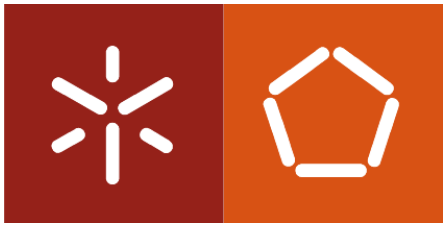


Universidade do Minho
Escola de Engenharia

Vítor Manuel Sá Pereira

Engenharia de Tráfego Robusta Usando Computação Evolucionária para Otimização de Protocolos de Encaminhamento

Braga, 2012

Vítor Manuel Sá Pereira

Engenharia de Tráfego Robusta Usando Computação Evolucionária para Otimização de Protocolos de Encaminhamento

Dissertação apresentada à Escola de Engenharia da Universidade do Minho, para a obtenção de grau de Mestre em Redes e Serviços de Comunicações.

Orientadores: Prof. Doutor Pedro Nuno Miranda Sousa
Prof. Doutor Miguel Francisco Rocha

Braga
2012

Agradecimentos

Quero agradecer, em primeiro lugar, aos meus orientadores, ao Professor Pedro Nuno Sousa pela sua disponibilidade, interesse e orientação académica, e ao Professor Miguel Rocha pelo acompanhamento, apoio e sugestões dadas no contexto da Computação Evolucionária.

Um agradecimento a todos os colegas da Universidade do Minho, pelo espirito de companheirismo, em especial ao Arthur e ao Miguel, pela amizade e partilha.

Por último, e não menos importante, um grande agradecimento à minha família e amigos por todo o apoio que me deram ao longo deste percurso, em especial nos momentos em que se esmoreciam os ânimos.

Resumo

Nos últimos anos, têm surgido várias soluções que visam alcançar configurações eficientes para o encaminhamento de tráfego OSPF (Open Shortest Path First). Este trabalho tem por intento a prossecução da investigação nessa área, com o objetivo de alcançar mecanismos de otimização eficientes que sejam resilientes a mudanças relevantes no ambiente de uma rede. Essas alterações podem resultar de vários fatores, tais como, mudanças nas necessidade de tráfego, novas restrições de Qualidade de Serviço que têm de ser consideradas, alteração de compromissos entre objetivos de otimização, mudanças na topologia de rede, que resultem de atualizações (ou falhas) na infra-estrutura de rede, entre muitas outras possibilidades. Neste contexto, este trabalho irá consistir no desenvolvimento de técnicas de otimização, inspiradas no campo da Computação Evolucionária, que procuram resolver tais problemas. Em particular, serão testados vários algoritmos e configurações com o intuito de obter métodos que sejam capazes de fornecer configurações otimizadas de pesos OSPF robustas a alterações nos requisitos de tráfego e à falha de *links*. Para além de métodos que recorrem a multi-topologias, serão testados métodos que procuram otimizar configurações de pesos OSPF para duas matrizes de requisitos de tráfego, bem como métodos que procuram otimizar as configurações de pesos para a falha do *link* com maior carga. Para obter rapidamente novas configurações de pesos ideais (ou quase), também serão abordadas questões como a inicialização da população inicial em EAs (Algoritmos Evolucionários). Estes métodos e opções de configuração serão integrados numa aplicação amigável de apoio a administradores de redes.

Palavras-chave: Engenharia de Tráfego, OSPF, Computação Evolucionária

Abstract

Over the past years, several solutions have emerged with the purpose of achieving efficient OSPF (Open Shortest Path First) routing configurations. This work intends to pursue such research aiming to devise efficient optimization mechanisms resilient to relevant changes in the network environment. Those changes may result from several factors such as, changes in the considered traffic demands, new Quality of Service constraints that should be considered, new tradeoffs between the optimization goals, network topology changes resulting from updates (or failures) in the network infrastructure, among many other possibilities. Under such circumstances, this work will devise optimization techniques inspired in the field of Evolutionary Computation to address such problems. In particular, several algorithms and configurations will be tested to achieve methods that are able to provide optimized OSPF weights configurations robust to changes in traffic demands and to link failure. In addition to methods that use multi-topology, methods that attempt to optimize OSPF weights settings for two traffic demands will be tested, as well as methods that attempt to optimize weights settings for the loadest link failure. To rapidly accomplish new (near) optimal weights, issues such as the seeding of the initial population in EAs (Evolutionary Algorithms) will be addressed. These methods and configuration options will be integrated into a user friendly application to support network administrators.

Keywords: Traffic Engineering, OSPF, Evolutionary Computation

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

ABR	Area Border Router
AF	Assured Forwarding
APC	Average Path Change
AS	Autonomous System
ASBR	Autonomous System Boundary Router
BDR	Backup Designated Router
BGP	Border Gateway Protocol
CCCT	Centro de Ciências e Tecnologias de Computação
CoS	Class of Service
CRC	Cyclic Redundancy Check
DR	Designated Router
DSCP	DiffServ Code Point
DV	Distance Vector
EA	Evolutionary Algorithm
eBGP	External Border Gateway Protocol
EC	Evolutionary Computation
ECMP	Equal-Cost Multi-Path
EF	Expedited Forwarding
EGP	Exterior Gateway Protocol
EIGRP	Enhanced Interior Gateway Protocol
FIB	Forward Information Base
GUI	Graphic User Interface
HRW	Highest Random Weight

iBGP	Internal Border Gateway Protocol
IGP	Interior Gateway Protocol
IP	Internet Protocol
ISP	Internet Service Provider
IS-IS	Intermediate System-to-Intermediate System
LAN	Local Area Network
LS	Link-state
LSA	Link-state Advertisement
LSDB	Link-state Data Base
MED	Multi-Exit -Discriminator
MOEA	Multi-Objective Evolutionary Algoritm
MOP	Multi-Objective Problem
MPLS	Multi Protocol Label Switching
MT	Multi-Topologia
MVC	Model-View-Controller
NSGA	Non-dominated Sorting Genetic Algorithm
OSPF	Open Shortest Path First
QoS	Quality of Service
RIP	Routing Information Protocol
SLA	Service Level Agreement
SP	Shortest Path
SPEA	Strength Pareto Evolutionary Algorithm
SPF	Shortest Path First
TE	Traffic Engineering
TOS	Type of Service
VEGA	Vector Evaluated Genetic Algorithm
WAN	Wide Area Network

Capítulo 1

Introdução

1.1 Enquadramento e Motivação

O tráfego Internet, com a progressão do número de utilizadores e o aparecimento de aplicações cada vez mais exigentes, aumentou oito vezes nos últimos 5 anos, e triplicará nos próximos 5 anos [2]. Este crescimento impõe a necessidade urgente de otimizar os processos de distribuição de tráfego nas redes IP (*Internet Protocol*), preservando níveis aceitáveis de Qualidade de Serviço (QoS). Neste contexto, os protocolos de encaminhamento têm um papel importante no desempenho de uma infraestrutura de rede. A Engenharia de Tráfego (TE) dedica-se a esta questão, preocupando-se com o mapeamento efetivo, na topologia física da rede, da quantidade prevista de tráfego, para alcançar os objetivos de desempenho desejados e evitar situações de congestionamento.

São várias as causas que podem conduzir ao congestionamento da rede. Um inadequado provisionamento de recursos ou uma distribuição de tráfego desequilibrada dentro da rede, são alguns exemplos. Uma distribuição de tráfego desequilibrada surge quando o tráfego que atravessa uma determinada na rede não é adequadamente mapeado para os recursos disponíveis, fazendo com que algumas zonas da rede estejam congestionadas enquanto outras estão subutilizadas. Além disso, no contexto de domínios com QoS, os *Internet Service Providers* (ISPs) têm contratos de *Service Level Agreements* (SLAs) com os seus clientes, bem como com *peered* ISPs, que têm de ser rigorosamente cumpridos para evitar penalizações financei-

ras. Com o intuito de se obter uma infraestrutura com aptidões de Qualidade de Serviço, diferentes soluções de QoS e mecanismos de controle de tráfego têm sido propostos, como por exemplo, a priorização de tráfego e a implementação de soluções seletivas de reserva de recursos. No entanto, independentemente dos mecanismos de QoS instalados na infraestrutura de comunicação, existem outros fatores que têm um papel crucial no desempenho da rede, tal como a configuração dos protocolos de encaminhamento de tráfego.

Os *Interior Gateway Protocol* (IGPs) mais utilizados nos intra-domínios atuais são o *Open Shortest Path First* (OSPF) e o *Intermediate System-to-Intermediate System* (IS-IS). Cada *router* no domínio recorre ao algoritmo de Dijkstra [9] para calcular os caminhos mais curtos de si próprio para todos os outros *routers* no domínio. O tráfego da rede é, posteriormente, encaminhado através desses caminhos mais curtos. O trabalho desenvolvido por Bernard Fortz e Mikkel Thorup, do qual alguns resultados são apresentados em [16], indicam que, num contexto em que são conhecidas estimativas dos volumes médios de tráfego que atravessam um determinado domínio, o ajuste inteligente dos pesos do algoritmo OSPF é uma ferramenta poderosa para aumentar a capacidade de uma rede. Assim, o OSPF, com ajustes inteligentes de pesos, pode prover uma grande parte dos potenciais ganhos de engenharia de tráfego, mesmo quando comparado com os esquemas *Multi Protocol Label Switching* (MPLS), muito mais flexíveis [33]. Neste contexto, o OSPF, com uma configuração de peso quase ótima, apresenta-se como uma boa solução desprovida da complexidade do MPLS.

1.2 Objetivos

A necessidade de melhorar as configurações o protocolo OSPF com preocupações de QoS conduziu ao aparecimento de vários trabalhos e propostas. Algumas dessas propostas, como as apresentadas em [12, 32], recorrem a métodos do campo da Computação Evolucionária (EC), para produzir configurações quase ótimas de pesos OSPF, tomando em consideração as necessidade de tráfego, a topologia da rede e outras características do domínio. Por causa do tempo necessário para produzir soluções e dos elevados requisitos computacionais inerentes, a reação a

alterações nas condições de funcionamento de uma rede poderá ser lenta. Essas alterações no contexto de operação de uma rede podem resultar de vários fatores, tais como mudanças nas necessidades de tráfego, novas restrições de QoS, falha de *links*, ou outras mudanças na topologia de rede. Além disso, as variações de pesos, por muito reduzidas que sejam, podem provocar efeitos indesejados no encaminhamento do tráfego.
O trabalho desenvolvido consistirá em dar continuidade aos esforços apresentados em [36, 32, 31], de modo a fornecer mecanismos capazes de responder às mudanças relevantes no contexto de um domínio OSPF, com particular atenção à alteração de necessidades de tráfego e a falha de *links*. A este objetivo acresce a necessidade de introduzir uma maior celeridade ao processo de otimização, tornando-o mais reativo. Serão feitas considerações que visam reduzir a necessidade e frequência das mudanças de pesos de *links*, dado que estas podem conduzir a consequências indesejadas no desempenho da rede. Os problemas serão abordados desenvolvendo modelos de otimização inspirados no campo da Computação Evolucionária. Em particular, serão testados vários algoritmos e configurações, como o recurso a multi-topologias, para obter métodos que sejam capazes de produzir novos pesos OSPF otimizados para as novas condições, e capazes de dotar a rede de uma maior resiliência a alterações de necessidades de tráfego e a situações de falha do *link* com maior carga.

1.3 Sumário das Principais Contribuições

O resultado final deste trabalho irá integrar uma *framework* capaz de fornecer, aos administradores de redes, diversas configurações de encaminhamento, que respondam a um conjunto variado de alterações nas condições de operação de um domínio OSPF. As novas soluções de otimização de pesos OSPF incluem:

- Otimização de pesos OSPF que considerem soluções anteriores e pesos InvCap OSPF como ponto de partida;

- Otimização de pesos OSPF para duas matrizes de tráfego, como por exemplo uma matriz de tráfego diurno e uma matriz de tráfego noturno;

- Redução dos níveis de congestão de uma rede através da otimização de pesos para OSPF Multi-Topologia;
- Otimização de pesos OSPF para a situação de falha do link com maior carga de tráfego.

Estas novas soluções serão integradas numa *framework* que, através de uma interface simples e intuitiva, disponibilizará opções de configuração e ferramentas de apoio à decisão para administradores de redes.

1.4 Estrutura da Dissertação

Este documento está organizado em 5 capítulos, dos quais se apresenta uma breve descrição:

- **Introdução:** neste capítulo é feito o enquadramento e contextualização do trabalho expondo o âmbito da sua origem, a introdução do tema geral da dissertação e os principais objetivos do trabalho a desenvolver.
- **Estado da Arte:** neste capítulo são abordados conceitos fundamentais dos protocolos de encaminhamento Link-State com particular atenção ao protocolo OSPF e à sua otimização. Serão apresentados também fundamentos dos Algoritmos Evolucionário bem como algumas soluções existentes para otimização de pesos OSPF.
- **Métodos e Resultados:** neste capítulo serão analisados os efeitos, no tempo de convergência de EAs, da inclusão de informações já existentes em populações iniciais, tal como soluções de otimizações anteriores e configurações de pesos normalmente utilizadas. Serão também analisados os efeitos da alteração das necessidades de tráfego em topologias com configurações de pesos OSPF otimizadas. Serão introduzidas novas propostas para métodos de configuração de pesos em domínios OSPF que sejam robustas a alterações de necessidades de tráfego e à falha de links. Incluem-se nestas propostas a otimização das configurações de pesos OSPF Multi-topologia, a otimização da configuração de pesos OSPF para duas matrizes de tráfego

e a otimização da configuração de pesos OSPF para a falha do link com maior carga.

- **Melhorias à *Framework* NetOpt:** neste capítulo serão identificadas as contribuições efetuadas na *framework* NetOpt, uma ferramenta de apoio a administradores de rede para a otimização de pesos OSPF com diversas opções de configuração.
- **Conclusões:** neste último capítulo são apresentadas as principais conclusões. É feita uma análise de todos os capítulos e das principais contribuições. Por fim, é feita uma descrição de futuros desenvolvimentos.

O trabalho completa-se com um anexo que inclui a definição da principal topologia de rede utilizada nos diferentes testes e simulações realizadas.

Capítulo 2

Estado da Arte

Neste capítulo são abordados conceitos fundamentais dos protocolos de encaminhamento Link-State, com particular atenção ao protocolo OSPF e à sua otimização. Serão também apresentados os fundamentos dos Algoritmos Evolucionários, utilizados na resolução de problemas de otimização. Por fim, será apresentada uma framework que recorre aos algoritmos evolucionários para otimizar a configuração de pesos OSPF.

2.1 Engenharia de Tráfego e Protocolos de Encaminhamento

A Engenharia de Tráfego (TE) lida com questões de avaliação e otimização de desempenho de redes IP, através da aplicação de tecnologias e princípios científicos para a medição, caraterização, modelação e controle de tráfego [3]. Os principais objetivos da engenharia de tráfego podem ser organizados em dois grupos:

- Orientados aos recursos - Intenta-se uma otimização da utilização dos recursos da rede de modo a evitar o congestionamento e sobrecarga, ou a pouca utilização, de certas partes da rede. Neste contexto, uma das principais funções da engenharia de tráfego é gerir de forma eficiente a utilização dos recursos disponíveis, como a capacidade dos *links*, em função dos níveis de tráfego que são expectáveis de ocorrer no domínio.

- Orientados ao tráfego - Concentra-se em questões de QoS associadas aos fluxos individuais ou agregados de tráfego que circulam na rede. Procura-se melhorar as medidas de desempenho, tais como a variação de atraso (*jitter*), o atraso (*delay*), a perda de pacotes e o débito efetivo. Neste contexto poderão ser utilizados mecanismos envolvendo processos como a classificação e priorização de pacotes, reserva de recursos, controlo de admissão, entre outros.

Esta dissertação enquadra-se maioritariamente no primeiro grupo de objetivos. O trabalho desenvolvido procurará melhorar o desempenho de uma rede operacional através de uma gestão económica e confiável dos recursos de rede, tendo presente os requisitos de tráfego existentes. Os protocolos responsáveis pelo encaminhamento do tráfego desempenham, neste contexto, um papel fundamental, e é sobre esses que recai a nossa particular atenção.

2.1.1 Protocolos de Encaminhamento

A Internet é um conglomerado de sistemas autónomos (AS) que define a autoridade administrativa e as políticas de encaminhamento de diferentes organizações. Os sistemas autónomos são compostos por dispositivos, denominados *routers*, que direcionam o tráfego entre os *hosts*. Os *routers* executam *Interior Gateway Protocols* (IGPs), como o *Routing Information Protocol* (RIP) [20], o *Enhanced Interior Gateway Routing Protocol* (EIGRP) [10], o *Open Shortest Path First* (OSPF) [24], e o *Intermediate System-to-Intermediate System* (IS-IS) [26], dentro dos limites de um AS. A interconexão entre diferentes sistemas autónomos é conseguida através de um *Exterior Gateway Protocol* (EGP). O EGP mais usado da Internet é, atualmente, o *Border Gateway Protocol* versão 4 (BGP-4) [29]. Os *routers*, com recurso aos protocolos de encaminhamento, constroem tabelas que contêm informações sobre os melhores caminhos para todos os destinos que conhecem, designadas por tabelas de encaminhamento. Os protocolos de encaminhamento baseiam-se em dois tipos de algoritmos com estratégias diferentes de conhecimento da rede: Distance Vector (DV), com um conhecimento descentralizado, e Link-state (LS), com um conhecimento global.

2.1.2 Protocolos Link-state

Os protocolos Link-state baseiam-se na existência de um mesmo mapa global da rede, distribuído em todos os *routers* que executam o protocolo. Este mapa é construído dinamicamente, não sendo imposto por qualquer fonte externa. O mapa de rede, e todas as informações sobre os *routers* e *links* (bem como rotas), são mantidos numa base de dados de estados em cada *router*. A base de dados não é um "mapa", no sentido usual da palavra, mas um conjunto de registos que representam a topologia da rede como uma série de *links* entre *routers*.
Os *routers* anunciam um conjunto de informações, através de *Link-state Advertisements* (LSA), que incluem as redes a que estão ligados, ou informações de acessibilidade redistribuída por outro protocolo de encaminhamento. Quando um *router* inicializa, ele obtém uma imagem completa da base de dados dos seus vizinhos, e constrói uma tabela de encaminhamento calculando os melhores caminhos para cada prefixo de destino. O protocolo Link-state passa então a receber somente LSA *updates* que refletem algum tipo alteração. O cálculo dos caminhos é efetuado utilizando o algoritmo do caminho mais curto, *shortest path first* (SPF), também conhecido como o algoritmo de Dijkstra. O SPF pode ser executado, quando necessário, recalculando todos os caminhos para para os vários destinos (SPF completo), ou efetuando um cálculo parcial de caminhos, por exemplo no caso em que uma única rota externa é alterada.
Os algoritmos Link-state adaptam-se dinamicamente, e de forma rápida, às mudanças nas condições de rede, pois as alterações são propagadas independentemente do processo de cálculo de caminhos de cada *router*. Como cada *router* conhece a totalidade da topologia da rede, o algoritmo SPF converge rapidamente. O IS-IS e o OSPF são os protocolos Link-state mais utilizados.

2.1.3 Algoritmo do Caminho Mais Curto

O algoritmo do caminho mais curto (SP) foi desenvolvido por Edsger Dijkstra em 1956 [9], e é o algoritmo mais conhecido para encontrar o caminho mais curto entre dois vértices u e v de um grafo direto com pesos $G = (V,E)$.
A cada arco do grafo é associado um peso w, representado por um número real. O peso $w(p)$ de um caminho $p = \langle v_0,v_1,\ldots,v_k\rangle$ é a soma dos pesos dos arcos que

o constituem [7]:

$$w(p) = \sum_{i=1}^{k} w(v_{i-1},v_i)$$

Esta soma é minimizada, encontrando o caminho com menor peso $\delta(u,v)$:

$$\delta(u,v) = \begin{cases} min\left\{w(p) : u \overset{p}{\rightsquigarrow} v\right\}, & \text{se existe um caminho de } u \text{ para } v \\ \infty, & \text{caso contrário} \end{cases}$$

Qualquer caminho p, entre os vértices v e u, que verifica $w(p) = \delta(u,v)$ é um caminho mais curto entre esses dois vértices. O algoritmo de Disjktra é um

Algoritmo 1: Algoritmo do Caminho Mais Curto

```
1  d[s] = 0
2  p[s] = s
3  foreach v ∈ V, v ≠ s do
4      d[v] = INF
5      p[v] = NULL
6  end
7  S = NULL
8  Q = V
9  while Q ≠ ∅ do
10     u = EXTRACT_MIN(Q)
11     S = S ∪ {u}
12     foreach v adjacente a u do
13         if d[v] > d[u] + w[u,v] then
14             d[v] = d[u] + w[u,v]
15             p[v] = u
16             DECREASE_KEY[v,Q]
17         end
18     end
19 end
```

algoritmo *greedy* que, efetuando escolhas locais, resolve o problema do caminho mais curto com uma única origem para pesos não negativos, ou seja, dado um vértice de origem $s \in V$, pretende-se encontrar os caminhos mais curto de s para todos os restantes vértices $v \in V$ de destino. Um pseudo-código deste algoritmo é apresentado em Algoritmo 1.

Inicialização do algoritmo - Para cada vértice v é mantida uma variável d com a menor distância encontrada de s para v. Uma variável p é utilizada para guardar o predecessor de cada vértice v no caminho mais curto. No início do algoritmo as distâncias $d(v)$, para todos os nós $v \neq s$, são infinitas. O predecessor de s é ele próprio, e os restantes predecessores são nulos.

Relaxamento - O processo de relaxamento de um arco (u,v) consiste em testar se é possível melhorar o caminho mais curto, encontrado até ao momento, para v passando por u. Caso essa alteração no caminho represente uma melhoria, os valores de d e p são atualizados.

O algoritmo de Dijkstra mantém um conjunto S de vértices para os quais a distância mais curta, a partir da origem, foi encontrada (*settled nodes*), e uma fila Q de vértices não examinados (*unsettled nodes*), que traduz o invariante $Q = V - S$ em cada iteração. O resultado produzido pelo algoritmo é uma árvore sem ciclos de caminhos mais curtos para todos os nós, com o *router* local como raiz.

A seguir apresenta-se uma descrição do Algoritmo 1, contemplando todos os passos.

- **1-8:** Processo de inicialização. O vértice s é o nó inicial, que tem um custo zero para si próprio. A distância para os restantes nós é infinita. Todos os nós na base de dados link-state são adicionados à fila Q.

- **9:** Início do ciclo. O ciclo termina quando não existirem nós não tratados.

- **10-11:** O nó u, cuja distância à raiz é a menor, é removido de Q e é adicionado a S. Na primeira iteração, o nó u coincide com s.

- **12-18:** Processo de relaxamento.

A árvore de caminhos mais curto é construída a partir da lista de predecessores. A complexidade do algoritmo pode ser delimitada a $O(l + n \log(n))$, onde $n = |V|$ é o número de vértices e $l = |E|$ é o número de arcos.

2.1.4 Princípios de Funcionamento do OSPF

O OSPF é um protocolo Link-state, e consequentemente cada *router* precisa conhecer a totalidade da topologia de rede. Por motivos de escalabilidade, o OSPF divide o domínio de encaminhamento ou sistema autónomo (AS) em várias áreas. As áreas OSPF de um AS são dispostas em torno de uma área 0 ou a área de *backbone*, à qual se ligam as restante áreas. Todas as rotas OSPF com origem numa área e destino noutra área precisam passar através da área de *backbone*. Os *routers* com interfaces em várias áreas são conhecidos como *area border routers* (ABRs). Além disso, alguns *routers* conhecidos como *autonomous system boundary routers* (ASBRs), podem ter links para *routers* de outros sistemas autónomos. A divisão de um domínio de encaminhamento em várias áreas permite reduzir a informação topológica necessária, limitando-se às áreas nas quais possui interfaces.

Os *routers* OSPF, com interfaces em LANs de *broadcast* ou links ponto-a-ponto, descobrem os restantes *routers* na sua vizinhança imediata através da troca periódica de *hellos*. Cada *router* envia uma mensagem *multicast* de *hello* através das suas interfaces após cada intervalo de tempo (*HelloInterval*). Na mensagem de *hello*, o *router* inclui uma lista com todos os nós dos quais recebeu recentemente um *hello*. Se um *router* A descobre que se encontra listado na mensagem de *hello* do vizinho B então a sua adjacência com o vizinho é bidireccional. Se o *router* A pretender estabelecer uma adjacências completa com vizinho B, este inicia o processo de sincronização da sua base de dados Link-state (LSDB) com a LSDB do vizinho. O *router* A gera então um novo LSA listando o estado de adjacência de todas as suas interfaces que pertencem a uma mesma área (como a ligação entre ele e o vizinho B) e envia o LSA através das suas interfaces. Quando um *router* vizinho recebe esse LSA, este reenvia-o por todas as interfaces na área, à exceção daquela pela qual o LSA foi recebido. Assim, o LSA é transmitido por toda a área. Quando um *router* não recebe um aviso de recepção de LSA, proveniente de um vizinho, dentro de um determinado intervalo de tempo (*RxmtInterval*), este retransmite o mesmo LSA ao vizinho. Cada router da área recebe assim o LSA inicialmente transmitido pelo router A e toma conhecimento dos vizinhos com o qual esse router estabeleceu uma adjacência completa.

Dois *routers* permanecem adjacentes enquanto trocarem periodicamente mensagens de *hello*. A adjacência é quebrada quando um *router* não recebe qualquer *hello* do vizinho durante um período de tempo *RouterDeadInterval*. Isto acontece se o *link* entre o *router* e o vizinho falhar ou se o *router* vizinho deixar de estar funcional. Em alguns casos, o protocolo da camada de enlace pode informar um *router* sobre a falha de um *link*, e assim permitir que o *router* termine a adjacência sem esperar que o *RouterDeadInterval* expire. A quebra de uma adjacência provoca a geração um novo LSA no *router*. Este LSA é enviado para toda a área, informando assim os restantes *routers* da quebra de adjacência. Quando um *router* recebe um novo LSA, este recalcula e atualiza sua tabela de encaminhamento ou *Forward Information Base* (FIB).

Resumidamente, a convergência de uma alteração na topologia é constituída pelas etapas seguintes [24]:

- Deteção de uma mudança de topologia pelos *routers* na vizinhança.
- Estabelecimento ou quebra de adjacências pelos *routers* afetados pela mudança de topologia.
- Geração de novos LSAs e consequente *flooding* por toda a área.
- Cálculo da tabela de encaminhamento por cada *router* ao receber os novos LSAs.

O tempo de convergência global depende do tempo necessário para completar cada uma das etapas mencionadas. Para minimizar a quantidade de informação trocada, e tornar o protocolo mais escalável, o OSPF elege um *router* para tornar-se *designated router* (DR), bem como um *router* para ser *backup designated router* (BDR). Assim, em vez de cada *router* trocar informações de *update* com os restantes *routers* da área, todos os *routers* trocam informações de adjacência com o DR e BDR, sendo estes responsáveis por gerar *updates* de informações de topologia e transmitir aos restantes. Este procedimento também permite agilizar a sincronização de bases de dados Link-state. Com a centralização da base de dados, efetuar a manutenção da base de dados, como adicionar um novo *router*,

torna-se um problema linear.

No contexto deste trabalho, alguns dos aspectos operacionais do protocolo OSPF serão simplificados. Neste sentido, e tendo em consideração o objetivo de otimização das configurações de pesos OSPF, será assumido um modelo matemático considerando um conjunto de vértices e arcos que representam, respetivamente, os *routers* e *links* de uma topologia de rede. A cada arco será associado um peso para ser utilizado no cálculo dos caminhos mais curtos.

2.1.5 Optimização de Pesos OSPF

O protocolo OSPF encaminha o tráfego de rede através dos caminhos mais curtos obtidos a partir da definição de um conjunto de pesos. Os *links* que se encontram nos caminhos mais curtos, em consequência de elevadas necessidades de tráfego, ou por configuração inadequada de pesos, podem ficar congestionados. Bernard Fortz e Mikkel Thorup em [16] introduziram uma função custo que permite avaliar a congestão de uma rede tomando como parâmetros uma matriz de pedidos de tráfego e um conjunto de pesos. Nesse trabalho, fazendo uso dessa mesma função custo, é proposta uma solução para o problema de configuração dos pesos OSPF, isto é, dada uma matriz de requisitos de tráfego, encontrar um conjunto de pesos OSPF que otimiza o desempenho da rede. Apesar do problema de otimização ser *NP-hard*, como demonstrado pelos autores, esse trabalho tem conduzido a várias investigações, como a que será apresentada neste trabalho.

Função de Avaliação

Uma topologia de rede é modelada por um grafo direto $G = (N,A)$, cujos vértices N e arcos A representam *routers* e *links* respetivamente. As necessidade de tráfego são modeladas numa matriz D que identifica os requisitos de tráfego entre cada origem s e destino t, $D(s,t)$. Os requisitos de tráfego, aos quais nos referiremos a partir deste ponto como *demands*, refletem estimativas de tráfego entre nós de *ingress* e *egress* do domínio (ou *edge-to-edge*) e podem ser obtidas por várias técnicas [22].
Para cada arco $a \in A$, a capacidade é expressa por $c(a)$, e a carga total por $\ell(a)$.

A carga total é a soma dos fluxos $f_a^{(s,t)}$, que para cada par (s,t) e cada arco a reflete quanto trafego de s para t passa por a, ou seja,

$$\ell(a) = \sum_{(s,t)\in N\times N} f_a^{(s,t)} \tag{2.1}$$

Uma função custo $\Phi(\ell(a))$ traduz, para cada arco $a \in A$, quão próxima se encontra a carga $\ell(a)$ da capacidade $c(a)$. Esta função linear, crescente e convexa (Figura 2.1), penaliza os arcos com maior carga e cresce exponencialmente quando a carga $\ell(a)/c(a)$ ultrapassa os 110% da capacidade, modelando desta forma o custo de retransmissão de pacotes devido à ocorrência de perda de pacotes. Para cada arco $a \in A$, Φ_a é uma função contínua tal que $\Phi_a(0) = 0$, ou seja, não aplica qualquer penalização quando a carga for nula. As restante penalizações, em função da carga, são traduzidas pela sua função derivada 2.2.

$$\Phi_a'(x) = \begin{cases} 1 & \text{para} & 0 \leq x/c(a) < 1/3 \\ 3 & \text{para} & 1/3 \leq x/c(a) < 2/3 \\ 10 & \text{para} & 2/3 \leq x/c(a) < 9/10 \\ 70 & \text{para} & 9/10 \leq x/c(a) < 1 \\ 500 & \text{para} & 1 \leq x/c(a) < 11/10 \\ 5000 & \text{para} & 11/10 \leq x/c(a) < \infty \end{cases} \tag{2.2}$$

Definidas as penalizações para a taxa de ocupação dos arcos, o objetivo consiste em distribuir as *demands* de tráfego de modo a minimizar a soma de todos os custos, como expresso na Equação 2.3.

$$\Phi = \sum_{a\in A} \Phi_a(\ell(a)) \tag{2.3}$$

Problema de Programação Linear

O problema de otimização de pesos OSPF pode ser formulado como um problema de fluxos de multi-comodidades, ou seja, um problema de tráfego de bens ou produtos entre origens e destinos, em que a quantidade de tráfego não é limitada à capacidade do meio, ou seja, não é aplicada a restrição $\ell(a) \leq c(a)$.
O problema de programação linear para a otimização de pesos OSPF é definido

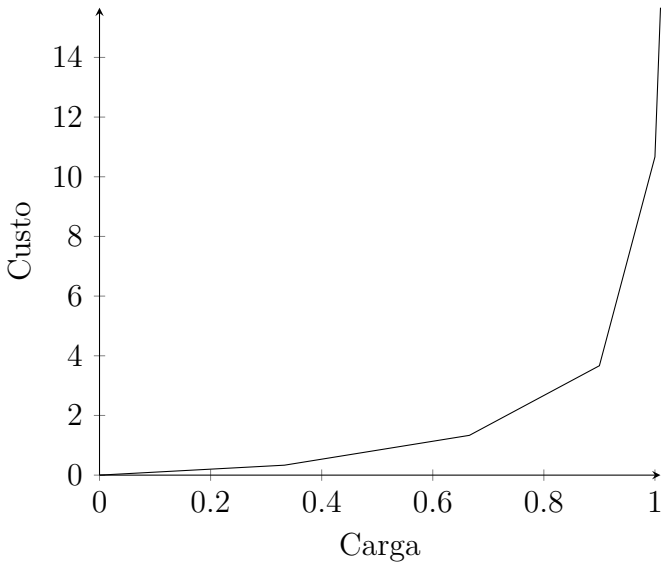


Figura 2.1: Função Φ_a para $c\,(a) = 1$

pela função objetivo 2.4 e pelas restrições que a seguir se descrevem:

Função objetivo:

$$\min\Phi = \sum_{a \in A} \Phi_a \tag{2.4}$$

Restrições:

$$(1) \quad \sum_{u:(u,v)\in A} f_{(u,v)}^{(s,t)} - \sum_{u:(v,u)\in A} f_{(v,u)}^{(s,t)} = \begin{cases} -D\,(s,t) & \text{if } v = s, \\ D\,(s,t) & \text{if } v = t, \\ 0 & \text{caso contrário} \end{cases} \qquad v,s,t \in N,$$

$$(2) \quad \ell\,(a) = \sum_{(s,t)\in N\times N} f_a^{(s,t)} \qquad a \in A,$$

$$(3) \quad \Phi_a \geq \ell\,(a) \qquad a \in A,$$

$$(4) \quad \Phi_a \geq 3\ell\,(a) - \tfrac{2}{3}c\,(a) \qquad a \in A,$$

$$(5) \quad \Phi_a \geq 10\ell\,(a) - \tfrac{16}{3}c\,(a) \qquad a \in A,$$

$$(6) \quad \Phi_a \geq 70\ell\,(a) - \tfrac{178}{3}c\,(a) \qquad a \in A,$$

$$(7) \quad \Phi_a \geq 500\ell\,(a) - \tfrac{1468}{3}c\,(a) \qquad a \in A,$$

$$(8) \quad \Phi_a \geq 5000\ell\,(a) - \tfrac{19468}{3}c\,(a) \qquad a \in A,$$

$$(9) \quad f_a^{(s,t)} \geq 0 \qquad a \in A; s,t \in N.$$

A restrição (1) garante a preservação de fluxo, ou seja, que as *demands* entre s e t são encaminhadas como desejado. A restrição (2) define a carga para cada arco. As restrições de (3) a (8) definem o custo aplicado a cada arco. A restrição

(9) garante que para todo o para origem/destino, a quantidade de tráfego que transita pelo arco a é não negativa.

Otimização do Protocolo OSPF

Num domínio OSPF, o tráfego com origem num nó s e destino t é encaminhado pelo caminho mais curto de s para t. Esse caminho mais curto foi calculado pelo algoritmo SPF, com base na configuração dos pesos atribuídos a cada link.
Os pesos OSPF são valores inteiros no intervalo de 1 a 65535 ($2^{16}-1$). No entanto, esse intervalo pode ser reduzido para um intervalo mais pequeno $[w_{min},w_{max}]$, aumentando assim a probabilidade de surgirem caminhos de custo igual. Com esta técnica de TE é possível conseguir um melhor balanceamento de tráfego, e consequentemente obter um melhor nível de congestão na rede.
O processo de divisão do tráfego entre caminhos, não sendo uma tarefa fácil, pode seguir três estratégias distintas [43]:

- *Round Robin* sobre pacotes;
- Divisão de destinos entre próximos saltos;
- Divisão do tráfego usando uma função de *hash* sobre os endereços de origem e destino de tráfego.

Por outro lado, a quantidade de tráfego pode não ser dividida de forma equitativa pelos próximos saltos. Em [45] é apresentada uma proposta para o problema de otimização de pesos OSPF na qual o tráfego com origem num nó s e destino t pode ser distribuído com rácios arbitrários pelos caminhos mais curtos entre s e t. Em [40], uma divisão de tráfego baseada em endereços de origem e destino é convertida numa divisão somente por destino, e para prefixos particulares, são somente disponibilizados alguns próximos saltos, conseguindo dessa forma rácios desiguais.
Neste trabalho os pesos OSPF assumem valores no intervalo de 1 a 20. A divisão do tráfego segue a seguinte estratégia:

- Se um link (u,v) não estiver no SP então $f_{(u,v)}^{(s,t)} = 0$.

- Se dois links (u,v) e (u,v') estiverem no SP então é efetuado um balanceamento de carga, *Equal-Cost Multi-Path* (ECMP), ou seja, $f_{(u,v)}^{(s,t)} = f_{(u,v')}^{(s,t)}$.

O custo obtido para o encaminhamento com OSPF é denotado por $\Phi_{OptOSPF}$.

Normalização da Função Custo

Para poder comparar custos entre topologias diferentes é necessário definir um fator de normalização. Esse fator, Φ_{Uncap}, é definido pela Equação 2.5,

$$\Phi_{Uncap} = \sum_{(s,t)\in N\times N} \left(D\left(s,t\right) \times dist_1\left(s,t\right)\right), \tag{2.5}$$

onde $dist_1$ é a distância entre nós medida com pesos unitários, ou seja, o número de saltos. Denotando por $\Phi_{unitOSPF}$ a função custo para pesos unitários em todos os arcos, podem ser definidas as seguintes propriedades:

(i) Φ_{Uncap} é a carga total se o tráfego fluir através de SP com peso unitário.

(ii) $\Phi_{Uncap} = \Phi_{UnitOSPF}$ se todos os arcos tiverem capacidade ilimitada.

(iii) Φ_{Uncap} é a carga total mínima da rede.

(iv) $\Phi_{Uncap} \leq \Phi_{OPT}$

(v) $\Phi_{UnitOSPF} \leq 5000.\Phi_{Uncap}$

Se denotarmos a solução ótima do problema por Φ_{OPT} e as normalizações por

$$\Phi^* = \Phi/\Phi_{Uncap}, \tag{2.6}$$

então estas propriedades permitem obter a seguinte relação de ordem:

$$1 \leq \Phi^*_{OPT} \leq \Phi^*_{OptOSPF} \leq \Phi^*_{UnitOSPF} \leq 5000 \tag{2.7}$$

Quando $\Phi^* = 1$, pode-se concluir que todos os link têm carga inferior a 1/3 da sua capacidade. Por outro lado, quando todos os arcos possuem carga igual ao limite das suas capacidades, Φ^* é igual a 10 2/3. Este último valor é então considerado como o limite da região de funcionamento aceitável da rede.

Implementações

O trabalho de Bernard Fortz e Mikkel Thorup conduziu ao desenvolvimento de distintas implementações do processo de otimização de pesos OSPF, baseadas na função ϕ, que recorrem a métodos de otimização diferentes. De seguida são apresentados alguns desse métodos.

- **Método de Procura Local**
  O método de procura local é uma meta-heurística utilizada na resolução de problemas difíceis de otimização, que recorre a tentativa e erro para encontrar uma solução. Um algoritmo de procura local parte de uma solução candidata e, em seguida, move-se de forma iterativa para uma solução vizinha. Isto só é possível se existir uma relação de vizinhança no espaço de procura. Como cada solução candidata tem tipicamente mais do que uma solução vizinha, a escolha da solução para a qual se move o algoritmo é feita tomando a melhor solução na vizinhança [15]. Os algoritmos de procura local são considerados incompletos, pois a procura pode terminar mesmo quando a solução encontrada não é a ideal.

- **Algoritmos Evolucionários**
  Os algoritmos evolucionários são técnicas de otimização que derivam dos princípios da seleção natural. Estes algoritmos foram aplicados pela primeira vez ao problema de configuração dos pesos OSPF em [12]. Estes algoritmos, que serão descritos na secção 2.2, constituem uma das principais ferramentas usadas neste trabalho para otimizar pesos OSPF.

- **Arrefecimento simulado**
  O arrefecimento simulado ou *simulated annealing* é um meta-algoritmo probabilístico genérico para o problema de otimização globais, ou seja, que localiza uma boa aproximação para o ótimo global da função num espaço de procura de grandes dimensões. Este algoritmo baseia-se no processo físico de arrefecimento de sólidos. Um sólido é inicialmente aquecido até uma temperatura elevada e, em seguida, é arrefecido lentamente, até que que o sistema entre em equilíbrio termodinâmico. Na aplicação do *simulated annealing* ao problema de otimização de pesos OSPF [39], o algoritmo inicia

com a escolha aleatória de um vetor de pesos. Alguns pesos nesse vetor são alterados para produzir uma nova solução. Após a avaliação da nova solução, é calculada a variação relativamente à solução atual, e se essa variação for negativa, a solução atual é substituída pela nova solução. O processo termina quando, de acordo com os parâmetros do algoritmo, não é encontrada uma solução melhor do que a atual.

- **Algoritmos Genéticos Híbridos**
  A utilização de um algoritmo genético híbrido foi proposto em [5], e estende um algoritmo genético efetuando uma procura na vizinhança da cada descendente. Em cada iteração do algoritmo genético, um processo de melhoria local é aplicado a cada descendente obtido por cruzamento. O procedimento de melhoria local analisa as soluções na vizinhança do vetor solução atual, procurando uma solução com menor custo de encaminhamento. Se tal solução existir, esta substitui a solução atual.

Um estudo comparativo das abordagens descritas pode ser encontrado em [17].

2.2 Algoritmos Evolucionários

Muitos problemas de otimização são de resolução difícil devido, em parte, ao número exponencial de possíveis soluções. Nestes casos, realizar uma procura exaustiva da melhor solução não é de todo viável. Como a otimização de pesos OSPF é um problema NP-difícil, meta-heurísticas da área da Computação Evolucionária (EC) podem ser utilizadas para melhorar as configurações de encaminhamento [12, 32]. Os Algoritmos Evolucionários (EA), um tipo de EC, são técnicas de otimização derivadas dos princípios da seleção natural e teoria da evolução das espécies [30], cuja a aplicação a problemas de otimização se tem demonstrado robusta.
Nas secções seguintes serão descritos princípios fundamentais dos EA e configurações essenciais na prossecução do trabalho desenvolvido.

2.2.1 Visão Global dos Algoritmos Evolucionários

Um EA transforma uma população de soluções individuais, cada uma com um valor de aptidão ou *fitness* associado, numa nova geração da população, utilizando o princípio darwiniano da sobrevivência dos mais aptos. Cada possível solução do espaço de procura do problema é codificada numa representação adequada ao problema. Distintas soluções são obtidas aplicando operadores genéticos como cruzamento e mutação. Em cada iteração, novas soluções são criada pelo processo de reprodução, sendo posteriormente alvo de seleção. Um algoritmo genético simples é descrito no Algoritmo 2. Nesse pseudo código, a população no momento t é representada por $P(t)$. Os passos de um EA, de acordo com o pseudo código são:

1. Criar uma população inicial aleatória $P(0)$ de indivíduos.
2. Realizar iterativamente, até se verificar o critério de paragem, os seguintes passos na geração corrente da população:
   (a) Atribuir um valor de aptidão a cada individuo com recurso a uma função de avaliação.
   (b) Selecionar progenitores para entradas do processo de reprodução.
   (c) Criar descendentes a partir dos progenitores selecionados utilizando operadores genéticos (cruzamentos e mutações).
   (d) Identificar os melhores indivíduos obtidos até ao momento na iteração corrente (a seleção é realizada de forma estocástica).

Em qualquer EA, dois conjuntos fundamentais de operadores são aplicadas durante o processo de procura (ver Figura 2.2), e que são essenciais à obtenção de soluções ótimas [11].

- **Operadores para variação da população** (recombinação e mutação): são responsáveis por criar diversidade e pelo surgimento de características genéticas inexistentes até à fase de evolução considerada;

Algoritmo 2: Pseudocódigo de um algoritmo genético simples

```
1  t = 0
2  INICIALIZAR P(0)
3  AVALIAR P(0)
4  while !CRITÉRIO DE PARAGEM do
5      t = t + 1
6      SELECIONAR progenitores de P(t)
7      RECOMBINAR pares de progenitores
8      APLICAR MUTAÇÃO nos descendentes obtidos
9      AVALIAR os novos candidatos
10     SELECIONAR indivíduos para a próxima geração
11 end
```

- **Seleção**: atua como elemento responsável por garantir a qualidade das soluções, levando a escolha das soluções para alternativas que possam resolver o problema considerado, ou seja, é o elemento responsável pela convergência.

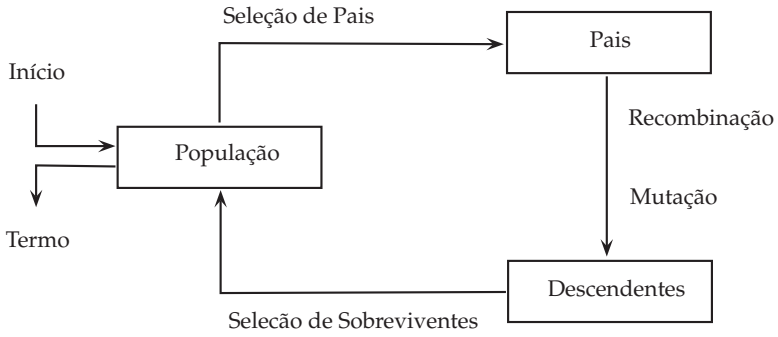


Figura 2.2: Esquema geral de um algoritmo evolucionário

Juntos, estes dois componentes são responsáveis pelo aumento da qualidade das soluções geração após geração. Diversos outros componentes, não menos importantes, devem ser considerados para se obter um modelo computacional adequado para solução de problemas específicos.

Outro parâmetro importante dos EAs é a dimensão da população, que identifica o número de indivíduos existentes em cada geração. Caso a população seja reduzida, existirão poucos indivíduos para realizar recombinações e somente uma pequena região do espaço de procura é explorada. Se, por outro lado, existirem demasiados indivíduos, o tempo necessário para cada iteração do EA aumenta. Os resultados mostram que, a partir de um determinado tamanho da população, não se regista qualquer ganho na resolução do problema. O tamanho da população depende também do tipo de representação utilizada.

Representação

Em 1886, Gregor Mendel [23] percebeu que a natureza faz a distinção entre o código genético de um individuo e sua aparência exterior. O genótipo representa todas as informações armazenadas no genoma e permite descrever um indivíduo ao nível dos genes. O fenótipo descreve a aparência externa de um indivíduo. Existe um mapeamento, ou representação, genótipo-fenótipo, que utiliza as informações genotípicas para construir o fenótipo. Para representar o grande número de fenótipos possíveis com apenas quatro nucleótidos, a informação genotípica não é armazenado nos próprios alelos, mas na sequência de alelos. Ao interpretar a sequência de alelos, a natureza pode codificar um grande número de diferentes expressões fenotípicas utilizando apenas alguns tipos diferentes de alelos.

Quando se fala de indivíduos numa população, devemos distinguir cuidadosamente entre genótipos e fenótipos. A aparência fenotípica de um indivíduo determina o seu sucesso na vida. Assim, quando se compara a capacidade de diferentes indivíduos, é necessário avaliar ao nível do fenótipo. No entanto, quando se trata de reprodução, devemos olhar para os indivíduos o nível do genótipo.

A identificação do método adequado para codificar soluções em cromossomas é uma questão fundamental no uso de EAs. Este problema tem sido investigado em muitos aspectos, como o mapeamento de caracteres do espaço genotípico para o espaço fenotípico quando os indivíduos são decodificados em soluções, bem como propriedades de metamorfose quando os indivíduos são manipulados por operadores genéticos. A codificação binária foi um dos primeiros métodos de codificação utilizado, todavia, hoje em dia, sabe-se que tem inconvenientes

graves devido à existência de *Hamming cliffs*[1], pares de codificações que têm uma distância de *Hamming* grande mas que pertencem a pontos de distância mínima em espaço fenótipo. Por exemplo, o par 011111111 e 1000000000 pertencem a pontos vizinhos no espaço fenótipo (distância euclidiana mínima) mas têm uma distância de *Hamming* máxima no espaço genótipo. Nos últimos anos, vários métodos de codificação foram criados para conseguir implementações efetivas de algoritmos evolucionários, e que se adaptem à particularidade dos problemas. De acordo com a simbologia utilizada como alelos de um gene, os métodos de codificação podem ser classificados como:

- codificação binária
- codificação real ou de vírgula flutuante
- codificação inteira
- codificação de permutação de literais
- codificação em estrutura (por exemplo em árvore)

entre muitos outros.
A aplicação de EAs à resolução de qualquer problema de otimização, inicia-se sempre com a definição do método de codificação ou representação. No contexto deste trabalho é utilizada uma representação por números inteiros, com correspondência direta aos pesos OSPF, sendo mantidos os limites inferior e superior.

Função de Aptidão

A aptidão no sentido biológico é um valor de qualidade, uma medida da eficiência reprodutiva dos indivíduos. Nos algoritmos evolucionários, a aptidão ou *fitness* é usada para alocar características reprodutivas de indivíduos à população e, assim, atuar como medida de qualidade a ser otimizada. Isto significa que os indivíduos com melhor valor de aptidão terão maior probabilidade de serem selecionados e integrarem uma nova geração. Uma função de aptidão avalia a qualidade das

[1] A distância de Hamming entre duas cadeias de comprimento igual é o número de posições em que os símbolos correspondentes são diferentes.

soluções, e é responsável, em conjunto com os métodos de seleção, pela melhoria contínua das soluções candidatas. Tecnicamente,a função de aptidão é um procedimento que pode ser implementado através de uma expressão matemática ou de simulações. No caso concreto deste trabalho, a função de aptidão é a função Φ (ver Equação 2.3) descrita na secção 2.1.5.

Mecanismos de Seleção

Os mecanismos de seleção são a força motriz dos EAs. Tipicamente, no início de uma pesquisa genética, uma menor pressão de seleção é aplicada, ou seja indivíduos com pior aptidão podem ser selecionados. Desta forma, favorece-se uma maior exploração do espaço de procura. No final do processo de pesquisa, é recomendada uma redução gradual na dispersão do valor de aptidão dos indivíduos selecionados para reduzir o espaço de procura. A seleção deverá dirigir a busca para regiões promissoras. Os tipos mais comuns da seleção são:

- ***Roulette wheel selection***: O processo consiste em selecionar estocasticamente indivíduos de uma geração para criar a base da próxima geração. Os indivíduos mais aptos têm maior probabilidade de sobrevivência do que os mais fracos. Isto reproduz o processo de seleção natural na qual os indivíduos mais aptos tendem a ter uma melhor probabilidade de sobrevivência, sendo assim utilizados para formar a *pool* de acasalamento para a próxima geração. A probabilidade de seleção de um indivíduo é proporcional à sua aptidão, $p_i = Apt_i / \sum Apt$, (ver Figura 2.3). Os indivíduos mais fracos não estão desprovidos de hipóteses de seleção, pois podem ser úteis para as futuras gerações.

- **Seleção determinística**: Estes são procedimentos determinísticos que selecionam os melhores cromossomas dos progenitores e dos descendentes.

- **Selecção por torneio**: Este método escolhe aleatoriamente um conjunto de indivíduos e desse conjunto é selecionado o melhor.

- ***Steady-state reproduction***: Em cada geração, são selecionados alguns indivíduos (com aptidão elevada), tipicamente dois, para a criação de uma

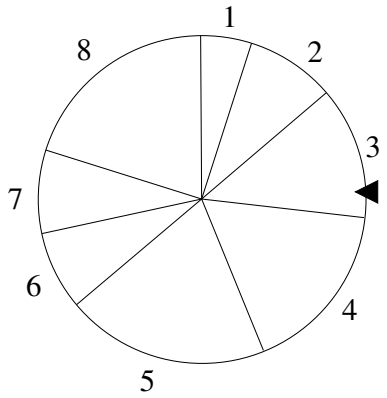


Figura 2.3: *Roulette wheel selection*
Os indivíduos 3,4,5 e 8 possuem melhor aptidão e consequentemente maior probabilidade de serem selecionados.

nova prole. Em seguida, o mesmo número de indivíduos (com baixa aptidão) são removidos e substituídos por descendentes. A restante população sobrevive na nova geração.

Cruzamento

Os operadores de cruzamento ou de recombinação são operadores n-ários que recebem dois ou mais indivíduos e produzem um novo descendente com material genético combinado dos progenitores. Estes operadores têm subjacente a ideia que um novo indivíduo pode ser melhor que os seus parentes se receber as melhores caraterísticas dos seus progenitores. Os cruzamentos ocorrem durante o processo de evolução de acordo com uma probabilidade definida pelo utilizador. Alguns operadores de cruzamento são descritos a seguir.

- **Cruzamento de Um Ponto**
  Este operador seleciona aleatoriamente um ponto de cruzamento de dois progenitores e gera dois descendentes trocando todos os genes que antecedem esse ponto (ver Figura 2.4).

- **Cruzamento de Dois Pontos**
  São selecionados aleatoriamente dois pontos de cruzamento nos progenitores. Tudo entre os dois pontos é trocado entre os progenitores, na geração de

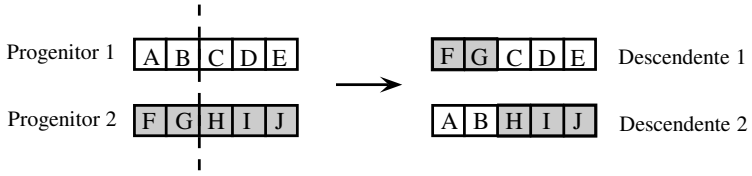


Figura 2.4: Ilustração do operador de cruzamento de um ponto

dois descendentes (ver Figura 2.5).

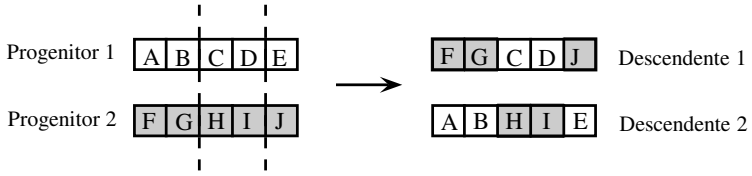


Figura 2.5: Ilustração do operador de cruzamento de dois pontos

- **Cruzamento Uniforme**
  Dois progenitores são utilizados para gerar dois descendentes. Para cada posição no genoma, uma variável binária aleatória é gerada:
  - se o valor desta variável for 1, o primeiro descendente recebe o gene do primeiro progenitor nessa posição, enquanto o segundo descendente recebe o gene do segundo progenitor nessa posição.
  - se o valor desta variável for 0, o papel dos progenitores é invertido.

  Este operador encontra-se representado na Figura 2.6.

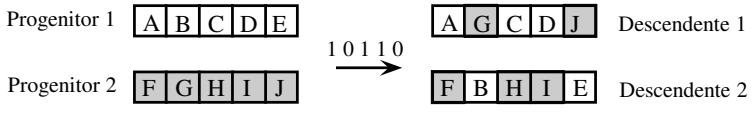


Figura 2.6: Ilustração do operador de cruzamento uniforme

Mutação

Depois da realização de *crossovers* são efetuadas mutações. A mutação é um operador unário utilizado para manter a diversidade genética de uma geração

para outra. Ela ocorre no processo de evolução de acordo com uma probabilidade definida que, geralmente, é baixa. Através do uso de mutações, um ou mais genes de um indivíduo são alterados estocasticamente, ajudando a evitar a estagnação da população num ótimo local. Os operadores de mutação utilizados neste trabalho são:

- **Operador de Mutação Aleatória**
  Este operador apresenta um esquema simples de mutação. Dado um indivíduo $x^{(t)} = \left(x_1^{(t)},...,x_n^{(t)}\right)$ pertencente a t-ésima população P_t, é criada uma distribuição uniforme discreta no intervalo $\left(x_k^{(L)}, x_k^{(U)}\right)$, onde $x_k^{(L)}$ e $x_k^{(U)}$ são os limites superior e inferior para as componentes k $(k = 1,...,n)$. O novo valor valor para a componente ou gene de $x^{(t)}$ é obtido aleatoriamente no intervalo discreto $\left(x_k^{(L)}, x_k^{(U)}\right)$ com probabilidade $1/\left(x_k^{(U)} - x_k^{(L)} + 1\right)$.

- **Operador de Mutação Incremental/Decremental**
  Este operador substitui um dado gene, selecionado aleatoriamente no genoma de um indivíduo, pelo valor seguinte ou pelo valor anterior (com probabilidades iguais) dentro do intervalo permitido.

Seleção de Sobreviventes

Os esquemas de seleção de sobreviventes, também denominados de estratégias de substituição, são utilizados nos algoritmos evolucionários para determinar como os novos indivíduos serão assimilados pela população. O objetivo geral consiste em, tendencialmente, selecionar os indivíduos mais aptos e eliminar os menos aptos, mantendo a população com um tamanho fixo.

No caso particular deste trabalho, o processo de seleção é efetuado definindo uma ordem parcial sobre a população, com base no valor de aptidão de cada indivíduo, e aplicando um esquema de roleta. Em cada geração, 50% dos indivíduos são mantidos da geração anterior, e 50% são criados pela aplicação dos operadores de reprodução.

2.2.2 Algoritmos Evolucionários Multi-Objetivo

Os problemas multiobjetivo (MOP) são aqueles em que se pretende otimizar mais do que uma função objetivo simultaneamente. Isto pode envolver a maximização de todas as funções, a minimização de todas as funções, ou uma combinação de maximização e minimização. Quando existem muitos objetivos a otimizar simultaneamente, possivelmente conflituosos, não existe uma solução ótima, mas sim um conjunto de soluções possíveis de qualidade equivalente. Os EAs, por lidarem simultaneamente com um conjunto de soluções possíveis, permitem obter conjuntos dessas soluções, executando uma única vez o algoritmo. Para além disso, os EAs são menos susceptíveis à forma e continuidade das funções a otimizar. Na secção seguinte serão definidos formalmente os conceitos fundamentais dos algoritmos evolucionários multi-objetivo (MOEA).

Definições Básicas

Um problema multi-objetivo com M objetivos pode ser definido por:

$$\begin{aligned} \text{Max/Min} \quad & f_k(x), && k = 1,..,K; && (2.8)\\ \text{Sujeito a} \quad & g_j(x) \geq 0, && j = 1,...,J; && (2.9)\\ & h_m(x) = 0, && m = 1,...,M; && (2.10)\\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, && i = 1,...,n && (2.11) \end{aligned}$$

onde x é uma solução com n variáveis de decisão $x = (x_1,x_2,...,x_n)$ e f_k são as K funções objetivo. A última condição define restrições às variáveis de decisão e constitui o espaço de decisão (ver Figura 2.7).
Na otimização de um único objetivo, procura-se obter a melhor solução, que seja absolutamente superior a todas as outras alternativas. No caso da otimização multi-objetivo, não existe necessariamente uma solução que seja a melhor com respeito a todos os objetivos, por existirem conflitos entre estes. Uma solução pode ser a melhor no que respeita a um objetivo mas comparativamente pior noutros. Normalmente, existe um conjunto de soluções que não podem ser comparadas por uma relação ordem parcial. Todavia, a **dominância de Pareto** permite definir

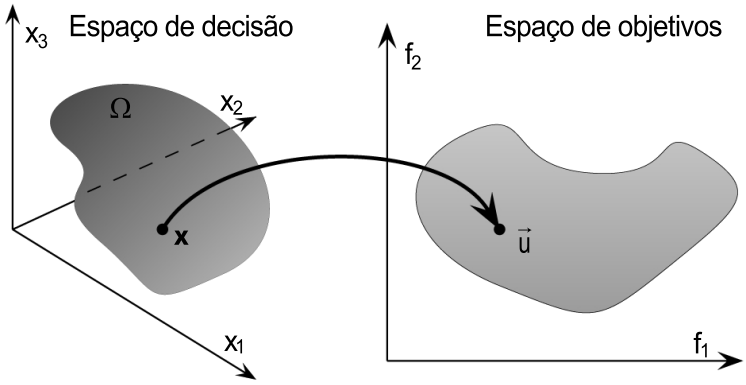


Figura 2.7: Representação do espaço de decisão e correspondente espaço de objetivos

uma relação de dominância entre as soluções. Um vector $\vec{u} = (u_1,u_2,...,u_k)$ domina outra solução $\vec{v} = (v_1,v_2,...,v_k)$, que denotaremos por $\vec{u} \preceq \vec{v}$, se $\vec{u}$ for parcialmente menor do que $\vec{v}$, ou seja,

$$\vec{u} \preceq \vec{v} \quad \equiv \quad \forall i \in \{1,...,k\}, u_i \leq v_i \quad \wedge \quad \exists i \in \{1,...,k\} : u_i < v_i \tag{2.12}$$

Uma solução x é ***Pareto optimal*** se não existir qualquer outra solução y no espaço de decisão Ω tal que para $\vec{u} = F(x) = (f_1(x),...,f_k(x))$ e $\vec{v} = F(y) = (f_1(y),...,f_k(y))$, $\vec{v} \preceq \vec{u}$. Uma solução *Pareto optimal* não pode ser alterada dentro do seu espaço de decisão (genótipo) de modo produzir uma melhoria simultânea de todos os componentes no espaço de objetivos (fenótipo). Por outras palavras, não é possível uma melhoria em qualquer objetivo, sem sacrificar pelo menos uma das outras funções objetivos.

O **conjunto *Pareto optimal*** $\mathcal{P}^*$, é o conjunto de todas as soluções *Pareto optimal*, e é definido por

$$\mathcal{P}^* := \{x \in \Omega | \neg \exists y \in \Omega \quad F(y) \preceq F(x)\} \tag{2.13}$$

A **frente de Pareto** (*Pareto Front*), é a representação no espaço de objetivos do

conjunto *Pareto optimal*, definido formalmente como

$$\mathcal{P}F^* := \{\vec{u} = F(x) = (f_1(x),...,f_k(x)) \,|x \in \mathcal{P}^*\}. \tag{2.14}$$

Os elementos da frente de Pareto são apelidados de vectores não dominados (*nondominated vectores*)

Classificação de MOEAs

Em contraste com a otimização para problemas com um único objetivo, onde a função objetivo e função de aptidão são muitas vezes idênticas, nos MOPs quer a avaliação de *fitness* quer o processo de seleção deverá permitir vários objetivos. Em geral, podem distinguir-se os MOEAs onde os objetivos são considerados separadamente, abordagens que são baseados em técnicas de clássicas de agregação, e os métodos que fazem uso direto do conceito de dominância de Pareto.

- **Seleção agregada com parâmetros de variação**
  A abordagem mais simples de lidar com problemas multi-objetivos é combiná-los num único valor escalar. Estas técnicas são normalmente conhecidas por "funções de agregação", porque combinam todos os objetivos do problema num único. Um exemplo dessa abordagem é a função de aptidão

  $$min \sum_{i=1}^{k} w_i.f_i(x) \tag{2.15}$$

  onde os $w_i \geq 0$ são coeficientes de ponderação que representam a importância relativa que é dada a cada objetivo. Assume-se normalmente que $\sum_{i=1}^{k} w_i = 1$.

- **Seleção baseada em populações**
  Neste tipo de abordagem, a população é usada para diversificar a busca, mas o conceito de dominância de Pareto não é diretamente incorporado no processo de seleção. Em vez de combinar os objetivos num único valor escalar de *fitness*, esta classe de MOEAs alterna os objetivos durante a fase de seleção. Cada vez que um indivíduo é escolhido para reprodução, um objetivo potencialmente diferente decide qual o elemento da população

que será copiado para o *pool* de acasalamento. O exemplo clássico deste tipo de abordagem é o *Vector Evaluated Genetic Algorithm* (VEGA), que consiste basicamente num algoritmo genético simples, com o mecanismo de seleção modificado. Em cada geração, são geradas um número de sub-populações proporcional ao número de funções objetivo. Para um problema com k objetivos, são geradas k sub-populações de tamanho M/k, sendo M o tamanho total da população. Estas sub-populações são então misturadas para obter uma nova população de tamanho M, sobre a qual são aplicados os operadores de cruzamento e mutação.

- **Seleção baseada em Pareto**
  O conceito de cálculo da aptidão de um indivíduo baseada em dominância de Pareto foi sugerida por Goldberg [18]. Ele introduziu um processo de *ranking* iterativo: aos primeiros indivíduos não dominados é atribuído o primeiro lugar, sendo temporariamente removidos da população. Aos indivíduos não dominados seguintes é atribuído o segundo lugar, e assim sucessivamente. Finalmente, a classificação de um indivíduo determina o seu valor de *fitness*. Este procedimento permite que a aptidão esteja relacionada com a totalidade da população, enquanto que, com outras técnicas de agregação, o valor de *fitness* de um indivíduo de é calculado independentemente de outros indivíduos.

  Esta ideia, retomada por vários investigadores, deu origem a diversos esquemas de atribuição de *fitness* baseados na dominância de Pareto, como os utilizados nos algoritmos NSGA II e SPEA2 com estratégias de elitismo diferentes.

  No contexto de um EA com um único objetivo, elitismo significa que as melhores soluções encontradas até ao momento, durante a pesquisa, sobrevivem sempre para a geração seguinte. Neste contexto, todas as soluções não dominadas descobertas por um EA multi-objetivo são consideradas soluções de elite. No entanto, a execução de elitismo em otimização multi-objetivo não é tão simples como na otimização de objetivo único, principalmente devido ao grande número de possíveis soluções elitistas. Os EAs multi-objetivo usam duas estratégias para implementar elitismo: manutenção de soluções

elitistas na população (NSGA II), e o armazenamento de soluções elitistas numa lista externa secundária e reintroduzi-las à população (SPEA2).

Neste trabalho são implementadas soluções de otimização para a configuração dos pesos OSPF que utilizam os métodos elitistas NGSA II e SPEA2.

2.3 A *framework* NetOpt

O modelo de Fortz e Thorup, apresentado na secção 2.1, deu origem a inúmeras extensões que procuram responder a diversas problemáticas do encaminhamento intra-domínio. Uma dessas extensões, apresentada em [36, 32], inclui no processo de otimização um conjunto mais alargado de restrições de QoS. O trabalho consiste na inclusão de uma estrutura de otimização capaz de calcular pesos que otimizam o congestionamento do tráfego e que, simultaneamente, cumpram com requisitos específicos de atraso.
Para disponibilizar estas novas funcionalidades a administradores de rede, foi desenvolvida pelo Centro de Ciências e Tecnologias de Computação (CCTC) da Universidade do Minho, uma aplicação *user friendly*, NetOpt [31], totalmente desenvolvida na linguagem de programação Java.

2.3.1 Otimização Simultânea da Congestão e do Atraso

Os requisitos de atraso para cada par de nós (s,t) na rede são modelados como uma matriz DR. Uma nova função de custo γ^* foi concebida para avaliar o cumprimento de atraso para cada cenário de pesos OSPF. Esta função aplica as mesmas penalizações definidas para Φ^*, substituindo os rácios de carga de link $\ell(a)/c(a)$ pela razão $dc_{st} = Del_{st}/DR_{st}$, onde Del_{st} é a média dos atrasos do trafego de origem s e destino t, e DR_{st} a restrição de atraso entre esses mesmos nós.
A nova função de aptidão γ é normalizada dividindo os valores obtidos pela soma de todos os valores mínimos de atraso fim-a-fim, como se constata na Equação 2.16.

$$\gamma^{*}(w) = \frac{\gamma(w)}{\sum_{(s,t)\in N\times N} \min Del_{st}} \tag{2.16}$$

O problema de configuração de pesos OSPF é redefinido para incluir restrições de atraso. Para uma rede, representada por um grafo $G = (N,A)$, e dados uma matriz de tráfego D e um conjunto de restrições de atraso DR, pretende-se encontrar um conjunto de pesos w que minimizem simultaneamente as funções $\Phi^{*}(w)$ e $\gamma^{*}(w)$. A otimização multi-objetivo é conseguida, utilizando EAs, por meio da função agregadora:

$$f(w) = \alpha\Phi^{*}(w) + (1-\alpha)\gamma^{*}(w),\ \alpha \in [0;\ 1] \tag{2.17}$$

O parâmetro α é utilizado para definir o *trade-off* entre as funções $\Phi^{*}(w)$ e $\gamma^{*}(w)$. Se somente for pretendido otimizar a congestão da rede, é atribuído a α o valor 1. Se, pelo contrário, se pretender somente otimizar o atraso, α será configurado com o valor 0. Desta forma possível otimizar as configurações de rede quer para congestão quer para restrições de atraso.
A otimização multi-objetivo também pode ser conseguida utilizando MOEAs. Para disponibilizar a um administrador de rede um conjunto alargado de soluções, com distintos compromissos entre a otimização da congestão e a otimização dos atrasos, foram desenvolvidas opções de otimização que recorrem aos métodos NSGA-II e SPEA2. Os administradores poderão assim selecionar a solução que lhe é mais adequada. Para além destas opções de otimização de configuração de pesos OSPF, no seguimento do trabalho apresentado em [37], encontra-se prevista a inclusão na *framework* NetOpt de opções de configuração para distintas classes de serviço (CoS).

2.3.2 Descrição da *Framework* NetOpt

A *framework* NetOpt, cuja arquitetura é apresentada na Figura 2.8, permite obter soluções para o problema de otimização de pesos OSPF. Esta *framework* amigável disponibiliza funcionalidades que permitem, por exemplo, otimizar o congestionamento do tráfego, enquanto são cumpridos requisitos específicos de atraso. Para o efeito, a *framework* permite a criação de modelos para redes de longa

distância (WAN), a partir de uma lista de nós e de uma lista de links, geradas, para validação da plataforma, no *Boston University Representation Internet Topology Generator*, ou BRITE [21]. As necessidades de tráfego, bem como as restrições de *delay*, são representadas na forma matricial e podem ser lidas de ficheiros de texto. Para validação da plataforma, estas também podem ser geradas aleatoriamente a partir da definição de um valor médio de congestão expetável, no caso das necessidade de tráfego, e de um fator de escala para as restrições de atraso.

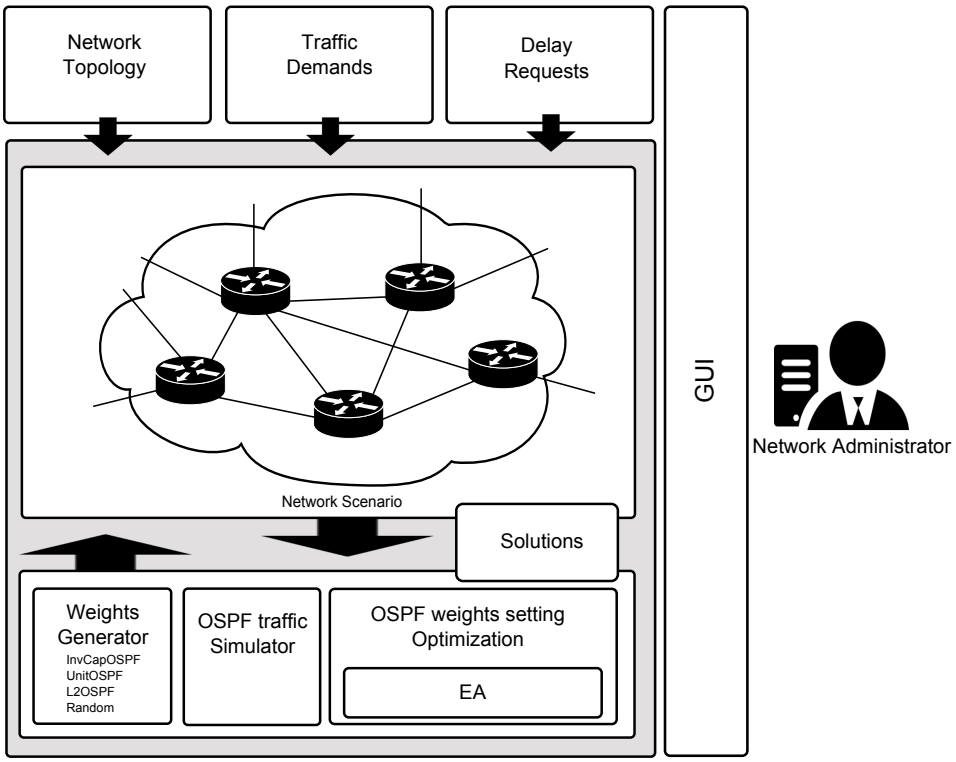


Figura 2.8: Arquitetura da *Framework* NetOpt

Os principais componentes da *framework* são: um gerador de topologia, um gerador de *demands* de tráfego e restrições de *delay*, um simulador de OSPF, um conjunto de heurísticas de otimização e um módulo de execução do EA proposto. Para avaliar a magnitude das melhorias de congestão conseguidas com a otimização de pesos OSPF, também foram implementas heurísticas para gerar configurações de pesos tradicionais:

- InvCap - atribui a cada link um peso inversamente proporcional à sua

capacidade;

- L2 - atribui a cada link pesos diretamente proporcionais à sua distância euclidiana;
- Aleatório - atribui a cada link pesos gerados aleatoriamente;
- Unit - atribui a cada link um peso unitário.

Os pesos são limitados superiormente por um valor inteiro definido pelo utilizador. As funcionalidades de otimização utilizam uma biblioteca de software livre desenvolvida em Java, JEColi [13], que implementa algoritmos de otimização metaheurísticos com enfase em métodos da computação evolucionária. A representação gráfica das topologias conseguida utilizando o JUNG [1], um conjunto de bibliotecas para a modelação, análise e visualização de grafos.
A aplicação é totalmente construída em cima do AIBench, uma *framework* de desenvolvimento de software que nasceu como um projeto de colaboração entre a Universidade de Vigo e da Universidade do Minho. O AIBench é leve, não intrusivo e baseado num modelo *Model-view-controller* (MVC) para aplicações Java. A sua utilização permitiu a ágil integração dos módulos lógicos e de execução com uma interface de fácil utilização. A *framework* NetOpt encontra-se disponível no sítio `http://darwin.di.uminho.pt/netopt/`.
O presente trabalho dedicou-se ao desenvolvimento e inclusão de novas funcionalidades na camada lógica bem como ao nível da interface com o utilizador. Estas novas funcionalidades serão descritas nos capítulos seguintes.

2.4 Sumário

Neste capítulo foram explorados diversos conceitos essenciais ao trabalho desenvolvido, nos quais se incluem os princípios de funcionamento do protocolo de encaminhamento OSPF. Também foram explorados os princípios de funcionamento dos algoritmos evolucionários e de que forma podem ser utilizados para alcançar configurações de pesos OSPF otimizadas para reduzir os níveis de congestão de uma rede. Nesse contexto, foi introduzida a função de avaliação de congestão Φ^* que resulta dos trabalhos de Bernard Fortz e Mikkel Thorup. Foram

ainda identificadas algumas implementações de processos de otimização baseadas nessa mesma função, com particular atenção à *framework* NetOpt que serve de base ao trabalho desenvolvido e que permitiu a obtenção dos resultados que serão apresentados nos capítulos seguintes.

Capítulo 3

Métodos e Resultados

Neste capítulo serão analisados os efeitos nos processos de otimização da inclusão de informações já existentes em populações iniciais, tal como soluções de otimizações anteriores e configurações de pesos normalmente utilizadas, como a configuração de links com pesos inversamente proporcionais à sua capacidade. Serão também analisados os efeitos da alteração dos requisitos de tráfego em topologias com configurações de pesos OSPF otimizadas. Serão igualmente introduzidas novas propostas para métodos de configuração de pesos em domínios OSPF que sejam robustas a alterações de necessidades de tráfego e à falha de links. Incluem-se nestas propostas a otimização das configurações de pesos OSPF Multi-topologia, a otimização da configuração de pesos OSPF para duas matrizes de tráfego e a otimização da configuração de pesos OSPF para a falha do link com maior carga de tráfego.

3.1 Introdução

As redes de comunicação são sistemas dinâmicos cujas condições variam no tempo. Essas alterações resultam de um conjunto diversificado de fatores como alterações nas necessidades de tráfego, alterações na topologia da rede, como falha de links, ou alterações de rotas BGP. Por outro lado, um administrador de rede poderá também necessitar de efetuar alterações ao comportamento da rede para responder a novos SLAs (*Service Level Agreements*), a novas considerações de

QoS ou mesmo para proceder a alterações de *trade-off* entre os diversos objetivos considerados.
Para responder às alterações das condições da rede torna-se necessário desenvolver ferramentas que possam auxiliar um administrador na tomada de decisões e na definição de pesos OSPF que, respondendo às novas condições, permitam preservar bons níveis de congestão. Neste contexto, e dando continuidade ao trabalho apresentado em [32, 36, 35], e descrito na secção 2.3, foram desenvolvidas e analisadas várias propostas, baseadas em técnicas da computação evolucionária, que procuram otimizar configurações de pesos OSPF que sejam robustas a alterações de *demands* e à falha de links.

3.2 Configuração do Processo de Otimização

3.2.1 Implementação do EA

A implementação do algoritmo evolucionário para a otimização dos pesos OSPF tem as seguintes caraterísticas previamente contextualizadas na secção 2.2.

- **Representação**
  A representação de uma solução deve permitir aos operadores genéticos produzir descendentes viáveis. A definição de uma codificação adequada para a representação de um problema de otimização, no contexto de um algoritmo evolucionário, é por vezes difícil. No entanto, este não é o caso para o problema de otimização de pesos OSPF. Uma solução do problema é representada por um ponto no espaço discreto de procura $[1,65535]^{|A|}$. A representação de um conjunto de pesos OSPF é um *array* $w = \left\langle w_1,w_2,...,w_{|A|} \right\rangle$, onde cada $w_i \in [1,65535]$. No trabalho desenvolvido, esse espaço foi reduzido para $[1,20]^{|A|}$, para induzir um maior número de caminhos com o mesmo custo (ECMP) para cada par (s,t) de origem e destino.

- **População inicial**
  As populações são geradas aleatoriamente escolhendo pontos no espaço de procura $[1,20]^{|A|}$. Para além destes pontos, no estudo de alguns problemas, foram incluídas na população inicial configurações de pesos InvCapOSPF,

UnitCapOSPF, L2OSPF, bem como soluções provenientes de otimizações anteriores. A inclusão de tais indivíduos na população inicial será descrita e analisada mais tarde.

- **Função de aptidão**

  A associação de cada solução a um valor de *fitness* é efetuada através da função de aptidão. No trabalho desenvolvido foi utilizada uma função custo baseada na função Φ^*, definida no contexto da secção 2.1, bem como algumas variações agregadoras. Essas variações serão apresentadas no contexto dos diversos problemas explorados.

- **Operadores de mutação e cruzamento** Os operadores de mutação e cruzamento utilizados para produzir novos indivíduos em cada iteração do algoritmo evolucionário são (ver secção 2.2):

  – Mutação aleatória

  – Mutação incremental/decremental

  – Cruzamento uniforme

As populações são constituídas por 100 indivíduos. Em cada iteração, os 50 indivíduos com pior valor de aptidão são substituídos por novos descendentes.
Os resultados apresentados são médias de várias simulações, procurando assim minimizar os efeitos não determinísticos dos EAs. Para cada cenário e diferentes configurações foram sempre executas no mínimo 10 processos de otimização.

3.2.2 Topologias de Redes e Matrizes de Tráfego

Um problema de otimização de pesos OSPF é definido para uma topologia de rede e um conjunto de *demands* de tráfego. No trabalho desenvolvido foram utilizadas duas topologias de rede identificadas como 30_2 (ver Apêndice A) e 30_4 com valência média de nó diferentes ($m = 2, 4$)[1]. Ambas as topologias são constituídas por 30 nós, diferindo somente no número e nas capacidades dos *links*. A topologia

[1]Na teoria dos grafos, o grau ou valência de um vértice é o número de arestas que nele incidem.

30_2 possui 55 links enquanto a topologia 30_4 possui 110 links. Na Figura 3.1 é apresentada uma representação da topologia 30_2. A largura de banda dos links foi gerada por uma distribuição uniforme que assume valores entre 1 e 10 Gbits/s. As redes foram criadas utilizando o *Brite topology generator* [21] com base no modelo Barabasi–Albert [4] com distribuição de cauda pesada.

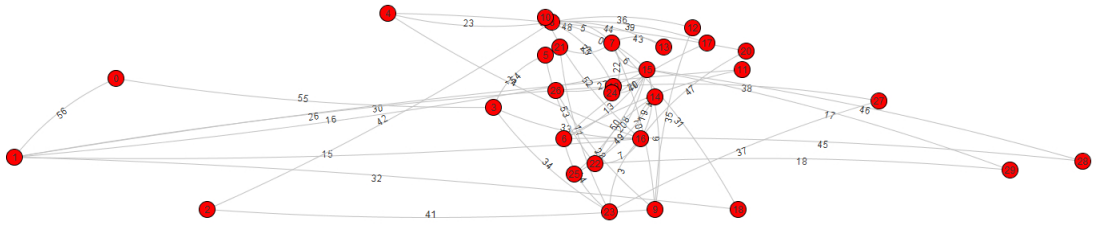


Figura 3.1: Topologia 30_2

Para cada rede, foram criados um conjunto de matrizes de tráfego D e *delay* DR. As matizes de tráfego diferem no valor médio de congestão espetável para cada link, controlado pelo parâmetro D_p, que assume valores em (0.2,0.3,0.4,0.5,0.6)[2], e representam casos onde a topologia é progressivamente sujeita a maiores níveis de carga.

3.3 Alteração de Caminhos

As mudanças nas configurações de pesos OSPF têm impacto na forma como o tráfego é encaminhado pelos vários *links* da topologia, e consequentemente nos níveis de congestão da rede. O estudo de processos de otimização de configurações de pesos OSPF não poderá assim descurar esse facto. Existem várias razões pelas quais as mudanças de peso devem ser evitadas tanto quanto possível [16].

1. Os pesos são frequentemente configurados manualmente, em oposição a algum mecanismo centralizado, e no caso em que centenas de pesos devam ser alterados, existe uma elevada probabilidade de erro humano.

2. É necessário algum tempo para que as informações sobre novos pesos sejam propagadas na rede, para calcular os novos SP, bem como para atualizar as

[2]D0.3 representará a matriz de *demands* gerada com $D_p = 0.3$

tabelas de encaminhamento. Caso sejam alterados um elevado número de pesos, durante o tempo de atualização, poderá criar-se uma instabilidade temporária na rede, com pacotes a chegar fora de ordem, degradando o desempenho dos diversos protocolos que nela operam. Além disso, as alterações podem afetar as rotas anunciadas para outros sistemas autónomos cujo encaminhamento pode então também experimentar flutuações. Este último aspeto será analisado na secção seguinte, dada a sua relevância nas alterações das matrizes de tráfego.

3. Um operador de rede responsável pelo encaminhamento tem de aprovar as alterações. O operador de rede pode ter vários requisitos para o encaminhamento que não são especificados no algoritmo de otimização, por exemplo, que certas exigências tem que ser feita ao longo certos *links*. É muito difícil verificar as consequências de centenas de alterações de pesos, mas as consequências de uma mudança mais reduzida de pesos OSPF deverá ser mais fácil de entender.

Para além das implicações no encaminhamento intra-domínio das alterações de pesos OSPF, não pode ser ignorado os efeitos dessas alterações no encaminhamento inter-domínios.

3.3.1 Encaminhamento Hot-Potato

O desempenho fim-a-fim da Internet depende da estabilidade e eficácia dos protocolos de encaminhamento subjacentes. Uma grande parte do tráfego da Internet atravessa vários Sistemas Autónomos, tornando o desempenho dependente do comportamento do encaminhamento em múltiplos domínios. Nos grandes AS, no núcleo da Internet, os *routers* encaminham os pacotes com base nas informações dos protocolos de encaminhamento intra e inter-domínio. O protocolo *Border Gateway Protocol* (BGP) [29] é utilizado para trocar anúncios de rotas com domínios vizinhos e propagar informações de acessibilidade dentro dos ASs. Os *routers* dentro de um AS podem utilizar um *Interior Gateway Protocol* (IGP), como o OSPF, para determinar como chegar até um outro AS. Um *router* combinará dessa forma as informações BGP e IGP para construir uma tabela de encaminhamento que

mapeia os prefixos de destino nas ligações de saída. Apesar desta arquitetura

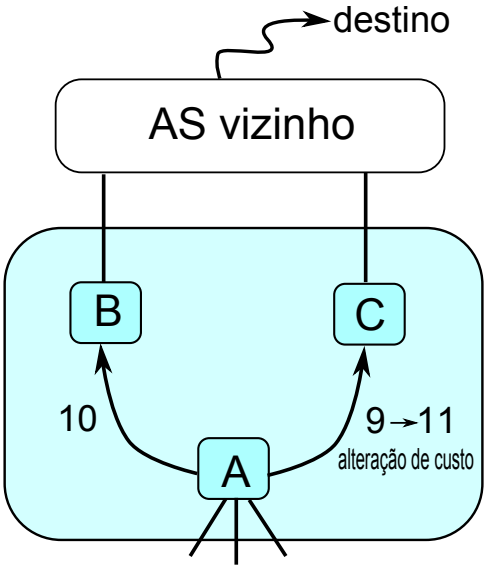


Figura 3.2: Encaminhamento Hot-Potato

de encaminhamento ser, em teoria, capaz de isolar o tráfego global de mudanças de encaminhamento que ocorram dentro de um AS, na prática, a interação entre o encaminhamento intra e inter domínio pode gerar interferência mútuas. O exemplo da Figura 3.2 apresenta duas sessões externas BGP (eBGP) de um AS com um AS vizinho. O AS vizinho anuncia rotas para um dado prefixo de destino. Os dois *routers* B e C propagam as rotas eBGP que aprenderam através de sessões BGP internas (IBGP) que mantêm com o *router* A. O *router* A deverá escolher uma das duas rotas BGP, com qualidade semelhante (por exemplo, com o mesmo número de saltos AS), como rota por defeito para o prefixo de destino. É prática comum, no encaminhamento *Hot-Potato*, que o *router* A encaminhe o tráfego para o ponto de *egress* mais próximo, ou seja, cujo caminho tenha o menor custo intra domínio, neste caso, o *router* C. No entanto, se o custo do caminho para C for alterado de 9 para 11, em resposta a uma falha de ligação no caminho original ou por uma alteração intencional do peso de um *link*, para engenharia de tráfego, esta mudança levará A a selecionar a rota através do ponto de *egress* B.

A falha de um *link*, a falha de um *router* ou uma alteração nos pesos OSPF poderá ter um impacto sobre os caminhos intra domínio e sobre a alcançabili-

dade. Assim, em alguns cenários, estas alterações de condições podem também produzir efeitos sobre as escolhas de encaminhamento BGP, e consequentemente, podem causar alterações de tráfego ou até tornar alguns prefixos inacessíveis a partir de determinados pontos de *ingress* [41]. Por outro lado, alterações no encaminhamento intra domínio podem ter um impacto sobre as rotas BGP que são anunciadas fora do domínio. Um evento IGP que parece insignificante no contexto global da Internet, como por exemplo a alteração de um peso OSPF, pode eventualmente ser visto por *routers* BGP na Internet [28]. Além disso, alguns ISPs usam o atributo BGP *Multi-Exit-Discriminator* (MED), quando possuem várias ligações de *peering* com um mesmo domínio vizinho. O MED é usado para informar os ASs vizinhos da qualidade de cada *router* de *ingress*, e pode conter, por exemplo, o custo OSPF do caminho entre os *routers* de *ingress* e de *egress*. Um AS vizinho irá escolher a rota com menor valor de MED, e assim, neste caso, cada vez que o custo do caminho IGP intra domínio mudar, uma mensagem de BGP UPDATE será emitida, atualizando o valor do MED. Normalmente, um *router* implementa os seguintes passos de decisão para selecionar o ponto de *egress* para encaminhar o tráfego BGP:

1. Prefere as rotas com maior preferência local que reflete as políticas de encaminhamento do domínio.

2. Prefere rotas com o caminho de AS mais curto.

3. Prefere rotas com o menor número de origem, por exemplo, as rotas provenientes do IGP que são mais fiáveis.

4. Prefere rotas com o menor MED

5. Prefere rotas aprendidas por eBGP em detrimento das aprendida por iBGP.

6. Prefere a rota com a menor distância IGP para o ponto de *egress*.

7. Se suportado, aplica *load sharing* entre caminhos. Caso contrário, aplica regras de desempate dependentes do domínio, por exemplo, selecionando o *router* com o menor *egress* ID.

O encaminhamento *Hot-Potato* ocorre quando o *router* de *egress*, para transmissão do tráfego BGP, é selecionado de acordo com as distâncias IGP. Assim, mudanças na configuração do IGP poderá ter algum impacto sobre a forma como o tráfego inter domínio flui através da rede, e consequentemente sobre as matrizes de tráfego que desempenham um papel importante no processo de TE aqui estudado. Torna-se assim importante minimizar as alterações de SP resultantes de alguma alteração no AS.

3.3.2 Alteração Média de Caminhos

No contexto de alguns mecanismos desenvolvidos neste trabalho, para avaliar as alterações de caminhos mais curtos, foi introduzida uma medida para comparar dois SP entre os nós s e t, resultantes de duas configurações de pesos OSPF, definida pela equação 3.1:

$$PathChange_{(s,t)} = \frac{\left|SP_{1(s,t)} \cap SP_{2(s,t)}\right|}{\max\left(\left|SP_{1(s,t)}\right|, \left|SP_{2(s,t)}\right|\right)} \tag{3.1}$$

onde $SP_{1(s,t)}$ e $SP_{2(s,t)}$ representam o conjunto de links no caminho mais curto de s para t na primeira e segunda configuração de pesos, respetivamente.

A média aritmética desta medida para todos os pares (s,t), s,$t \in N$ e $s \neq t$, é denotada por APC (*Average Path Change*), e varia no intervalo [0,1]. Um valor muito perto de 1 traduz alterações reduzidas nos SPs, em contrapartida, valores perto de 0 significam grandes alterações nos caminhos resultantes dos processos de otimização aplicados.

3.4 Hibridação dos EAs ao Nível Populacional

Algumas alterações no contexto de uma topologia requerem que se proceda a uma nova otimização da configuração de pesos OSPF. O tempo de reação depende do tempo de execução do EA, que exige um grande número de iterações, com um custo computacional elevado. Assim, com o objetivo de reduzir o número de iterações necessárias para a convergência do algoritmo evolucionário, foi estudado

o impacto da hibridação dos EAs utilizados ao nível populacional.

Numa heurística de otimização de base populacional, o método de seleção da população inicial é importante, pois afeta a procura nas várias iterações e, muitas vezes, tem uma influência sobre a solução final. Se a priori nenhuma informação estiver disponível sobre a solução ótima, a população inicial é escolhida aleatoriamente usando números pseudo-aleatórios. Todavia, com o intuito de reduzir o tempo de convergência dos EAs, é possível incluir nas populações iniciais indivíduos que reflitam algum conhecimento já existente. Esses conhecimentos podem, no contexto do problema em estudo, resultar de processos de otimização já realizados para uma mesma topologia, ou considerar os mecanismos de atribuição de pesos OSPF utilizados habitualmente.

Os pesos OSPF, atribuídos pelos administradores de rede, são valores inteiros que variam no intervalo [1,65535]. Quanto menor for o peso de um link, maior será a probabilidade do tráfego ser encaminhado através desse mesmo link. A Cisco Systems, Inc, um líder no mercado de fabricantes de equipamentos de rede, sugere que a atribuição de pesos OSPF seja inversamente proporcional à capacidade do link, técnica denominada por InvCapOSPF, promovendo dessa forma uma maior utilização dos links com maiores larguras de banda.

O peso InvCapOSPF $w = 1/c\,(a)$, de um arco a, precisa ser normalizado para um peso w^* que pertença a um intervalo $[w_{min},w_{max}] \subseteq$ [1,65535] definido. O processo de normalização é efetuado através da relação

$$w^* = \frac{w - c_{max}^{-1}}{c_{min}^{-1} - c_{max}^{-1}} \times \Delta + w_{min} \tag{3.2}$$

onde c_{min} e c_{max} são, respetivamente, a menor e maior capacidade dos links da rede, e $\Delta = w_{max} - w_{min}$.

Apesar de não terem em consideração as *demands* de tráfego e, consequentemente, não garantirem um funcionamento eficiente da rede, os pesos InvCapOSPF podem ser utilizado como ponto de partida no processo de otimização OSPF [16]. Para avaliar eventuais melhorias no tempo de convergência, foram realizadas simulações na topologia D30_2 para vários níveis de *demands*. A Tabela 3.1 apresenta a relação entre valores de congestão Φ^* e o número de iterações do EA com e sem a inclusão de pesos InvCapOSPF na população inicial.

	D0.3		D0.4		D0.5	
Iteração	Sem InvCap	Com InvCap	Sem InvCap	Com InvCap	Sem InvCap	Com InvCap
0	241,99	2,28	397,85	46,58	694,39	226,70
10	59,93	2,24	207,36	29,97	406,66	192,42
25	7,56	2,07	80,19	6,87	233,87	109,25
50	2,34	1,89	18,73	3,64	121,62	60,47
100	1,73	1,68	3,47	2,36	56,16	25,92
200	1,52	1,50	2,06	1,95	23,33	9,87
500	1,43	1,41	1,76	1,76	6,76	4,62
1001	1,40	1,40	1,73	1,70	5,28	3,91

Tabela 3.1: Otimização da medida de congestão Φ^* com e sem InvCapOSPF na população inicial

O ganho em número de iterações na inclusão de pesos InvCapOSPF na população inicial é relativamente reduzido. Nos casos analisados, o ganho é de aproximadamente 25 iterações. Todavia, a inclusão de pesos InvCapOSPF permite, nos casos analisados, a obtenção de soluções com menor valor de congestão como pode ser observado na Figura 3.3.

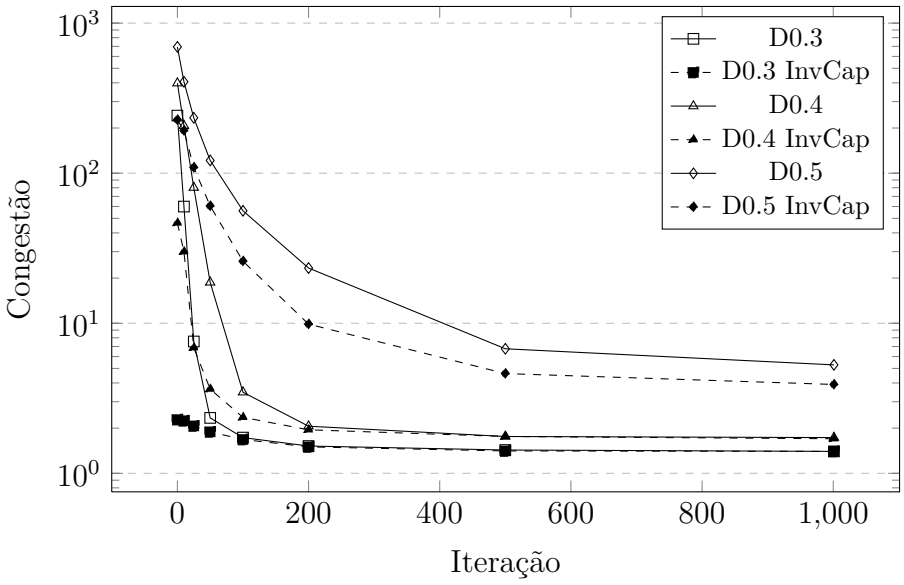


Figura 3.3: Utilização de pesos InvCapOSPF na população inicial

Dependendo do problema, um EA pode ter a tendência de convergir para um mínimo local, ou mesmo para pontos arbitrários, em vez de convergir para o

ótimo global do problema (ver Figura 3.4). Na prática, isso significa que o EA não sabe como sacrificar aptidão de curto prazo para alcançar um *fitness* de longo prazo. A probabilidade de um EA convergir para um ótimo local depende da "paisagem" de *fitness*. Certos problemas podem proporcionar uma ascensão fácil para um ótimo global, enquanto outros podem estimular a função de aptidão a encontrar um ótimo local. Os resultados obtidos permitem constatar que a inclusão de pesos InvCapOSPF na população inicial introduz um estado local inicial que conduz a soluções finais com melhor valor de aptidão do que populações totalmente aleatórias, sendo este ganho mais significativo para níveis mais elevados de *demands*, como pode ser observado na Tabela 3.1 e na Figura 3.3 para níveis de *demands* D0.5.

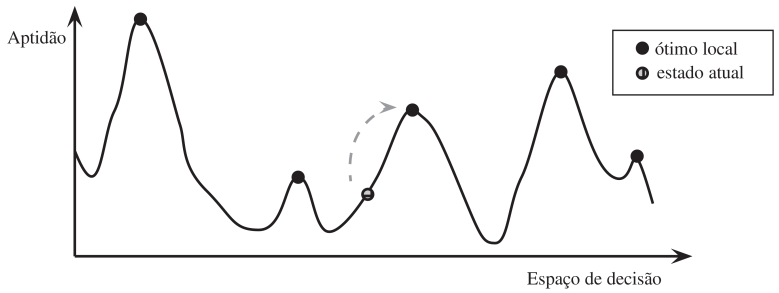


Figura 3.4: Função de aptidão - Ótimo local

Foram realizadas simulações semelhantes para pesos UnitOSPF, um vetor de pesos unitários que traduz uma seleção de caminho mais curto pelo menor número de saltos, bem como para pesos L2OSPF, com pesos diretamente proporcionais ao comprimento do link. Os resultados não demonstraram qualquer ganho na inclusão destes pesos na população inicial. De facto, não seria expetável qualquer ganho provocado por esses pesos dado não terem qualquer relação com a capacidade dos *links*, ao contrário do InvCapOSPF.

Outra estratégia estudada para reduzir o tempo de convergência de um EA para o problema de configuração dos pesos OSPF, consiste na inclusão de indivíduos provenientes de otimizações já realizadas, para uma mesma topologia, na população inicial. Os resultados e análise desta estratégia serão apresentados na secção seguinte.

3.5 Alteração de *Demands*

A engenharia de tráfego é um mecanismo importante para fornecedores de rede Internet que procuram otimizar a entrega de tráfego e o desempenho das suas infra-estruturas de comunicação. As abordagens tradicionais de engenharia de tráfego assumem que a matriz de tráfego na rede é conhecida ou pode ser medida. Teoricamente, se a matriz de tráfego é conhecida, os pesos OSPF atribuídos às ligações podem ser ajustados de acordo com essa matriz, e produzir uma utilização otimizada da rede. Todavia, na prática, medir e prever *demands* de tráfego entre nós é uma tarefa complexa, sendo unicamente possível obter estimativas aproximadas sobre os volumes de tráfego. Muitas aplicações para a Internet, como por exemplo VoIP, Peer-to-Peer e IPTV, são caraterizadas por comportamentos de tráfego altamente variável ao longo do tempo. Os requisitos de largura de banda destas aplicações mudam continuamente. Por isso, torna-se complexo medir e prever com uma exatidão considerável as *demands* de tráfego na rede.

Nesta secção serão analisados os impactos do aumento das necessidades de tráfego em topologias cujos pesos já se encontram previamente otimizados.

3.5.1 Re-otimização para Aumento de *Demands*

Tomando como ponto de partida uma solução obtida anteriormente para o problema de configuração de pesos OSPF, com pesos w e população final P, foram assumidos vários níveis de incrementos nas *demands* de tráfego. Os incrementos, $inc \in (15\%, 30\%, 45\%, 60\%, 70\%, 75\%)$, têm uma tolerância de 5%, ou seja, para cada par (s,t) de origem e destino, com *demands* d_{st}, foram calculadas novas *demands* d^*_{st}, tais que $d^*_{st} = d_{st} \times (1 + inc + \delta)$, com δ selecionado aleatoriamente no intervalo $[-0.025,0.025]$. A matriz de tráfego inicial usada neste estudo é D0.3, sendo a medida de congestão do cenário inicial 1,3946. As otimizações realizada procuram apenas melhorar a medida de congestão Φ^*.

Para cada aumento de *demands*, o processo de otimização foi executado fornecendo distintas populações iniciais, obtidas combinando indivíduos aleatoriamente selecionados de P com indivíduos gerados aleatoriamente. Foram avaliadas combinações, com de 0%, 10%, 50% e 100% de P. Cada configuração foi executada

várias vezes sendo os resultados apresentados as médias dos valores observados.

Iteração	15% de aumento				30% de aumento			
	0% Pop.	10% Pop.	50% Pop.	100% Pop.	0% Pop.	10% Pop.	50% Pop.	100% Pop.
0	358,069	1,612	1,612	1,612	493,671	1,944	1,944	1,944
10	150,285	1,609	1,605	1,609	157,049	1,939	1,931	1,937
50	4,135	1,595	1,591	1,599	5,358	1,917	1,912	1,914
100	2,348	1,587	1,581	1,589	2,618	1,906	1,897	1,905
200	1,877	1,583	1,576	1,580	2,194	1,888	1,888	1,894
500	1,685	1,579	1,576	1,577	1,949	1,868	1,880	1,861
1001	1,624	1,575	1,551	1,577	1,841	1,849	1,865	1,847
Iteração	**45% de aumento**				**60% de aumento**			
	0% Pop.	**10% Pop.**	**50% Pop.**	**100% Pop.**	**0% Pop.**	**10% Pop.**	**50% Pop.**	**100% Pop.**
0	616,044	2,596	2,596	2,596	645,936	5,854	5,854	5,854
10	318,382	2,571	2,546	2,544	295,267	5,253	5,353	5,173
50	38,998	2,502	2,506	2,481	55,751	4,766	4,300	4,333
100	5,635	2,469	2,464	2,433	17,781	4,464	3,973	4,240
200	3,114	2,436	2,433	2,420	6,990	4,243	3,908	4,048
500	2,487	2,430	2,409	2,389	4,074	4,126	3,853	3,943
1001	2,446	2,403	2,381	2,389	2,938	3,951	3,848	3,823
Iteração	**70% de aumento**				**75% de aumento**			
	0% Pop.	**10% Pop.**	**50% Pop.**	**100% Pop.**	**0% Pop.**	**10% Pop.**	**50% Pop.**	**100% Pop.**
0	680,244	15,448	15,448	15,448	804,592	29,156	29,156	29,156
10	505,972	13,069	13,163	12,686	393,120	20,057	21,993	22,300
50	153,530	10,064	9,812	9,435	78,620	16,310	17,214	17,883
100	70,710	9,418	8,731	8,237	35,382	14,707	15,064	14,818
200	16,756	7,924	7,832	7,365	16,577	13,398	14,245	14,128
500	8,341	7,619	7,286	7,151	11,136	12,756	13,426	13,324
1001	7,615	7,312	6,910	7,133	9,314	12,345	12,985	12,925

Tabela 3.2: Alteração de *demands* D0.3 na topologia 30_2

Os resultados apresentados na Tabela 3.2 e Figura 3.5, para a topologia 30_2, permitem observar que não existem diferenças significativas no comportamento do algoritmo para as diferentes combinações de populações de P (10%, 50%, 100%). Este resultado é consequência do processo de seleção de sobreviventes em EAs. De uma geração para a seguinte, sobrevivem somente os melhores indivíduos da população. Por outro lado, ao longo do processo de convergência dos EAs, a diversidade da população diminui gradualmente. Perante estes resultados, e para aumentar a diversidade inicial, nas simulações seguintes serão apenas consideradas populações totalmente aleatórias e populações que contenham 10% de soluções provenientes de processos de otimização já realizados.

O estudo da tolerância à variação de *demands* incluiu mais dois conjuntos de cenários. A amplitude dos intervalo de variação dos aumentos de requisitos de tráfego foi alargado, passando dos anteriores 5% para 20%, ou seja, para cada

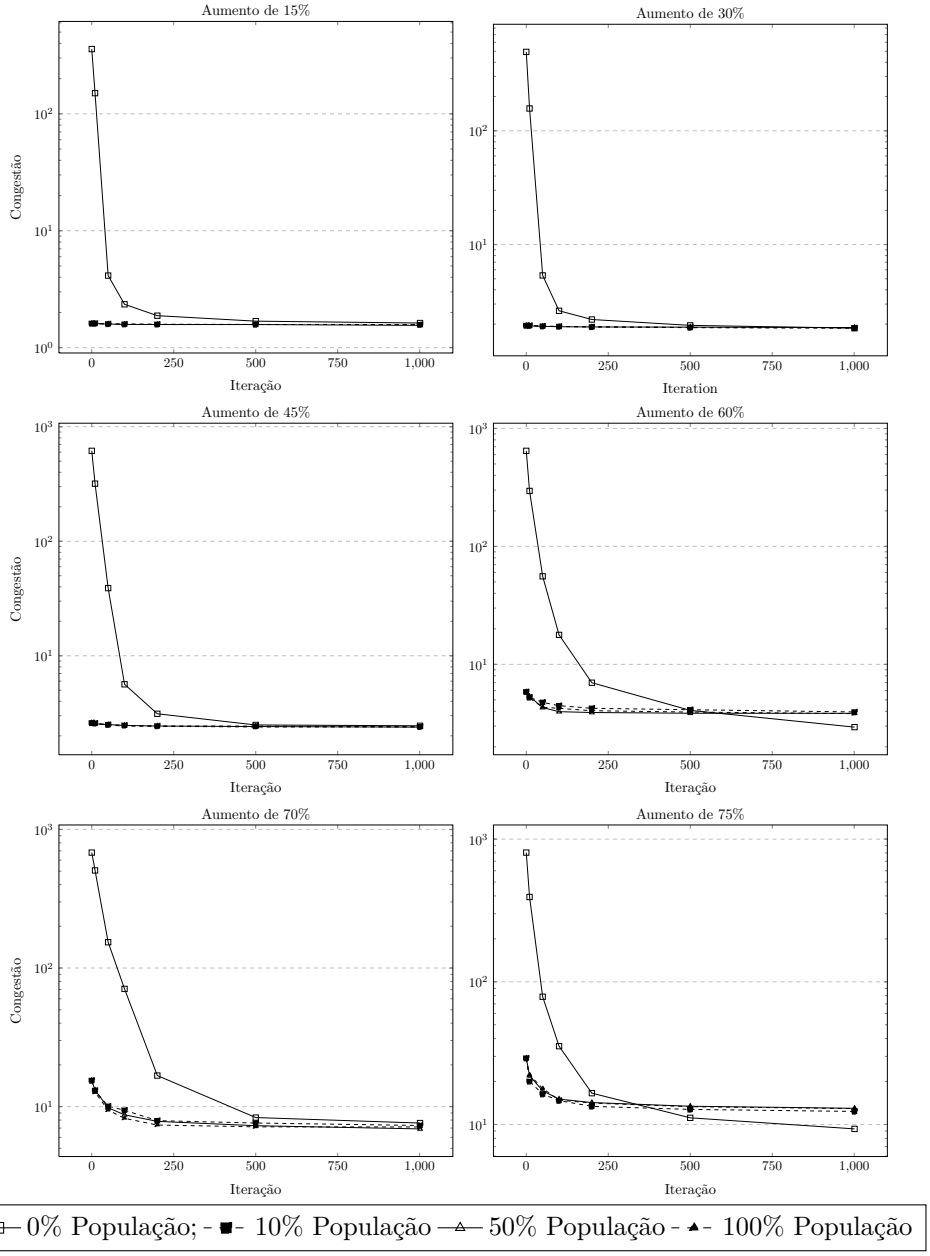


Figura 3.5: Distintas percentagens de soluções anteriores na população inicial (relativa à Tabela 3.2)

incremento $inc \in$ (20%,40%,60%,70%,80%), os requisito de tráfego entre cada par de nós com origem s e destino t, são escolhidos aleatoriamente num intervalo $[inc - 0.1, inc + 0.1]$, procurando contrariar eventuais efeitos de aumentos excessivamente lineares. Este novo cenário foi aplicado às topologias 30_2 e 30_4 que diferem no grau médio dos nós. As Tabelas 3.3 e 3.4 apresentam os resultados obtidos para as topologia 30_2 e 30_4, respetivamente. Os cenários iniciais têm medida de congestão 1,426 na topologia 30_2 e 1,513 na topologia 30_4.

	20% de aumento		**40% de aumento**		**60% de aumento**		**70% de aumento**		**80% de aumento**	
It	0%	**10%**	0%	**10%**	0%	**10%**	0%	**10%**	0%	**10%**
0	408,854	**1,850**	571,703	**2,463**	642,036	**7,514**	836,299	**34,801**	952,596	**67,086**
10	138,164	**1,844**	288,761	**2,430**	360,910	**6,795**	447,676	**29,113**	558,522	**61,172**
50	7,883	**1,802**	40,811	**2,350**	86,977	**5,093**	115,511	**17,853**	190,939	**40,076**
100	2,615	**1,778**	8,536	**2,304**	22,040	**4,501**	40,388	**12,551**	92,371	**32,886**
250	1,928	**1,755**	3,260	**2,233**	6,322	**4,032**	14,078	**8,990**	35,568	**27,906**
500	1,788	**1,742**	2,355	**2,186**	4,486	**3,895**	9,897	**7,477**	23,864	**24,545**
750	1,733	**1,736**	2,265	**2,164**	4,105	**3,826**	9,046	**7,186**	22,417	**24,059**
1000	1,716	**1,729**	2,228	**2,154**	3,682	**3,807**	8,189	**7,042**	21,385	**23,511**
1500	1,706	**1,721**	2,178	**2,134**	3,372	**3,799**	7,410	**6,757**	16,889	**23,092**
2001	1,701	**1,717**	2,155	**2,125**	3,295	**3,776**	6,635	**6,658**	16,111	**22,826**

Tabela 3.3: Alteração de *demands* na topologia 30_2

	20% de aumento		**40% de aumento**		**60% de aumento**		**70% de aumento**		**80% de aumento**	
It.	0%	**10%**	0%	**10%**	0%	**10%**	0%	**10%**	0%	**10%**
0	1922,039	**2,128**	2321,463	**7,629**	2590,071	**27,556**	2805,017	**48,369**	2744,351	**86,379**
10	1397,551	**2,116**	1689,011	**7,343**	1826,602	**26,047**	1967,141	**46,854**	2157,554	**82,187**
50	595,967	**2,017**	817,584	**6,310**	1007,034	**21,486**	997,680	**38,958**	1172,237	**68,689**
100	302,374	**1,980**	499,870	**5,908**	627,304	**18,566**	673,022	**35,795**	775,023	**62,814**
250	70,468	**1,937**	156,470	**5,255**	252,713	**15,354**	277,735	**31,444**	372,515	**51,319**
500	10,023	**1,901**	47,796	**4,359**	101,426	**13,151**	107,518	**26,424**	182,030	**43,171**
750	4,678	**1,881**	20,911	**4,057**	59,586	**11,985**	61,097	**23,502**	117,726	**35,874**
1000	3,768	**1,861**	9,651	**3,656**	40,458	**11,188**	41,681	**21,137**	88,016	**30,633**
1500	2,278	**1,834**	5,274	**3,250**	17,212	**10,115**	21,850	**15,971**	55,052	**21,232**
2001	2,152	**1,821**	3,057	**2,836**	11,020	**8,719**	16,630	**13,064**	40,910	**19,308**

Tabela 3.4: Alteração de *demands* na topologia 30_4

Os resultados, para as duas topologias 30_2 e 30_4, permitem observar que o número de pesos a otimizar se reflete no tempo de convergência. O tempo necessário para a convergência do EA é maior para topologias com mais links. No caso da topologia 30_2, com 55 *links*, o algoritmo converge na generalidade ao fim de 1000 iterações. Em contrapartida, para a topologia 30_4, com 110 *links*,

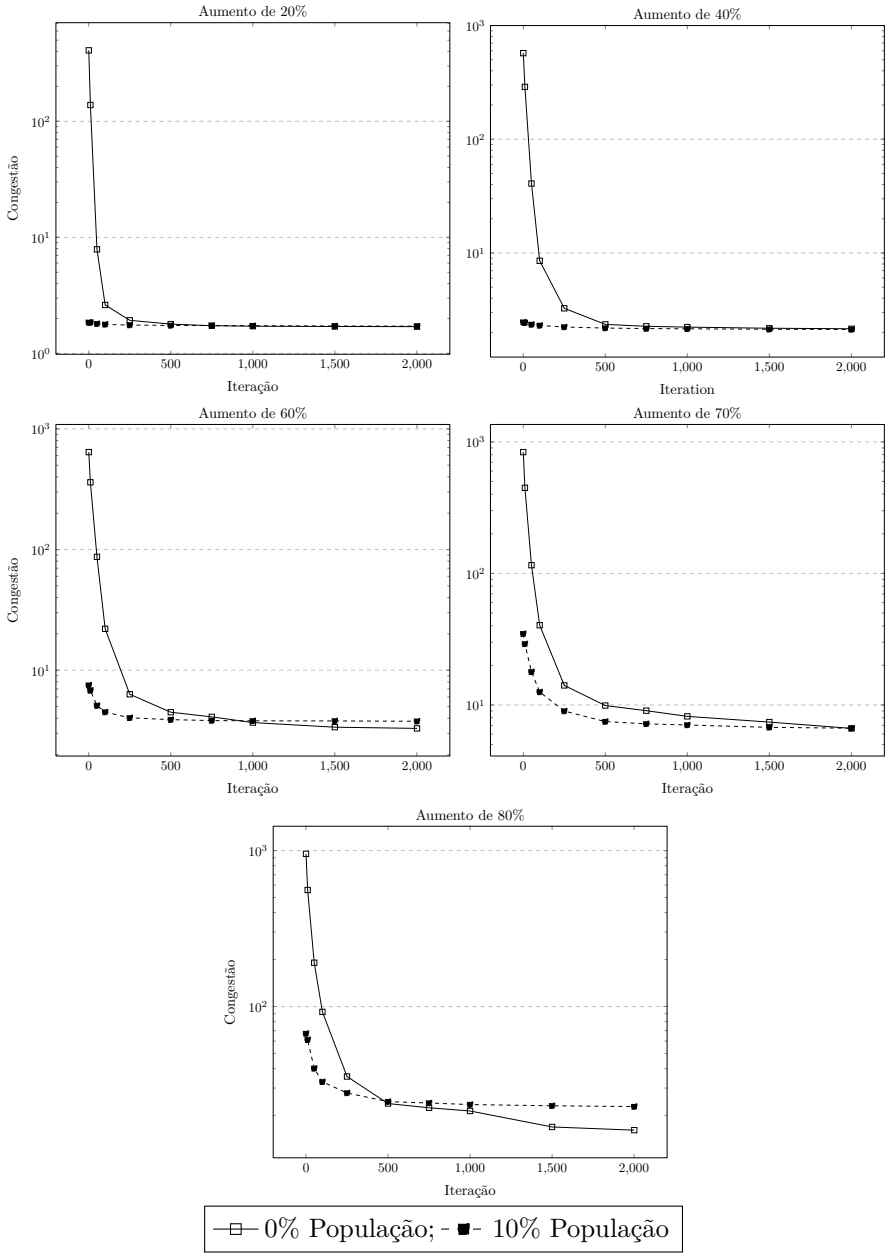


Figura 3.6: Alteração de *demands* na topologia 30_2

o algoritmo converge somente ao fim de cerca de 2000 iterações.
Nos problemas mais fáceis, ou seja, quando se verifica um aumento reduzido dos requisitos de tráfego (20% e 40%), a rede apresenta um aumento no nível de congestão que poderá ser descurado pelo administrador, como pode ser observado na Tabela 3.5. Valores da medida de congestão Φ^* inferiores a 10 2/3 são considerados aceitáveis (ver secção 2.1). Nesses casos, é preferível não efetuar alterações de pesos, sacrificando o nível de congestão, mas evitando efeitos indesejáveis provocados por essas alterações.

	Aumento de demands					
Topologia	**0%**	**20%**	**40%**	**60%**	**70%**	**80%**
D30_2	1,426	1,85	2,463	7,514	34,801	67,086
D30_4	1,513	2,128	7,629	27,556	48,369	86,379

Tabela 3.5: Níveis de congestão após aumento de *demands*

Nos problemas mais difíceis, para os quais é necessário otimizar os pesos OSPF, o recurso a indivíduos provenientes de otimizações anteriores, permite uma redução significativa de iterações. Nos casos analisados o tempo de procura é reduzido em cerca de 1000 iterações na topologia 30_4. Todavia, em certos casos (*ver* 80% de aumento na Tabela 3.3), a introdução de tais indivíduos na população inicial pode provocar a convergência para uma solução não ótima. Esta conclusão é mais facilmente observável nas Figuras 3.5 e 3.6 nos casos de maior aumento das necessidades de tráfego.
Para avaliar o impacto na definição de caminhos mais curtos, foram comparados os caminhos mais curtos para cada par de nós de origem e destino, antes e depois do processo de otimização, nas várias configurações de população e aumento de necessidade de tráfego. A comparação dos caminhos mais curtos foi efetuada utilizando a medida *Average Path Change* (APC) definida na secção 3.3. Os resultados obtidos, apresentados nas Tabelas 3.6 e 3.7, permitem constatar que em otimizações na quais foi efetuado *seeding* da população inicial, com soluções do processo de otimização inicial, registou-se um menor número de alterações nos caminhos mais curtos (mais perto de 1), quando comparadas com as alterações efetuadas após otimizações com populações iniciais totalmente aleatórias.

Topologia	**Aumento de 20%**		**Aumento de 40%**		**Aumento de 60%**	
	0% Pop.	**10% Pop.**	0% Pop.	**10% Pop.**	0% Pop.	**10% Pop.**
30_2	0,839	**0,913**	0,839	**0,903**	0,850	**0,914**
30_4	0,688	**0,932**	0,679	**0,922**	0,668	**0,872**

Tabela 3.6: Valor da medida APC para aumentos de *demands* de 20%, 40% e 60%

Topologia	**Aumento de 70%**		**Aumento de 80%**	
	0% Pop.	**10% Pop.**	0% Pop.	**10% Pop.**
30_2	0,826	**0,900**	0,829	**0,902**
30_4	0,678	**0,843**	0,677	**0,844**

Tabela 3.7: Valor da medida APC para aumentos de *demands* de 70% e 80%

O recurso a populações obtidas de processos de otimização anteriores, para inclusão na população inicial de uma nova otimização, não se apresenta assim como uma solução que garanta, à partida, uma melhoria do desempenho do algoritmo evolucionário. Apesar destas permitirem reduzir o número de alterações necessárias nos caminhos mais curtos, e diminuir o número de iterações necessárias à convergência do algoritmo, este processo introduz informações nas configurações das populações iniciais que poderão em certos casos provocar a não convergência do algoritmo para uma solução ótima.

Quando as alterações de *demands* forem significativas, tornando-se necessário proceder a um novo processo de otimização, a população inicial poderá ser configurada com pesos InvCapOSPF para se obter ganhos de desempenho, mesmo que reduzidos.

No seguimento dos resultados obtidos, na secção seguinte é apresentada uma proposta baseada em encaminhamento multi-topologia que visa melhorar os níveis de congestão de uma rede e, simultaneamente, aumentar a tolerância a alterações de necessidade de tráfego.

3.6 OSPF Multi-Topologia

Nas redes OSPF, para lidar com a mudança da matriz de tráfego, é necessário redefinir os pesos das ligações, usando por exemplo um método de otimização. Todavia, as alterações de pesos podem conduzir a uma instabilidade na rede bem como a alterações de tráfego provocadas por mudanças nos protocolo externos, como analisado anteriormente. Para atenuar o efeito da atualização de pesos quando a matriz de tráfego sofre alterações, os autores em [16] sugerem re-otimizar os pesos das ligações para a nova matrizes de *demands*, alterando o menor número possível de pesos. Todavia, qualquer alteração de pesos, mesmo que reduzida, deverá ser evitada. Neste contexto é proposta uma configuração de encaminhamento de tráfego baseada em multi-topologia OSPF, que para além de procurar reduzir os níveis de congestão da rede, permite que, em cenários de alterações nas *demands* de tráfego, esses mesmos níveis se mantenham aceitáveis, sem com isso ser necessário efetuar alterações nas configurações de pesos OSPF.

3.6.1 Encaminhamento OSPF em Multi-Topologia

O encaminhamento OSPF em multi-topologia associa a cada topologia lógica, definida sobre uma topologia física existente, um conjunto de pesos diferentes. Apesar de existirem semelhanças com o TOS-OSPF (*Type Of Service OSPF*)[24], estes apresentam algumas diferenças [27]:

- No TOS-OSPF, o TOS ou *DiffServ Code Point* (DSCP) no cabeçalho IP é mapeado diretamente para o correspondente SPF OSPF e tabela de encaminhamento. Isto limita o número e definição das topologias para os 16 valores de TOS especificados no TOS-OSPF. No encaminhamento Multi-Topologia esta correspondência não existe, sendo o mapeamento livre.

- A distribuição do tráfego entre as diversas topologias não é baseada em classes de serviço [37], mas, na proposta aqui apresentada, numa distribuição equitativa baseada em fluxos. Este aspeto é particularmente importante caso exista muito mais tráfego *best-effort* do que tráfego marcado com DSCP *Expedited Forwarding* (EF) ou *Assured Forwarding* (AF) [25].

- No encaminhamento de tráfego TOS-OSPF, o tráfego cujo o prefixo é inacessível na tabela de encaminhamento do TOS correspondente reverte para a tabela de encaminhamento TOS 0. No encaminhamento Multi-topologia esta reversão é opcional.

- No encaminhamento TOS-OSPF, ligações individuais ou prefixos não podem ser excluído de uma topologia. Se os LSAs tiverem a opção T-bit definida, todos os links ou prefixos são anunciados de forma explícita ou, por defeito, no TOS 0. No encaminhamento Multi-topologia, os links ou prefixos que não são anunciados de forma explícita para uma topologia não existem nessa topologia.

3.6.2 Balanceamento do Tráfego pelas Topologias

O recurso a multi-topologias para responder ao problema da congestão de uma rede, requer a definição de um mecanismo de balanceamento de tráfego entre as diversas topologias lógicas. Em [44] é apresentada uma proposta que procura uma divisão ótima do tráfego por topologias lógicas parciais. Em contrapartida, a proposta que é aqui apresentada exige um processo de configuração simples e utiliza mecanismos já existentes nos dispositivos de roteamento.
O balanceamento de tráfego pelas várias topologias é efetuado por fluxo, ou seja, com base num endereço de origem e destino. O mecanismo exato da divisão pode ser algo complicado, dependendo da implementação [43]. Existem vários algoritmos que efetuam a divisão de tráfego com base nos fluxos. O *Modul-N Hash*, o *Hash-Threshold* e o *Highest Random Weight* (HRW), introduzidos em [42] e [19], são alguns exemplos. Todos estes algoritmos mantêm um mesmo fluxo numa mesma topologia. As principais diferenças entre estes algoritmos residem no fator de perturbação e na complexidade computacional. O fator de perturbação é uma medida que quantifica o número de fluxos cujo caminho é alterado com a adição ou supressão de *next-hop*. De acordo com [42, 19], o HRW tem o melhor fator de perturbação, mas possui a complexidade computacional

mais elevada. O *Hash-Threshold* tem um fator de perturbação quase tão bom quanto o HRW sendo computacionalmente menos complexo. Em [6] e [34], os autores propõem o Threshold-Hash (CRC16) como o melhor algoritmo *static load balancing*. É importante salientar que todos os equipamentos de encaminhamento que implementam MT-OSPF não requererão qualquer alteração para implementar a proposta que é apresentada neste trabalho.

3.6.3 Cálculo de Aptidão

No modelo matemático para otimização multi-topologia sobre a rede física, representada pelo grafo $G = (N,A)$, são definidas T topologias lógicas $G_\tau = (N_\tau,A_\tau)$ com $N_\tau \subseteq N$,$A_\tau \subseteq A$ e $\tau = 1..T$. As *demands* D são divididas uniformemente por fluxo em matrizes de *demands* D_τ que são mapeadas nas T topologias lógicas, e em que cada elemento d^τ_{st} representa o tráfego com origem s e destino t que flui na topologia τ. Para cada topologia lógica são definidos um conjunto de pesos w_τ, que concatenam no genótipo num único cromossoma $w = \left(w_{(1,1)},...,w_{(n,1)},w_{(2,1)},...,w_{(n,T)}\right)$, sendo $n = |N|$. Para cada arco $a \in A$, $f^\tau_{st,a}$ representa a quantidade de tráfego de origem s e destino t que passa por a na topologia τ. A carga total na topologia τ de um arco a é representada por $\ell_\tau(a)$ e pode ser definida por

$$\ell_\tau(a) = \sum_{(s,t)\in N\times N} f^\tau_{st,a} \quad , \tag{3.3}$$

e a carga total de um arco a na topologia física $\ell(a)$ resulta da adição das cargas totais nas topologias lógicas, isto é,

$$\ell(a) = \sum_{\tau=1..T} \ell_\tau(a) \tag{3.4}$$

O cálculo da aptidão $f(w)$ para uma configuração de conjuntos de pesos $w = \left(w_{(1,1)},...,w_{(n,1)},w_{(2,1)},...,w_{(n,T)}\right)$ segue a definição usual da função Φ^*, definida na secção 2.1. Na Figura 3.7 é apresentado um esquema representativo do processo de cálculo de aptidão. A cada uma das T topologias é atribuída uma configuração de pesos W_1,...,W_T. Os T requisitos de tráfego, que resultam da divisão dos requisitos totais de tráfego pelas T topologias, são distribuídos pelos caminhos mais curtos

obtidos para as topologias em função das configurações de pesos W_τ. A carga total de cada arco na topologia física é obtida de acordo com a equação 3.4.

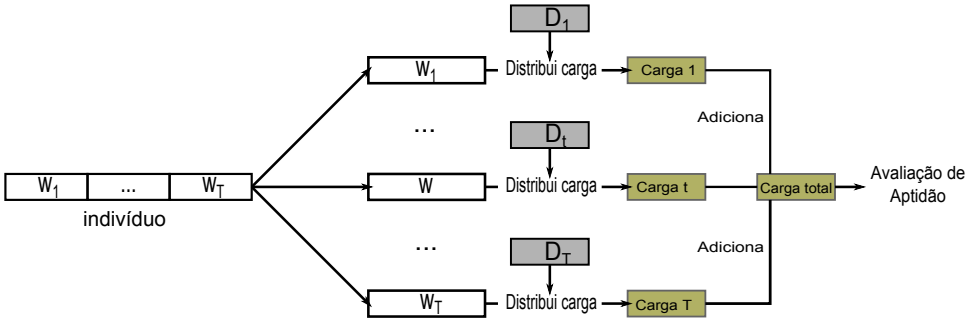


Figura 3.7: Cálculo de aptidão multi-topologia

3.6.4 Otimização de Congestão em MT-OSPF

Para avaliar o modelo, foram realizadas simulações para a topologia 30_2, considerando matrizes de tráfego com diferentes níveis de *demands*. Foram testadas configurações de encaminhamento de tráfego com 2, 3 e 4 topologias. As *demands* de tráfego foram distribuídas uniformemente pelas várias topologias. Os resultados obtidos, que são as médias de 10 simulações para cada cenário, encontram-se listados na Tabela 3.8, com a respetiva representação gráfica na Figura 3.8.

N.º de topologias	**Demands Totais**			
	D0.3	D0.4	D0.5	D0.6
1	1,401	1,707	4,370	34,270
2	1,363	1,634	2,419	6,230
3	1,357	1,619	2,343	5,926
4	1,360	1,613	2,365	5,338

Tabela 3.8: Congestão em Encaminhamento OSPF Multi-Topologia

Os resultados mostram que, para além de reduzir a congestão, existe neste caso um número de topologias a partir do qual não se regista qualquer ganho. O número de topologias necessárias para reduzir o nível de congestão da rede depende da dificuldade do problema. Para as *demands* D0.3, o uso de mais do que duas

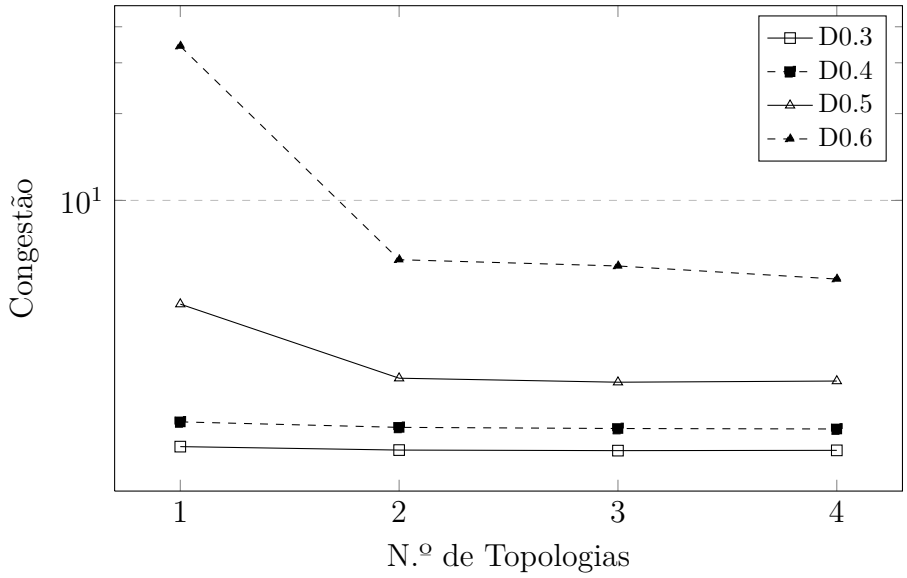


Figura 3.8: Congestão Φ^* em encaminhamento OSPF Multi-Topologia

topologias não traduz ganhos significativos (ver Figura 3.8). O recurso a mais topologias, e consequentemente a mais tabelas de encaminhamento, traduzir-se-ia somente num aumento das necessidades de processamento nos nós da rede. Em contra partida, para *demands* D0.6 ainda é possível observar ganhos com a utilização de quatro topologias, mesmo que residuais. Neste último caso, com duas topologias, é já possível passar de valores de congestão superiores ao limite de congestão aceitável, 10 2/3 (ver secção 2.1), para valores de congestão aceitáveis que garantam uma melhor distribuição do tráfego.

Com a otimização de pesos para MT-OSPF é conseguido um melhor aproveitamento dos recursos da topologia. As diferentes configurações de pesos aplicadas a cada topologia originam uma maior diversidade de caminhos mais curtos entre as várias origens e destinos. Na Tabela 3.9 são comparados os caminhos mais curtos obtidos, por otimização dos pesos OSPF, entre cada uma das quatro topologias lógicas (T1,T2,T3 e T4), para o caso de *demands* D0.6. A medida de comparação é a definida na secção 3.3, e varia entre 0 e 1. Quanto mais perto de 1 for o valor da medida, maior será o número médio de links comuns entre caminhos mais curtos para cada par (s,t) de origem e destino. Os resultados permitem constatar

que os caminhos mais curtos possuem, em média, e para cada par de topologias comparadas, diferenças de cerca de 40%.

Topologia	T2	T3	T4
T1	0,591	0,639	0,598
T2		0,590	0,584
T3			0,699

Tabela 3.9: Comparação de caminhos mais curtos em MT-OSPF com 4 topologias (topologia 30_2 com D0.6).

Uma melhor exploração dos recursos da topologia traduz-se numa melhor taxa de ocupação dos links. A distribuição da taxa de utilização de cada link na topologia física, para este caso particular, é apresentada na Figura 3.9. A distribuição permite constatar que cerca de 70% dos links têm uma taxa de utilização entre os 66% (2/3) e os 100% (1).

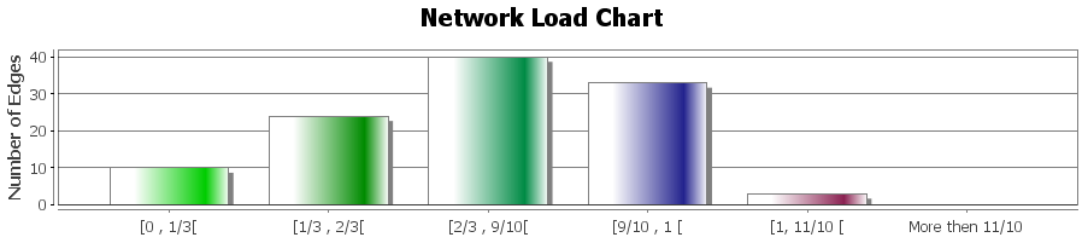


Figura 3.9: Distribuição da taxa de utilização dos *links* em MT-OSPF (topologia 30_2, D0.6 e utilização de 4 topologias)

Podemos também comparar a distribuição da taxa de utilização dos *links* no cenário com quatro multi-topologia (Figura 3.9), com a distribuição da taxa de utilização dos *links* no cenário sem multi-topologia (Figura 3.10), ou seja, com uma configuração de pesos OSPF otimizada unicamente para uma a topologia física. É possível constatar que, para uma mesma matriz de requisitos de tráfego (D0.6), na otimização para MT-OSPF existe um menor número de *links* com taxa de ocupação inferior a 2/3 da capacidade, e que, consequentemente, existe um menor número de *links* nos quais o volume de tráfego é superior à capacidade.
A otimização das configurações de pesos MT-OSPF apresenta-se assim como uma boa solução para reduzir os níveis de congestão num domínio OSPF utilizando

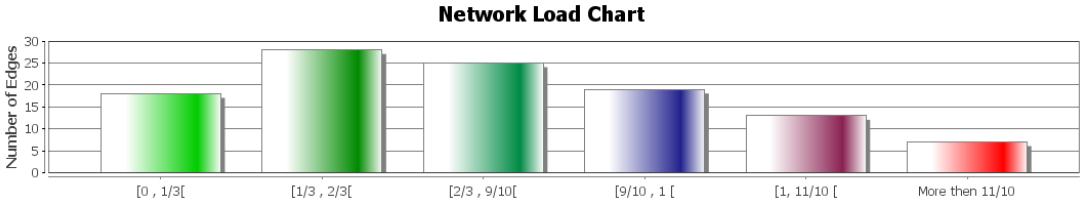


Figura 3.10: Distribuição da taxa de utilização de links sem MT-OSPF (topologia 30_2, D0.6)

os mecanismos já existentes.

3.6.5 Alteração de *Demands* em MT-OSPF

Para testar a tolerância do modelo proposto a alterações de tráfego, foram efetuados diversos incrementos nas *demands* de tráfego. A congestão foi avaliada nas várias configurações, com os novos requisitos de tráfego, sem proceder a qualquer alteração de pesos OSPF. Alguns dos resultados obtidos encontram-se apresentados na Tabela 3.10.

N.º de topologias	**Demands Totais**			
	D0.3 + 60 %	**D0.4 + 50%**	**D0.5 + 20%**	**D0.6 + 10 %**
1	8,327	34,005	58,351	131,667
2	6,016	9,870	9,912	32,219
3	5,640	5,603	8,511	29,579
4	5,724	6,360	8,457	23,930

Tabela 3.10: Alteração de *demands* em MT-OSPF na topologia 30_2

Os resultados demonstram que existe uma maior tolerância a aumentos de *demands* em redes com multi-topologia. Comparando os valores da função de congestão Φ^* para 1 e 4 topologias, observa-se no caso de uma aumento de 60% sobre as *demands* D0.3, uma redução do nível de congestão de 8.327 para 5.724. No caso do aumento de 20% sobre as *demands* D0.5, com uma única topologia o valor de congestão é 58.351, descendo para 8,457 com 4 topologias. Ambos os exemplos evidenciam um ganho considerável no nível de congestão da rede em cenários em que se registam aumentos consideráveis nos requisitos de tráfego.

O recurso à otimização das configurações de pesos OSPF em muti-topologias apresenta-se como uma solução promissora para reduzir os níveis de congestão de uma rede, sobretudo pela tolerância evidenciada em cenários em que se registam alterações de *demands*, dispensando assim a necessidade de efetuar alterações aos pesos OSPF.

3.7 Otimização para Duas Matrizes de Tráfego

Para uma determinada rede, o problema tradicional de encaminhamento lida com a seleção de caminhos para transferir um determinado conjunto de *demands* entre origens e destinos. Nesta definição geral, presume-se que o volume de tráfego entre cada par de origem e destino são já conhecidos. No entanto, o aumento e variedade de serviços no mundo empresarial contemporâneo, que se traduz em variações de tráfego, dificultam o planeamento de redes confiáveis, utilizando uma única matriz de *demands*.

A título de exemplo, numa base diária, existem grandes diferenças estruturais entre o tráfego diurno e o tráfego noturno. No entanto, este modelo, para diferentes dias, mantem-se e é relativamente semelhante, isto é, o tráfego sofre mudanças periódicas previsíveis numa base diária [14, 8]. A abordagem natural para lidar com as mudanças diárias consistiria em executar um novo processo de otimização para adaptar os pesos OSPF às mudanças. No entanto, tal não é uma boa prática. Torna-se assim necessário procurar uma configuração de pesos que promova um bom nível de congestão para todas as mudanças periódicas típicas que se verificam durante um determinado intervalo de tempo, por exemplo um dia. Uma solução alternativa, poderia passar por super estimar as necessidades de tráfego, de forma a contemplar as necessidades diurnas e noturnas. Todavia, este tipo de solução, levaria ao desperdício de recursos de rede.

Para lidar com este tipo de mudanças periódicas, sugerimos otimizar a configuração de pesos para duas matrizes de tráfego representativas, como uma matriz diurna e uma matriz noturna. Neste contexto torna-se necessário redefinir o problema de otimização.

Para uma rede, representada por um grafo $G = (N,A)$, e dadas duas matrizes de tráfego D_1 e D_2, pretende-se encontrar um conjunto de pesos w que minimizem

simultaneamente a funções $\Phi_1^*(w)$ e $\Phi_2^*(w)$, onde $\Phi_1^*(w)$ e $\Phi_2^*(w)$ representam a função $\Phi^*(w)$ considerando respetivamente as *demands* D_1 e D_2. A otimização multi-objetivo é conseguida, utilizando EAs, com recurso a uma função agregadora, definida na Equação 3.5.

$$f(w) = \alpha\Phi_1^*(w) + (1-\alpha)\,\Phi_2^*(w)\,,\ \alpha \in [0;\ 1] \tag{3.5}$$

Para avaliar o desempenho do EA para duas matrizes de *demands* foram efetuadas várias simulações. Na Tabela 3.11 encontram-se os resultados da otimização da topologia 30_2 para duas matrizes de tráfego $D0.5_1$, diurno, e $D0.5_2$, noturno. Para matrizes de tráfego menos exigentes, e dado existir uma maior tolerância à variação *demands*, os resultados não são tão significativos.

	Função custo $f(w)$		
	$\alpha = 1$	$\alpha = 0$	$\alpha = 0.5$
Φ_1^* (D0.5$_1$)	2,552	48,841	3,682
Φ_2^* (D0.5$_2$)	53,687	2,614	3,483

Tabela 3.11: Otimização da função $f(w)$ para duas matrizes de tráfego D0.5$_1$ e D0.5$_2$ ($\alpha = 0, 1, 0.5$)

Como o processo de otimização depende das matrizes de tráfego recebidas como *input*, no caso de se verificar uma alteração significativa de *demands*, as configurações de pesos já otimizadas poderão não ser suficientemente boas para as novas necessidades *demands*. Uma otimização para as *demands* $D0.5_1$, com medida de congestão 2.552, revela-se inadequada para as *demands* $D0.5_2$. É possível, todavia, obter uma configuração adequada para as duas matrizes, comprometendo ligeiramente os níveis de congestão. O parâmetro α permite ajustar o nível de compromisso na otimização para as duas matrizes. Um administrador de rede poderá pretender comprometer o tráfego noturno, $D0.5_2$ favorecendo o tráfego diurno, $D0.5_1$. Neste contexto o valor de α deverá traduzir esse compromisso assumindo um valor mais perto de 1. No caso apresentado, α foi configurado com o valor 0.5, dando igual importância ao tráfego diurno e ao tráfego noturno. No caso de $\alpha = 0$, os requisitos de tráfego diurno não serão considerados, e a otimização será feita somente para o tráfego noturno.

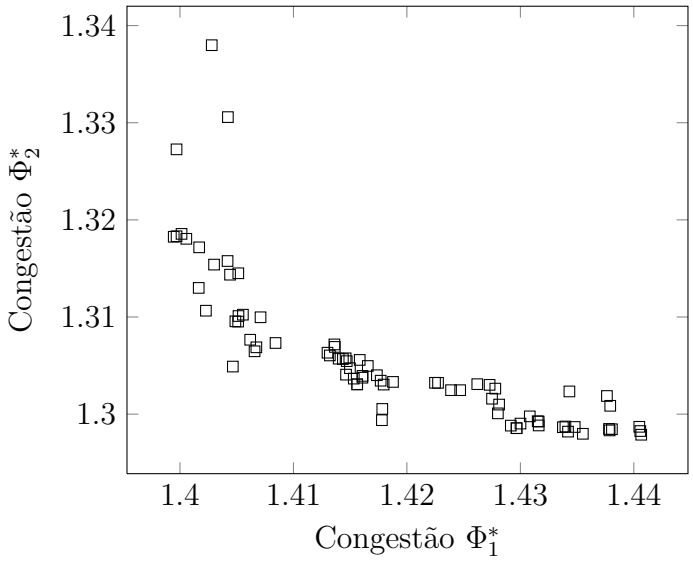


Figura 3.11: Gráfico de dispersão das soluções obtidas com MOEA para duas matrizes D0.3 e $\alpha = 0.5$.

A otimização para duas matrizes de tráfego também foi desenvolvida para MOEA, SPEA2 e NSGAII. Recorrendo à implementação multi-objetivo, um administrador de rede poderá escolher a solução mais adequada dentro de um ótimo de Pareto, onde co-existem vários possíveis *trade-offs* entre os dois objetivos de otimização, como pode ser observado na Figura 3.11 para a otimização com duas matrizes de *demands* D0.3.

3.8 Falha de Link

A otimização de pesos OSPF é aplicada a uma topologia fixa, para um conjunto específico de *demands* de tráfego. Todavia, a falha de um *link* origina uma alteração na topologia e uma subsequente atualização das tabelas de encaminhamento.
Os pesos existentes, otimizados para a topologia original, podem não ser suficientemente bons para a nova topologia com falha de *link*. Após a falha de ligação, o tráfego que fluía por esse *link* passa a ser encaminhado por outras ligações disponíveis. Todavia, e dado que que a rede não foi otimizada para a nova topologia, o mapeamento de tráfego pode tornar-se ineficiente, podendo registar-se congestio-

namento em algumas partes da rede, especialmente em casos de otimização com *demands* de nível elevado.

3.8.1 Função Custo para a Falha de Link com Maior Carga

Para responder à falha do *link* com maior carga média de tráfego, propõe-se um algoritmo evolucionário que otimiza as configurações de pesos OSPF para o cenário de falha. O algoritmo, cujo o pseudo-código é apresentado no Algoritmo 3, para cada solução w, avalia o nível de congestão Φ_n^* da rede sem falha, e o nível de congestão Φ_{n-1}^* com a falha do link com maior carga. A avaliação de uma solução que suporte a falha do *link* com maior carga é efetuada utilizando uma função agregadora definida na Equação 3.6.

$$f(w) = \alpha\Phi_n^*(w) + (1-\alpha)\,\Phi_{n-1}^*(w)\,,\ \alpha \in [0;\ 1] \tag{3.6}$$

A falha do *link* com maior carga numa rede configura um dos piores cenários na falha de um único *link*. Em [38] é proposta uma solução semelhante que difere essencialmente no fator de ponderação. Enquanto que nesta proposta o fator de ponderação α pode assumir qualquer valor no intervalo [0, 1], em [38] o fator de ponderação é constante e igual a 0.5.

A variabilidade do fator de ponderação α permite um conjunto mais flexível de configurações por parte do administrador de rede. Quando $\alpha = 1$, é realizada a otimização para uma única topologia, a topologia sem falha de *link*. Em processos de otimização com $\alpha = 0.5$, pretende-se atribuir o mesmo nível de importância à congestão antes e depois da falha de *link* com maior carga. Todavia, e dado que a otimização para falha de *link* poderá comprometer o nível de congestão da rede num estado original, um administrador de rede poderá desejar privilegiar o desempenho da rede no estado normal em detrimento do estado de falha que poderá não ocorrer. Neste contexto, o administrador de rede poderá configurar α com um valor compreendido entre 0.5 e 1, que melhor traduza o grau de compromisso desejado.

Algoritmo 3: Pseudocódigo do EA para otimização para falha de link

```
1  t = 0
2  INICIALIZAR P(0)
3  AVALIAR P(0)
4  while !CRITÉRIO DE PARAGEM do
5      t = t + 1
6      SELECIONAR progenitores
7      RECOMBINAR pares de progenitores
8      APLICAR MUTAÇÃO nos descendentes obtidos
9      begin AVALIAR os novos candidatos
10         AVALIAR sem falha de link
11         IDENTIFICAR o link com maior carga
12         SET TO DOWN o link com maior carga
13         AVALIAR com falha de link
14         APLICAR TRADE-OFF entre as avaliações
15     end
16     SELECIONAR indivíduos para a próxima geração
17 end
```

Demands	Sem otimização para falha de link $\alpha = 1$		Com otimização para falha de link $\alpha = 0.5$	
	Antes da Falha	Depois da Falha	Antes da Falha	Depois da Falha
D0.3	1,395	30,429	1,479	1,504
D0.4	1,726	55,589	1,803	1,824
D0.5	3,915	199,636	6,059	5,414

Tabela 3.12: Otimização para falha de link com $\alpha = 1$ e $\alpha = 0.5$

Demands	$\alpha = 0.75$		$\alpha = 0.85$	
	Antes da Falha	Depois da Falha	Antes da Falha	Depois da Falha
D0.3	1,455	1,479	1,450	1,536
D0.4	1,754	1,810	1,807	1,850
D0.5	4,926	6,453	4,933	7,559

Tabela 3.13: Otimização para falha de link com $\alpha = 0.75$ e $\alpha = 0.85$

Nas Tabelas 3.12 e 3.13 são apresentados alguns resultados da aplicação deste algoritmo na topologia 30_2, para matrizes de *demands* $D0.3$, $D0.4$ e $D0.5$, com fatores de ponderação 1 (sem otimização para falha de *link*), 0.5, 0.75 e 0.85. Para cada cenário, foram executados 10 processos de otimização, sendo os resultados apresentados as médias dos valores observados.

Comparando os níveis de congestão sem otimização para falha de *link* ($\alpha = 1$) com os valores obtidos com otimização para a falha do *link* com maior carga média de tráfego (com $\alpha = 0.5$, $\alpha = 0.75$ e $\alpha = 0.85$), constata-se que uma penalização reduzida no nível de congestão da rede no estado normal, traduz grandes ganhos em todos os cenários. No caso particular onde $\alpha = 0.85$, apesar de, na rede sem falha, as penalizações nos níveis de congestão serem muito reduzidas, passando de 1.395 para 1.450 (D0.3), de 1.726 para 1.807 (D0.4), e de 3.915 para 4.933 (D0.5), o ganho nos níveis de congestão em caso de falha são muito significativos, baixando de 30.429 para 1.536 (D0.3), de 55.589 para 1.850 (D0.4), e de 199.636 para 7.559 (D0.5). Estes resultados indiciam que esta poderá ser uma boa solução para a otimização de configurações de pesos OSPF que contemplam a falha do link com maior carga. Será todavia necessário consolidar estes resultados testando outros cenários e topologias.

3.8.2 Função Custo com Restrições de Atraso

É possível ainda introduzir no processo de otimização restrições de *delay*. A função custo γ^*, introduzida em [32] e descrita na secção 2.3, pode ser agregada na função custo 3.6, recorrendo a um novo fator de ponderação $\beta \in [0,1]$, da forma expressa pela Equação 3.7.

$$\begin{aligned} f(w) = \quad & \alpha \quad && (\beta\Phi_n^*(w) + (1-\beta)\,\gamma_n^*(w)) + \\ & (1-\alpha) && \left(\beta\Phi_{n-1}^*(w) + (1-\beta)\,\gamma_{n-1}^*(w)\right) \end{aligned} \tag{3.7}$$

Esta nova função de aptidão permite otimizar simultaneamente a congestão e o atraso para cenários com e sem falha do *link* com maior carga. O parâmetro β quantifica o *trade-off* entre os objetivos de otimização congestão Φ^* e atraso γ^*.

Na Tabela 3.14 são apresentados os valores de congestão e atraso da otimização com e sem falha de *link* com maior carga, obtidos para a topologia, 30_2, com *demands* D0.3 e matriz de atrasos DR4.0. Com o parâmetros $\beta = 0.5$, os resultados mostram que, apesar de se registar resultados ligeiramente piores para as restrições de atraso, na otimização para a falha do *link* com maior carga, o ganho obtido com a preservação do nível de congestão justifica o recurso a tal processo de otimização de pesos OSPF. Todavia é possível reduzir ligeiramente o valor da medida de atraso γ^* com uma configuração adequada do parâmetro β, como pode ser constatado na Tabela 3.14 com $\beta = 0.15$. O mesmo valor de β foi utilizado no cenário que se segue.

Processo de otimização	Antes da Falha		Depois da Falha	
	Congestão	Atraso	Congestão	Atraso
$\alpha = 1$ e $\beta = 0.5$ (Sem Falha)	1,538	1,423	27.125	1.437
$\alpha = 0.5$ e $\beta = 0.5$ (Com Falha)	1,625	1,550	1,737	4,576
$\alpha = 1$ e $\beta = 0.15$ (Sem Falha)	1,628	1,384	67,905	1,393
$\alpha = 0.5$ e $\beta = 0.15$ (Com Falha)	1,747	1,604	1,766	4,271

Tabela 3.14: Otimização de congestão e atraso para um cenário falha de link (D0.3 e DR4.0) com $\beta = 0.5$ e $\beta = 0.15$

Processo de Otimização	Antes da Falha		Depois da Falha	
	Congestão	Atraso	Congestão	Atraso
$\alpha = 1$ e $\beta = 0.15$ (Sem Falha)	1,787	4,587	81,725	13,748
$\alpha = 0.5$ e $\beta = 0.15$ (Com Falha)	1,941	6,317	1,895	10,048

Tabela 3.15: Otimização de congestão e atraso para um cenário falha de link (D0.3 e DR3.0) com $\beta = 0.15$

Na Tabela 3.15, para a mesma topologia, são apresentados os resultados para duas configurações de otimização para um cenário com *demands* D0.3 e restrições de *delay* DR3.0. No cenário em que não é considerada a otimização para a falha do *link* com maior carga, depois da falha, o valor de congestão Φ^* aumenta drasticamente de 1.787 para 81.725, enquanto que no cenário em que é considerada a otimização para a falha do *link* com maior carga, depois da falha, o valor de congestão Φ^* mantem-se sensivelmente constante. Apesar de, nesse último caso,

as restrições de atraso serem ligeiramente penalizadas, passando de 6.317 para 10.048, a preservação do nível de congestão confere a este método virtudes que poderão ser exploradas por administradores de redes.

3.9 Sumário

Neste capítulo foram analisados os efeitos, na convergência dos processos de otimização de configurações de pesos OSPF, da hibridização das configurações de populações iniciais com configurações de pesos OSPF usadas tradicionalmente, como o InvCapOSPF, e com soluções provenientes de otimizações anteriores. Foram ainda propostos e analisados trés métodos de otimização de configurações de pesos OSPF. A otimização de congestão em MT-OSPF, revelou ser uma solução promissora para reduzir o nível de congestão de uma rede, introduzindo simultaneamente uma maior tolerância a variações nos requisitos de tráfego. A otimização para duas matrizes de tráfego permite apresentar possíveis soluções de configurações de pesos OFPF para cenários em que existam variações periódicas de requisitos de trafego modelados por duas matrizes de *demands*, como por exemplo, um cenário diurno e noturno. A otimização para a falha de *link* confere a uma rede uma maior tolerância à falha do *link* com maior carga média de tráfego, procurando preservar valores aceitáveis de congestão antes e depois desta alteração na topologia.

Capítulo 4

Melhorias à *Framework* NetOpt

Dando seguimento ao desenvolvimento dos métodos descritos no capítulo anterior, neste capítulo serão sumariadas as principais funcionalidades implementadas na framework NetOpt, bem como as melhorias realizadas à interface gráfica disponibilizada ao utilizador.

4.1 Principais Alterações da Camada Lógica

O NetOpt, descrito na secção 2.3, foi utilizado como suporte para os métodos descritos no capítulo anterior e para a recolha dos resultados apresentados. Para incorporar todas as novas funcionalidades descrita no capítulo 3, foi necessário proceder a algumas alterações e adaptações na *framework* original. Para o efeito foi utilizada a versão do código fonte disponibilizada em `http://darwin.di.uminho.pt/netopt/`. Todo o código foi integralmente analisado, bem como a biblioteca JEColi. Esta tarefa foi essencial para a integração dos novos métodos propostos.

4.1.1 Algoritmo de Dijkstra com ECMP

A implementação do algoritmo de Dijkstra existente na versão original do NetOpt respondia ao *single shortest path problem*. O balanceamento do tráfego era efetuado analisando a existência de caminhos de custo igual no processo de distribuição de tráfego. Todavia, para implementar a medida de comparação de caminhos era

necessário manter uma lista de caminhos mais curtos com o mesmo custo. Neste contexto, foram efetuadas alterações ao algoritmo do caminho mais curto de modo a contemplar caminhos com o mesmo custo. O Algoritmo 4 reflete as alterações efetuadas.

A principal diferença entre os dois algoritmos reside na definição de predecessor. Enquanto que no algoritmo *single shortest path* o predecessor é um único nó, no algoritmo para múltiplos caminhos mais curtos, para cada nó, é mantida uma lista de predecessores (ver passo 16). Os distintos caminhos de igual custo são posteriormente construídos a partir dessas listas de predecessores.

Algoritmo 4: Algoritmo de Dijkstra com caminhos de custo igual

```
1  d[s] = 0
2  p(s) = ListaVazia
3  p(s).adiciona(s)
4  foreach v ∈ V, v ≠ s do
5      d[v] = INF
6      p(v) = ListaVazia
7  end
8  S = NULL
9  Q = V
10 while Q ≠ ∅ do
11     u = EXTRACT_MIN(Q)
12     S = S ∪ {u}
13     foreach v adjacente a u do
14         if d[v] ≥ d[u] + w[u,v] then
15             d[v] = d[u] + w[u,v]
16             p(v).adiciona(u)
17             DECREASE_KEY[v,Q]
18         end
19     end
20 end
```

4.1.2 Populações

Após a execução de um processo de otimização, as populações da última iteração do algoritmos evolucionário não eram guardadas. Um utilizador tinha acesso a

uma única solução, construída a partir do indivíduo da população final com melhor valor de aptidão, sendo os restantes indivíduos eliminados. Para avaliar o efeito da inclusão de soluções provenientes de processos de otimização anteriormente efetuados, foi necessário definir uma população como novo tipo de dados. A conservação destas populações foi conseguida estendendo a interface *IAlgorithm* da biblioteca JEColi, bem como as respetivas implementações.

Para além de permitir a inclusão de indivíduos provenientes de populações de otimizações anteriores, tornou-se possível converter qualquer indivíduo dessas populações em soluções de configuração de pesos OSPF. Esta nova funcionalidade é particularmente interessante quando as populações resultam da execução dos MOEA, ou seja, otimização com NSGA-II e SPEA2. Cada indivíduo nessas populações representa um *trade-off* entre os objetivos de otimização. Um administrador de rede pode assim escolher a configuração de pesos que lhe é mais adequada de entre todos os indivíduos da população.

4.1.3 Estado de Links

A versão original do NetOpt não contemplava qualquer estado de *link*, à exceção da sua existência. Para modelar cenários de falha de *link* foi necessário introduzir um novo estado "*Link down*". Todo o processo de otimização e simulação teve de ser adaptado para contemplar essa nova definição de estado. Um utilizador pode optar assim por ativar ou desativar um *link* diretamente na representação gráfica da topologia ou bem na tabela de definição de *links*,.

Como cada processo de otimização corre num único *thread* que utiliza a definição de topologia comum a toda a *framework*, foi necessário tornar o processo de otimização *thread safe*. Assim, para todas as classes de definição de topologia e grafos foram implementados métodos de *deep copy* e *clone*, para que cada instância do processo de otimização pudesse ser executada de forma segura e independente.

Apesar desta nova funcionalidade ter sido utilizada somente no contexto da otimização de pesos OSPF para a falha de *link* com maior carga (ver secção 3.8), existem inúmeras outras aplicações, como por exemplo definir multi-topologias com seleções diferentes de *links*. Estas aplicações poderão ser exploradas em trabalhos futuros.

4.1.4 Edição da Topologia

Existem situações na quais as topologias de rede precisam de ser alteradas para refletir a adição ou remoção de *links* na topologia real. Além disso, os *links* têm caraterísticas que podem requerer atualizações, como é o caso da capacidade e do atraso de propagação. Para responder a essas necessidades foram implementadas opções para adicionar, editar e remover *links* da topologia. Em caso de adição ou remoção de *links*, as configurações de pesos já existentes para essa topologia passam a ser inadequadas. Assim, sempre que um *link* é removido da topologia, os pesos associados a esse *link* também são removidos das configurações de pesos e dos indivíduos das várias populações. Quando um novo *link* é adicionado, pesos unitários são adicionados às configurações de pesos e indivíduos de populações.
A remoção de *links* da topologia pode fazer com que o grafo, que modela essa mesma topologia, se torne desconexo. Para evitar a ocorrência dessa situação, sempre que um *link* é removido, a conectividade da topologia é avaliada. Caso deixe de ser possível alcançar algum nó, ou seja, se não existir um caminho entre todos os nós da topologia, o utilizador é informado. O utilizador poderá também, em qualquer momento, testar a conectividade da topologia. A análise de conectividade do grafo que modela a topologia é efetuada por um algoritmo de busca em profundidade.

4.2 Novas Funcionalidades

Os métodos de otimização descritos no capítulo 3 bem como as alterações descritas na secção anterior traduziram-se na integração de um conjunto de novas funcionalidades na *framework* NetOpt.

4.2.1 Opções de Otimização

Do trabalho de investigação levado a cabo, resultaram um conjunto de novas propostas de métodos de otimização de pesos OSPF bem como distintas configurações para *seeding* da população inicial dos EAs.

- **Otimização para *demands* e *delay***

À funcionalidade de otimização da congestão com restrições de atraso, já existente na *framework*, foram adicionadas opções de configuração que contemplam a inclusão de pesos InvCapOSPF, UnitOSPF e L2OSPF na população inicial (ver Figura 4.1).

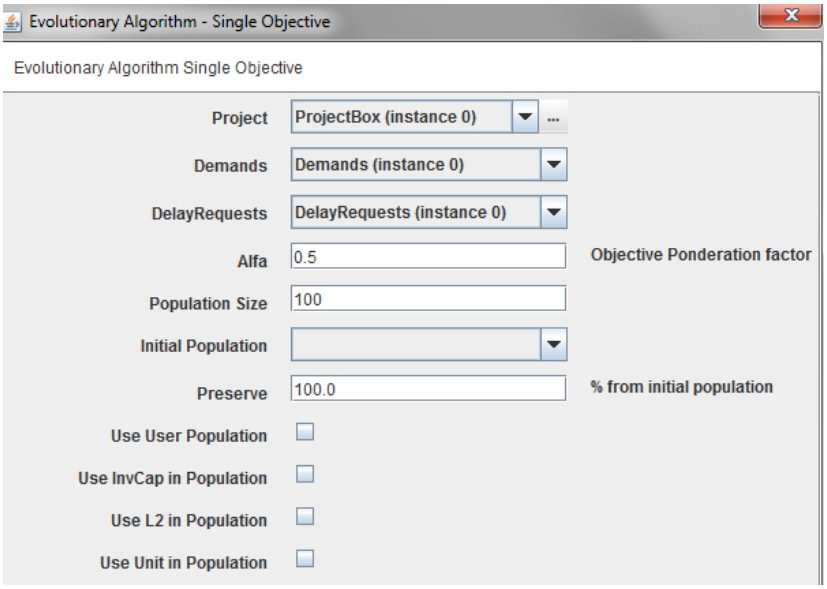


Figura 4.1: GUI com opções de *seeding* da população inicial.

As populações finais resultantes da execução de EAs *single objective* bem como de MOEAs (NSGA-II e SPEA2) podem ser utilizadas indistintamente para *seeding* da população inicial de uma otimização *single objective*.

- **Otimização para falha de *link* com maior carga**
  A otimização de configurações de pesos OSPF para a falha de *link* com maior carga foi implementada unicamente com EA *single objective* recorrendo à função agregadora descrita na secção 3.8.2. Na Figura 4.2 é apresentada a interface disponibilizada ao utilizador.

  A configurações por defeito não incluem a otimização dos atrasos. Para que esse objetivo seja considerado, um utilizador deverá configurar o parâmetro de *trade-off* β com um valor não negativo inferior a 1. O parâmetro α, traduz o compromisso entre a otimização dos objetivos para a topologia sem e com falha de *link* com maior carga.

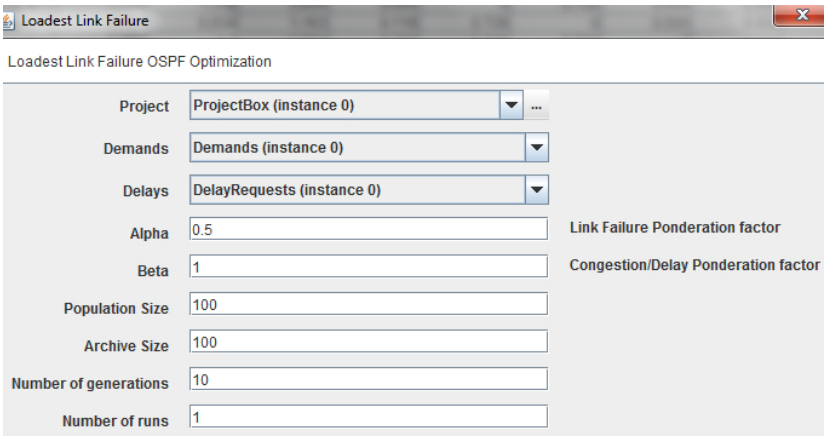


Figura 4.2: GUI da otimização para falha de *link*

- **Otimização para duas matrizes de tráfego**

  A otimização para duas matrizes de necessidades de tráfego permite obter configurações de pesos OSPF otimizadas para dois momentos distintos, tais como, dia e noite ou semana e fim de semana. Quando o processo de otimização é efetuado com recurso a um EA *single objective*, um fator de ponderação α, que assume valores entre 0 e 1, permite definir um compromisso entre os níveis de congestão espetáveis para os dois momentos. Estão também disponíveis implementações que utilizam métodos MOEA, o NSGA-II e o SPEA2, com as vantagens anteriormente apresentadas.

- **Otimização MT-OSPF**

  No contexto da otimização de configurações de pesos MT-OSPF, são disponibilizadas duas opções:

  – Otimização para duas topologias lógicas - O utilizador pode selecionar duas matrizes de tráfego, uma para cada topologia lógica.

  – Otimização para n topologias lógicas - o utilizador define o número de topologias lógicas e a matriz de necessidades de tráfego total. As necessidades de tráfego são distribuídas equitativamente pelas n topologias.

4.2.2 Opções de Análise de Soluções

As tomadas de decisão efetuadas pelos administradores de rede apoiam-se num conjunto de dados e informações existentes. Para dar um melhor suporte à tomada de decisões foram desenvolvidas interfaces que disponibilizam um conjunto variado de informações.

- **Distribuição de carga**

  Duas otimizações de configuração de pesos OSPF, que tenham um mesmo valor da medida de congestão Φ^* (ver secção 2.1.5), podem ter distribuições de carga, nos vários *links*, diferentes. Por exemplo, numa das configurações, as cargas podem ter uma distribuição simétrica em torno de uma taxa de ocupação de 2/3 da capacidade dos *links*, enquanto noutra, a distribuição poderá ser assimétrica com moda em 1/3 da capacidade dos *links*. Apesar dos níveis globais de congestão serem semelhantes, na segunda configuração existirão certamente *links* para os quais o fluxo de tráfego ultrapassa as respetivas capacidades. Nesse caso, será registada uma perda de pacotes muito superior à observada na configuração de distribuição de carga simétrica.

  Para possibilitar uma análise da distribuição da carga dos diferentes *links* da topologia, é disponibilizada uma interface gráfica de análise, como exemplifica a Figura 4.3.

  A percentagem de ocupação prevista para cada *link* é apresentada numa tabela que contém nas linhas a origem dos *links*, e nas colunas o destino. Um gráfico de barras da distribuição de carga agiliza a análise dos valores apresentados.

- **Comparação de pesos e caminhos mais curtos**

  Para comparar as mudanças nos caminhos mais curtos entre os vários nós da topologia, resultantes de alterações na configuração de pesos OSPF, foi implementada uma interface gráfica que apresenta, para par de nós de origem e destino, o valor da medida APC introduzida na secção 3.3.2. Os valores dessa medida são agregados numa tabela como mostra a Figura 4.4a. As células de cor verde, que têm o valor 1, identificam os caminhos nos quais

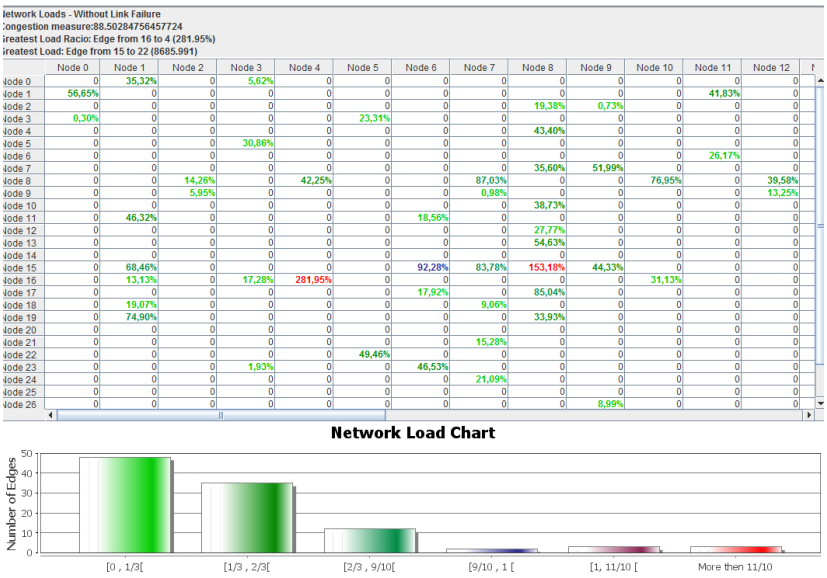


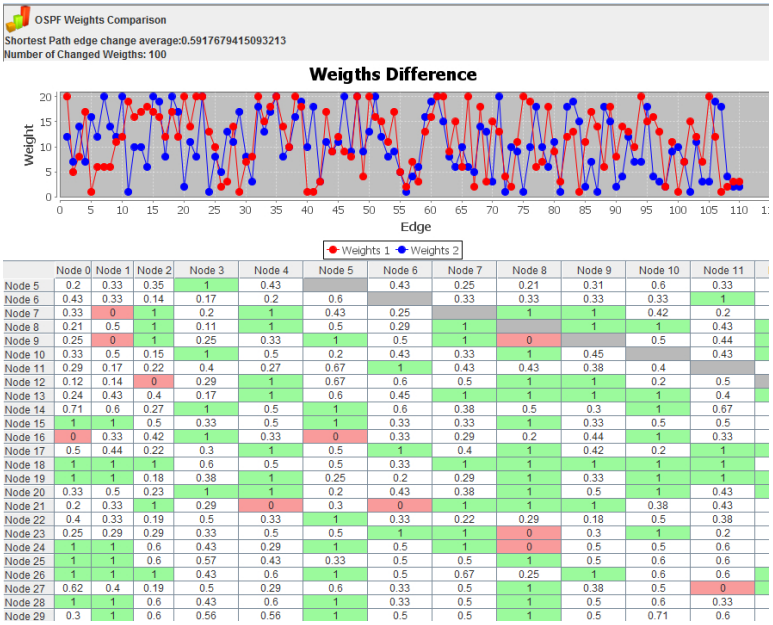
Figura 4.3: Distribuição assimétrica da carga dos *links*

não houve alteração, enquanto as células de cor vermelha, com o valor 0, identificam os caminhos mais curtos que foram totalmente alterados. As restantes células, com valores superiores a 1 e inferiores a 0, identificam alterações parciais em caminhos mais curtos.

Também é possível visualizar na representação gráfica da topologia essas alterações de caminhos mais curtos. Na Figura 4.4b são comparados os caminhos mais curtos de um nó "4" para um nó "5" resultantes de duas configurações de pesos. Os *links* apresentados com cor verde, são comuns nos caminhos mais curtos que resultam das duas configurações de pesos OSPF, enquanto os *links* de cor azul e cor vermelha são, respetivamente, os *links* não comuns da primeira configuração de pesos e da segunda configuração de pesos.

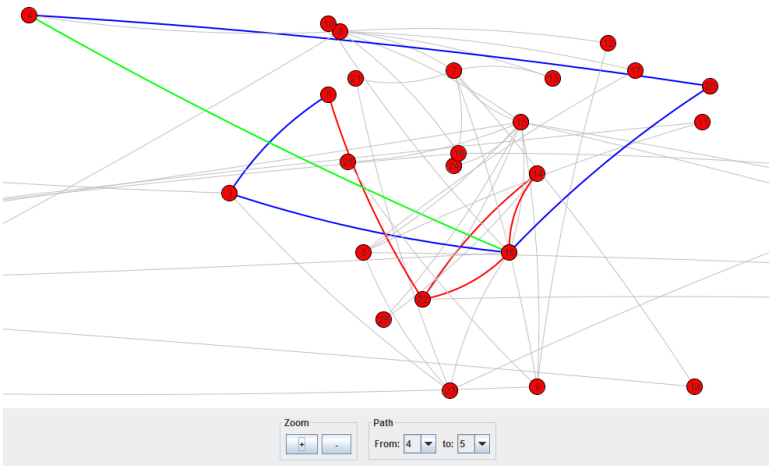
- **Distribuição do tráfego por ECMP**

  Uma outra funcionalidade desenvolvida permite também analisar a distribuição de tráfego por caminhos com o mesmo custo (ECMP). Como mostra



	Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9	Node 10	Node 11
Node 5	0.2	0.33	0.35	1	0.43		0.43	0.25	0.21	0.31	0.6	0.33
Node 6	0.43	0.33	0.14	0.17	0.2	0.6		0.33	0.33	0.33	0.33	1
Node 7	0.33	0	1	0.2	1	0.43	0.25		1	1	0.42	0.2
Node 8	0.21	0.5	1	0.11	1	0.5	0.29	1		1	1	0.43
Node 9	0.25	0	1	0.25	0.33	1	0.5	1	0		0.5	0.44
Node 10	0.33	0.5	0.15	1	0.5	0.2	0.43	0.33	1	0.45		0.43
Node 11	0.29	0.17	0.22	0.4	0.27	0.67	1	0.43	0.43	0.38	0.4	
Node 12	0.12	0.14	0	0.29	1	0.67	0.6	0.5	1	1	0.2	0.5
Node 13	0.24	0.43	0.4	0.17	1	0.6	0.45	1	1	1	1	0.4
Node 14	0.71	0.6	0.27	1	0.5	1	0.6	0.38	0.5	0.3	1	0.67
Node 15	1	1	0.5	0.33	0.5	1	0.33	0.33	1	0.33	0.5	0.5
Node 16	0	0.33	0.42	1	0.33	0	0.33	0.29	0.2	0.44	1	0.33
Node 17	0.5	0.44	0.22	0.3	1	0.5	1	0.4	1	0.42	0.2	1
Node 18	1	1	1	0.6	0.5	0.5	0.33	1	1	1	1	1
Node 19	1	1	0.18	0.38	1	0.25	0.2	0.29	1	0.33	1	1
Node 20	0.33	0.5	0.23	1	1	0.2	0.43	0.38	1	0.5	1	0.43
Node 21	0.2	0.33	1	0.29	0	0.3	0	1	1	1	0.38	0.43
Node 22	0.4	0.33	0.19	0.5	0.33	1	0.33	0.22	0.29	0.18	0.5	0.38
Node 23	0.25	0.29	0.29	0.33	0.5	0.5	1	1	0	0.3	1	0.2
Node 24	1	1	0.6	0.43	0.29	1	0.5	1	0	0.5	0.5	0.6
Node 25	1	1	0.6	0.57	0.43	0.33	0.5	0.5	1	0.5	0.6	0.6
Node 26	1	1	1	0.43	0.6	1	0.5	0.67	0.25	1	0.6	0.6
Node 27	0.62	0.4	0.19	0.5	0.29	0.6	0.33	0.5	1	0.38	0.5	0
Node 28	1	1	0.6	0.43	0.6	1	0.33	0.5	1	0.5	0.6	0.33
Node 29	0.3	1	0.6	0.56	0.56	1	0.5	0.5	1	0.5	0.71	0.6

(a) Tabela APC



(b) Comparação na topologia

Figura 4.4: Comparação de pesos e caminhos mais curtos

a Figura 4.5, para cada *link* no caminho mais curto, é identificada a percentagem de tráfego, com origem num nó "0" e destino num nó "12", que flui por cada *link* nos caminhos mais curtos.

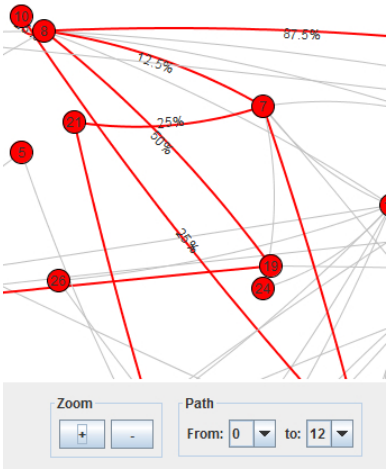


Figura 4.5: Distribuição do tráfego por ECMP

4.2.3 Outras Opções

O trabalho desenvolvido requereu a introdução de novas funcionalidade de menor importância. Um conjunto dessas funcionalidades, que efetuam operações sobre matrizes de tráfego, são:

- Adição de matrizes de tráfego;
- Divisão de uma matriz de tráfego por um valor escalar;
- Alteração de uma matriz de tráfego através da multiplicação de um valor escalar aleatório num dado intervalo;
- *Overload* de um *link*, ou seja, às necessidades de tráfego para um dado *link* são adicionadas novas necessidades de tráfego iguais à sua capacidade.

Foi também introduzida uma opção que permite obter o valor da medida de congestão Φ^* para MT-OSPF, dada uma matriz de necessidades totais de tráfego e configurações de pesos OSPF para cada uma das topologias lógica.

4.3 Sumário

Neste capítulo foram sumariadas algumas das contribuições introduzidas na *framework* NetOpt, de modo a providenciar a administradores de rede uma ferramenta de apoio à configuração de pesos OSPF. Neste contexto, além da incorporação de todos os métodos apresentados no capítulo 3, foram salientadas funcionalidades adicionais introduzidas na camada lógica e na camada de interface da *framework* NetOpt.

Capítulo 5

Conclusões

5.1 Resumo do Trabalho Desenvolvido

Com este trabalho procurou-se responder a problemáticas em domínios OSPF, passivos de serem abordados por otimização das configurações de pesos, que resultam de alterações de condições na rede, em particular, a alterações nas necessidades de tráfego e à falha do *link* com maior carga. Em particular procurou-se desenvolver e analisar métodos que, para além de proporcionarem configurações de pesos OSPF que otimizem os níveis de congestão, fossem capazes de dotar a rede de uma maior tolerância a essas alterações. Neste sentido, após um período de estudo da arte, que incluiu a análise do código fonte de uma *framework* já existente, NetOpt, que recorre a algoritmos evolucionários, foram delineadas estratégias para procurar alcançar as metas definidas.

Numa primeira etapa, e dando continuidade ao desenvolvimento da *framework*, procurou-se reduzir o tempo de convergência dos algoritmos já desenvolvidos recorrendo à hibridação das configurações de populações iniciais.

Numa segunda etapa, foram estudados e propostos novos métodos que procurassem dotar a rede de uma maior tolerância a alterações de *demands* e à falha de *links*. Cada um desses métodos foi implementado na *framework*. O métodos foram posteriormente testados em diversos cenários e configurações. Para permitir a análise dos resultados obtidos, foram desenvolvidas funcionalidades adicionais que foram incluídas no NetOpt. Dos vários métodos estudados, três apresentaram

resultados interessantes, tendo assim sido incluídos e enfatizados nesta dissertação. A falta de recursos computacionais, bem como as limitações de tempo, condicionaram o número e diversidade de cenários explorados. Assim, e recorrendo a um *cluster* computacional instalado na Universidade do Minho, serão realizadas experiências para validar os resultados obtidos em diferentes cenários.
Todo o trabalho desenvolvido culminou num aprimoramento da *framework* NetOpt que passa assim a disponibilizar um conjunto mais alargado de opções de configuração, podendo servir como ferramenta de apoio e análise de soluções aos administradores de redes.

5.2 Principais Contribuições

As principais contribuições deste trabalho são três propostas de métodos de Engenharia de Tráfego que, para além de permitir otimizar as configurações de pesos OSPF, conferem à rede uma maior tolerância a alterações de *demands* de tráfego e falha de *links*.

- **Optimização de configurações de pesos para MT-OSPF .** Este método apresenta-se como uma solução promissora para reduzir os níveis de congestão de uma rede, mas sobretudo pela resiliência evidenciada em cenários em que se registam alterações de *demands*, dispensando assim a necessidade de efetuar alterações aos pesos OSPF.

- **Otimização de configurações de pesos OSPF para duas matrizes de tráfego.** Oferece uma solução de configuração de pesos OSPF otimizados para a congestão, aplicável a cenários nos quais existe uma variação periódica entre dois conjuntos de *demands*, como por exemplo, um cenário noturno e diurnos, ou semana e fim de semana.

- **Otimização de configurações de pesos OSPF para a falha de *link* com maior carga.** Este método permite a otimização de pesos OSPF otimizados para a congestão, respondendo simultaneamente a restrições de atraso, de modo a que, em caso de falha do *link* com maior carga, os níveis de congestão sejam mantidos num nível aceitável.

Os trabalhos desenvolvidos incluem ainda uma análise da variação da congestão de uma rede, com configuração de pesos OSPF, em cenários de aumento dos níveis de *demands*. Este trabalho também inclui um estudo que mostra que a inclusão de pesos InvCapOSPF em populações iniciais de EAs, para a otimização de configurações de pesos OSPF, poderá contribuir para melhorar a convergência do EA. Por fim, todo o trabalho desenvolvimento resultou no aprimoramento da *framework* NetOpt, ferramenta de software livre disponível em `http://darwin.di.uminho.pt/netopt/`, com um novo conjunto alargado de opções, quer a nível da sua lógica interna, quer a nível da sua interface.

5.3 Trabalho Futuro

Cada um dos três métodos propostos poderão ser melhorados. Em particular, o método de otimização de configurações de pesos MT-OSPF foi analisado com um balanceamento equitativo de tráfego entre as várias topologias. O mesmo modelo poderá ser utilizado para explorar vantagens no balanceamento com rácios distintos. Como o mapeamento do tráfego pelas várias topologias é baseada em fluxos, e não em classes de serviço, como acontece no encaminhamento TOS-OSPF, novas restrições de QoS poderão ser incluídas no modelo MT-OSPF. Também poderão ser exploradas configurações de topologias lógicas que não utilizem a totalidade dos *links* da topologia física. Este modelo de encaminhamento, com otimização de configurações de pesos OSPF, demonstrou-se promissor sobretudo pela tolerância a alterações que induz na rede. As restrições de atraso de propagação poderão ser integradas no processo de otimização para duas matrizes de *demands*. Serão ainda efetuados um conjunto mais alargado de testes, que incluam casos reais, para validar os resultados obtidos. Para tornar a *framework* ainda mais atrativa para administradores de redes, poderão ser incluídos novos parâmetros de configuração como, por exemplo, configurações de endereçamento IP.

Bibliografia

[1] Jung java universal network/graph framework. http://jung.sourceforge.net/.

[2] Cisco visual networking index: Forecast and methodology, 2011–2016. White Paper, May 2012.

[3] D. Awduche, A. Chiu, A. Elwalid, I. Widjaja, and X. Xiao. Overview and principles of internet traffic engineering. May 2002. Status: INFORMATIONAL.

[4] A. Barabási and R. Albert. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[5] L. Buriol, M. Resende, C. Ribeiro, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in ospf/is-is routing. *Journal of Combinatorial Optimization*, 6:299–333, 2003.

[6] Z. Cao, Z. Wang, and E.W. Zegura. Performance of hashing-based schemes for internet load balancing. In *Proceedings of IEEE INFOCOM 2000*, volume 1, pages 332–341, 2000.

[7] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[8] P. Cortez, M. Rio, M. Rocha, and P. Sousa. Multiscale internet traffic forecasting using neural networks and time series methods. *Expert Systems*, 29(2):143–155, May 2012.

[9] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[10] V. Durairaj and P. Kalla. Enhanced interior gateway routing protocol, cisco white paper. In *Proceedings of Theory and Applications of Satisfiability Testing: 8th International Conference, SAT 2005*, pages 415–422, 2005.

[11] A. Eiben and E. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, 2003.

[12] M. Ericsson, M. Resende, and P. Pardalos. A Genetic Algorithm for the Weight Setting Problem in OSPF Routing. *Journal of Combinatorial Optimization*, 6:299–333, 2002.

[13] P. Evangelista, P. Maia, and M. Rocha. Implementing metaheuristic optimization algorithms with jecoli. In *Proceedings of ISDA*, pages 505–510. IEEE Computer Society, 2009.

[14] A. Feldmann, A. Greenberg, C. Lund, N. Reingold, J. Rexford, and F. True. Deriving traffic demands for operational ip networks: methodology and experience. *IEEE/ACM Transactions on Networking*, 9(3):265–280, June 2001.

[15] B. Fortz. Internet traffic engineering by optimizing ospf weights. In *Proceedings of IEEE INFOCOM*, pages 519–528, 2000.

[16] B. Fortz and M. Thorup. Optimizing ospf/is-is weights in a changing world. *IEEE Journal on Selected Areas in Communications*, 20(4):756–767, 2002.

[17] M. Ghazala and A. El-Sayed. Open shortest path first weight setting (ospfws) solving algorithms comparison and new method for updating weights. *International Journal of Computer Science and Network Security*, 9(5):191–197, 2009.

[18] E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1st edition, 1989.

[19] C. Hopps. Analysis of an equal-cost multi-path algorithm. RFC 2992 (Informational), November 2000.

[20] G. Malkin. RIP Version 2. RFC 2453 (Standard), November 1998. Updated by RFC 4822.

[21] A. Medina, A. Lakhina, I. Matta, and J. Byers. Brite: Universal topology generation from a user's perspective. Technical report, Boston, MA, USA, 2001.

[22] A. Medina, N. Taft, K. Salamatian, S. Bhattacharyya, and C. Diot. Traffic matrix estimation: existing techniques and new directions. In *Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '02, pages 161–174, New York, NY, USA, 2002. ACM.

[23] G. Mendel. Versuche über pflanzenhybriden. *Verhandlungen des Naturforschenden Vereines in Brünn*, 4:3—47, 1866.

[24] J. Moy. OSPF Version 2. RFC 2328 (Standard), April 1998. Updated by RFC 5709.

[25] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the differentiated services field (ds field) in the ipv4 and ipv6 headers. RFC 2474 (Standards Track), December 1998.

[26] D. Oran. OSI IS-IS Intra-domain Routing Protocol. Technical report, IETF, February 1990.

[27] P. Psenak, S. Mirtorabi, A. Roy, L. Nguyen, and P. Pillay-Esnault. Multi-Topology (MT) Routing in OSPF. RFC 4915 (Proposed Standard), June 2007.

[28] B. Quoitin and S. Tandel. A bgp solver for hot-potato routing sensitivity analysis. In *Proceedings of EUNICE 2005: Networks and Applications Towards a Ubiquitously Connected World*, volume 196, pages 75–90. IFIP International Federation for Information Processing, 2006.

[29] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). RFC 4271 (Draft Standard), January 2006. Updated by RFC 6286.

[30] M. Rocha and J. Neves. Computação Genética e Evolucionária. Texto de apoio às aulas de Bioinformática e Computação Natural, 2004.

[31] M. Rocha, T. Sá, P. Sousa, and P Cortez. Multiobjective evolutionary algorithms for intradomain routing optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 2272–2279, June 2011.

[32] M. Rocha, P. Sousa, P. Cortez, and M. Rio. Quality of Service Constrained Routing Optimization Using Evolutionary Computation. *Applied Soft Computing*, 11(1):356–364, 2011.

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

[34] S. Rost and H. Balakrishnan. Rate-aware splitting of aggregate traffic. Technical report, MIT, 2003.

[35] P. Sousa, P. Cortez, M. Rio, and M. Rocha. Traffic engineering approaches using multicriteria optimization techniques. In *Proceedings of Wired/Wireless Internet Comunications*, volume 6649/2011, pages 104–115, 2011.

[36] P. Sousa, M. Rocha, M. Rio, and P. Cortez. Efficient ospf weight allocation for intra-domain qos optimization. In *Proceedings of IPOM*, volume 4268 of *Lecture Notes in Computer Science*, pages 37–48. Springer, 2006.

[37] P. Sousa, M. Rocha, M. Rio, and P. Cortez. Class-based ospf traffic engineering inspired on evolutionary computation. In *Proceedings of the 5th international conference on Wired/Wireless Internet Communications*, WWIC '07, pages 141–152, Berlin, Heidelberg, 2007. Springer-Verlag.

[38] M. Sqalli, S. Sait, and S. Asadullah. Ospf weight setting optimization for single link failures. *International Journal of Computer Networks & Communications*, 3(1):168–183, January 2011.

[39] M. Sqalli, S. Sait, and M. Mohiuddin. An enhanced estimator to multiobjective ospf weight setting problem. In *Proceedings of NOMS*, pages 240–247. IEEE, 2006.

[40] A Sridharan, R Guérin, and C. Diot. Achieving near-optimal traffic engineering solutions for current ospf/is-is networks. In *Proceedings of IEEE/ACM Transactions on Networking*, pages 234–247, 2002.

[41] R. Teixeira, A. Shaikh, T. Griffin, and J. Rexford. Dynamics of hot-potato routing in ip networks. *SIGMETRICS Perform. Eval. Rev.*, 32(1):307–319, June 2004.

[42] D. Thaler and C. Hopps. Multipath issues in unicast and multicast next-hop selection. RFC 2991, 2000.

[43] C Villamizar. Ospf optimized multipath (ospf-omp) draft-ietf-ospf-omp-02.txt, February 1999. INTERNET-DRAFT (work in progress).

[44] X. Wang, S. Wang, and L. Li. Robust traffic engineering using multi-topology routing. In *Proceedings of the 28th IEEE conference on Global telecommunications*, GLOBECOM'09, pages 6345–6350, Piscataway, NJ, USA, 2009. IEEE Press.

[45] Z. Wang. *Internet QoS, Architecture and Mechanisms for Quality of Service.* Morgan-Kaufman Publishers, 2001.

Bibliografia

Apêndice A

Topologia 30_2

Neste apêndice está incluída a definição da topologia 30_2, como uma lista de nós e uma lista de arcos, utilizada nos testes realizados neste trabalho, e respeitante à qual foram obtidos grande parte dos resultados apresentados.

A.1 Tabela de Nós da Topologia

ID	xPos	yPos	inDegree	outDegree	Asid	Type
0	93.0	34.0	2	2	-1	RT_NODE
1	44.0	72.0	6	6	-1	RT_NODE
2	137.0	97.0	2	2	-1	RT_NODE
3	275.0	48.0	4	4	-1	RT_NODE
4	224.0	3.0	3	3	-1	RT_NODE
5	300.0	23.0	2	2	-1	RT_NODE
6	309.0	63.0	5	5	-1	RT_NODE
7	332.0	17.0	7	7	-1	RT_NODE
8	303.0	7.0	10	10	-1	RT_NODE
9	353.0	97.0	5	5	-1	RT_NODE
10	300.0	5.0	2	2	-1	RT_NODE
11	395.0	30.0	2	2	-1	RT_NODE
12	371.0	10.0	2	2	-1	RT_NODE
13	357.0	19.0	2	2	-1	RT_NODE

ID	xPos	yPos	inDegree	outDegree	Asid	Type
14	353.0	43.0	3	3	-1	RT_NODE
15	349.0	30.0	13	13	-1	RT_NODE
16	346.0	63.0	9	9	-1	RT_NODE
17	378.0	17.0	2	2	-1	RT_NODE
18	393.0	97.0	2	2	-1	RT_NODE
19	333.0	38.0	3	3	-1	RT_NODE
20	397.0	21.0	2	2	-1	RT_NODE
21	307.0	19.0	2	2	-1	RT_NODE
22	324.0	75.0	5	5	-1	RT_NODE
23	331.0	98.0	7	7	-1	RT_NODE
24	332.0	41.0	2	2	-1	RT_NODE
25	314.0	80.0	2	2	-1	RT_NODE
26	305.0	40.0	2	2	-1	RT_NODE
27	461.0	45.0	2	2	-1	RT_NODE
28	559.0	74.0	2	2	-1	RT_NODE
29	524.0	78.0	2	2	-1	RT_NODE

A.2 Tabela de *links* da Topologia

Id	From	To	Length	Delay	BandWidth	ASFrom	ASTo
0	15	8	51.42956348249516	0.1715505580947442	2759.506465461309	-1	-1
3	16	23	38.07886552931954	0.12701742326459572	5708.234371037585	-1	-1
4	16	15	33.13608305156178	0.11053007561505028	9753.06593201016	-1	-1
5	7	8	30.675723300355934	0.10232319887232097	4741.441385673896	-1	-1
6	7	15	21.400934559032695	0.07138583372578604	4178.701513754696	-1	-1
7	22	16	25.059928172283335	0.08359092266518371	7841.994649738487	-1	-1
8	22	15	51.478150704935004	0.17171262762365758	7985.115728094532	-1	-1
9	9	15	67.11929677819934	0.22388587500156307	3812.70161104778	-1	-1
10	9	7	82.71033792700887	0.27589199034156114	9163.795548646209	-1	-1
11	21	23	82.56512580987206	0.2754076148569157	6163.810472952047	-1	-1
12	21	7	25.079872407968907	0.08365744947449247	10297.299616851118	-1	-1
13	6	15	51.85556864985669	0.17297155837675107	8312.350029763784	-1	-1
14	6	23	41.340053217188775	0.1378955744683503	1207.72099590322	-1	-1
15	1	16	302.1340761979688	1.0078107975550499	8955.260598787096	-1	-1
16	1	15	307.8782226790326	1.026971207791467	8870.437143565587	-1	-1
17	29	15	181.46349495146399	0.6052970650497952	3503.113664269887	-1	-1
18	29	22	200.02249873451737	0.667203238096528	9729.390460840801	-1	-1
19	14	16	21.18962010041709	0.07068096456388204	9705.341620530258	-1	-1
20	14	22	43.18564576337837	0.14405180854609212	5993.75119081819	-1	-1
21	24	15	20.248456731316587	0.06754158148740548	10977.003148102433	-1	-1
22	24	7	24.0	0.08005538284755649	7171.137126661261	-1	-1
23	4	8	79.1012010022604	0.2638532054140615	3190.9962216539025	-1	-1
24	4	16	135.95587519485872	0.45349998496245936	1257.4594658755946	-1	-1
25	19	8	43.139309220245984	0.14389724647524652	8549.927658635555	-1	-1
26	19	1	290.9931270666027	0.9706485913885222	10632.558496535781	-1	-1
27	26	15	45.12205669071391	0.15051098013517708	8345.210128056435	-1	-1
28	26	9	74.51845409024533	0.24856680714177715	2779.7523314514065	-1	-1
29	11	6	92.11405973031478	0.30725942988970983	8846.94994078825	-1	-1
30	11	1	353.50388965328233	1.1791620510122451	2140.532910649117	-1	-1
31	18	7	100.60318086422517	0.33557608999031313	7720.146247784489	-1	-1
32	18	1	349.89426974444723	1.1671216550232468	7353.291421916889	-1	-1
33	3	16	72.56721022610694	0.24205815820125448	10616.920919808661	-1	-1
34	3	23	75.07329751649384	0.25041756559630945	10343.052272928315	-1	-1
35	12	9	88.84255736976509	0.29634687264135606	8906.044662712222	-1	-1
36	12	8	68.06614430096654	0.2270442183737876	10005.885686752063	-1	-1
37	27	23	140.38874598770374	0.46828645031391597	7168.05802195801	-1	-1
38	27	19	128.191263352851	0.4276000277260177	5377.621062082999	-1	-1
39	17	8	75.66372975210778	0.2523870355407933	4730.550014958861	-1	-1
40	17	6	82.92767933567175	0.2766169632448584	8085.657968842795	-1	-1
41	2	9	216.0	0.7204984456280085	8620.840671290458	-1	-1
42	2	8	188.8279640307547	0.629862289700279	6219.529185755309	-1	-1
43	13	7	25.079872407968907	0.08365744947449247	9693.126008017955	-1	-1
44	13	8	55.31726674375732	0.1845185403021624	9396.617004850088	-1	-1
45	28	6	250.2418829852429	0.834717072786544	7708.982715512105	-1	-1
46	28	15	214.56001491424257	0.7156951724057133	7220.883910945858	-1	-1
47	20	16	66.06814663663572	0.22037961554268226	8617.489172913443	-1	-1

Id	From	To	Length	Delay	BandWidth	ASFrom	ASTo
48	20	4	173.93389548906217	0.5801810247309896	5343.130345246692	-1	-1
49	25	14	53.75872022286245	0.17931978870149712	6828.525179222871	-1	-1
50	25	15	61.032778078668514	0.20358343397240672	10535.289572613614	-1	-1
51	10	8	3.605551275463989	0.012026824488906886	9358.677655300617	-1	-1
52	10	16	74.02702209328699	0.24692756644760888	6894.8481170934965	-1	-1
53	5	22	57.271284253105414	0.1910364411272328	7438.719407539451	-1	-1
54	5	3	35.35533905932738	0.11793271683748421	1400.3275690049813	-1	-1
55	0	3	182.53766734567415	0.6088801184774106	10010.970495078116	-1	-1
56	0	1	62.00806399170998	0.20683663760383852	2329.627575984019	-1	-1