdm-based multiservice network. Internet Draft. [citado na(s) páginas(s) 26]

Awduche, D., Chiu, A., Edwalid, Widjaja, A., and Xiao, X. (2002). Overview and principles of internet traffic engineering. Internet Enginnering Task Force. rfc 2702. [citado na(s) páginas(s) 27]

Blake, S. (1998). An architecture for differentiated services. Technical report, Network Working Group. [citado na(s) páginas(s) 1, 18]

Braden, Zhang, and Berson (1997). Resource reservation protocol. Technical report, Network Working Group. [citado na(s) páginas(s) 17, 26]

Braden, R. (1994). Integrated services in the internet architecture: an overview. Technical report, Network Working Group. [citado na(s) páginas(s) 16, 17]

Comer, D. E. (2007). Redes de Computadores e Internet. Pearson Education Inc. [citado na(s) páginas(s) 6]

Darwin, C. R. (1859). On the origin of species by means of natural selection. [citado na(s) páginas(s) 2]

de Assis, A. U., Ferraz, T. L., Alburqueque, M. P., Alburquerque, M. P., and Jr., N. A. (2002). Protocolo mpls. Technical report, CBPF. [citado na(s) páginas(s) 23]

de Castro, R. E. (2001). Otimização de Estruturas com Multi-Objetivos via Algoritmos Genéticos. PhD thesis, Universidade Federal do Rio de Janeiro. [citado na(s) páginas(s) 29]

Dijkstra, E. W. (1959). A Note on Two Problems in Connexion with Graphs. Numerische Mathematik 1. [citado na(s) páginas(s) 12, 13, 44]

Fielding, R., Irvine, U., and Gettys, J. (1999). Hypertext transfer protocol – http/1.1. Technical report, Network Working Group. RFC 2616. [citado na(s) páginas(s) 10]

Floyd, R. W. (1962). Algorithm 97 shortest path. Communications of the ACM, 5:345. [citado na(s) páginas(s) 12]

Forouzan, B. A. (2006). Comunicação de Dados e Redes de Computadores. [citado na(s) páginas(s) 4]

G.Cerf, V. and Kahn, R. E. (1974). A protocol for packet network intercomunication. IEEE. [citado na(s) páginas(s) 4]

Girish, M. K., Zhou, B., and Hu, J. Q. (2000). Formulation of the traffic engineering problems in mpls based ip networks. Proceedings of the Fifth IEEE Symposium on Computers and Communications (ISCC 00). [citado na(s) páginas(s) 20, 27]

Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. [citado na(s) páginas(s) 31]

Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. MIT Press. [citado na(s) páginas(s) 2, 31]

Kurose, J. F. and Ross, K. W. (2006). Redes de computadores e a Internet. [citado na(s) páginas(s) 7, 12]

Maia, N. A. (2006). Engenharia de Tráfego em Domínio MPLS utilizando Técnicas de Inteligência Computacional. PhD thesis. [citado na(s) páginas(s) 1, 2, 18, 19, 21, 26, 27, 30, 32]

Mockapetris, P. (1987). Domain names - concepts and facilities. Technical report, Network Working Group. RFC 1034. [citado na(s) páginas(s) 10]

Postel, J. (1980). User datagram protocol, rfc 768. Technical report, Internet Engineering Task Force. Atualizado pelo RFC 3168. [citado na(s) páginas(s) 11]

Postel, J. (1981a). Internet protocol, rfc 791. Technical report, Internet Engineering Task Force. Atualizado pelo RFC 3168. [citado na(s) páginas(s) 10]

Postel, J. (1981b). Transmission control protocol. Technical report, DARPA INTERNET PROGRAM. RFC 793. [citado na(s) páginas(s) 1]

Postel, J. and Reynolds, J. (1985). File transfer protocol (ftp). Technical report, Network Working Group. RFC 765. [citado na(s) páginas(s) 10]

Rosen, E., Viswanathan, A., and Callon, R. (2001). Multiprotocol label switching architecture. Technical report, RFC Editor. RFC 3031. [citado na(s) páginas(s) 1]

Saaty, T. L. (1980). The Analytic Hierarchy Process. McGraw-Hill. [citado na(s) páginas(s) 30]

Sampaio, P. R. (2011). Teoria, métodos e aplicações de otimização multiobjetivo. Master's thesis, USP. [citado na(s) páginas(s) 29]

Santos, F. A. (2009). Otimização multi-objetivo aplicada à alcação dinâmica de rotas em redes de telecomunicações. Universidade Federal de Minas Gerais, Ciências da Computação, 1:84. [citado na(s) páginas(s) 29]

Shao, H., Chen, X., and Wang, W. (2006). A multiobjectie optimization algorithm for lsp setup in diffserv and mpls networks. First International Conference on Communication and Networking. [citado na(s) páginas(s) 2, 31]

Shenker, S. and Wroclawski, J. (1997). General characterization parameters for integrated service network elements. Technical report. [citado na(s) páginas(s) 1]

Tanenbaum (2000). Computer Networks. Vrije Universiteit, quarta edition. [citado na(s) páginas(s) 5, 6, 8, 9, 11, 13, 14]

Trillium (2005). Multiprotocol label switching (mpls). Technical report, IEC. Technical Report 1. [citado na(s) páginas(s) 20, 22]

Wroclawski, J. (1997). Specification of the controlled-load network element service. Technical report, Network Working Group. [citado na(s) páginas(s) 26]

Xiao, X. and Ni, L. (1999). Internet qos: a big picture. IEEE Networks Magazine, 13:8–18. [citado na(s) páginas(s) 2]