Instituto de Engenharia de Sistemas e Computadores de Coimbra
Institute of Systems Engineering and Computers INESC - Coimbra

Teresa Gomes, Miguel Soares
José Craveirinha, Luísa Jorge e Paulo Melo

Determinação de k caminhos disjuntos nos SRLG

No. 5 2012

ISSN: 1645-2631

Instituto de Engenharia de Sistemas e Computadores de Coimbra
INESC - Coimbra
Rua Antero de Quental, 199; 3000-033 Coimbra; Portugal
www.inescc.pt

This work has been partially supported by FCT under project grant PEst-C/EEI/UI0308/2011, by programme COMPETE of the EC Community Support Framework III and cosponsored by the EC fund FEDER and national funds (FCT - PTDC/EEA-TEL/101884/2008) and PT Inovação by Project "Protecção extremo-a-extremo em redes GMPLS, considerando informação acerca de SRLG".

Determinação de k caminhos disjuntos nos SRLG

Teresa Gomes[(1,2)], Miguel Soares[(1,2)] , José Craveirinha[(1,2)], Luísa Jorge[(2,3)] e Paulo Melo[(3,4)]
[(1)]Departamento de Engenharia Electrotécnica e de Computadores da FCTUC,
Pólo 2 da Univ. Coimbra, 3030-290 Coimbra, Portugal
[(2)]INESC-Coimbra, Rua Antero de Quental 199,
3000-033 Coimbra, Portugal.
[(3)]Instituto Politécnico de Bragança, Campus de Sta Apolónia,
5301-857 Bragança, Portugal
[(4)]Faculdade de Economia, Universidade de Coimbra
Av. Dias da Silva, 165, 3004-512 Coimbra, Portugal

e-mail: teresa@deec.uc.pt, miguel.morais.soares@alunos.deec.uc.pt
jcrav@deec.uc.pt, ljorge@inescc.pt, pmelo@inescc.pt

Publicado em Fevereiro de 2012 e revisto em Fevereiro de 2013

Resumo

A determinação de um conjunto de caminhos disjuntos nos SRLG é um problema NP-Completo, de grande interesse no desenvolvimento de métodos de encaminhamento resiliente. Existem poucas heurísticas para a determinação de$k$ de caminhos disjuntos nos SRLG, e tanto quanto nos foi possível averiguar, não existe nenhuma heurística que tente minimizar o custo total do conjunto de caminhos a determinar. São aqui propostas duas heurísticas que procuram determinar um conjunto de k caminhos disjuntos nos nós e nos SRLG, de custo aditivo mínimo.

1 Introdução

Atualmente, devido à elevada largura de banda que as rede óticas apresentam, o volume de dados transportado por estas redes é muito grande. Como tal, uma avaria durante um período de tempo reduzido pode deixar um número muito grande de utilizadores sem serviço. Esta situação pode ter custos muito elevados, pelo que se tem investido no desenvolvimento de tecnologias que permitem prevenir e recuperar falhas, de forma a que estas não se manifestem como avarias.

Um conceito usual em proteção de redes é o conceito de "Shared Risk Link Group" (SRLG). Um SRLG é um grupo de ligações que partilham o mesmo recurso físico (cabo, conduta, nó, etc) cuja falha resulta na falha de todas as ligações desse grupo [13]. Considere-se por exemplo, uma conduta onde passa um conjunto de cabos de transmissão e que cada cabo representa uma ligação. Se essa conduta for acidentalmente cortada, todos os cabos nessa conduta são cortados e consequentemente todas as ligações associadas a esses cabos falham. Assim, o risco dessas ligações falharem é comum ao risco da conduta ser cortada. Logo, todos as ligações que utilizem os cabos dessa conduta estão associados ao mesmo SRLG. Os protocolos de encaminhamento em redes "Generalized MultiProtocol

Label Switching" (GMPLS) permitem distribuir informação acerca dos SRLG pelos nós da rede [13]. A determinação de um par de caminhos disjuntos nos SRLG é um problema NP-completo [7]. Em [7] encontra-se uma formalização para este problema.

O cálculo de caminhos em redes de grandes dimensões, com múltiplos domínios ou múltiplas camadas, é complexo e exige uma grande capacidade de computação[1]. Uma forma de resolver este problema é distribuir a capacidade de computação pelos nós da rede. Assim, a arquitetura base usada pelo GMPLS consiste em usar "Path Computation Elements" (PCE) em vários pontos da rede [3].

Um PCE é uma unidade computacional que calcula um caminho a pedido de um "Path Computation Client" (PCC), que é um elemento da rede que pretende determinar um ou vários caminhos no domínio da rede a que pertence. Para determinar um caminho, um PCE tem que recorrer a uma base de dados com a informação relativa ao estado da rede. Essa base de dados é denominada por "Traffic Engeneering Database".

Neste contexto o cálculo de rotas pode ser realizado recorrendo a um PCE, de forma centralizada ou distribuída. Um PCE centralizado possui, em geral, grande capacidade de processamento, mas trabalha em processos em que o tempo de resposta poderá ser da ordem dos segundos, pois responde a pedidos do sistema de gestão de rede. Contudo, se o PCE em causa estiver localizado num *router* – como pode acontecer num modelo distribuído – em geral o seu poder de cálculo e os seus recursos de memória são limitados, devendo contudo ter uma resposta rápida aos pedidos que lhe são feitos.

A determinação de um par de caminhos disjuntos nos arcos (nos nós), de custo aditivo mínimo visa, em geral, minimizar o custo dos recursos utilizados em proteção global dedicada. Este problema, designado por min-sum, resolve-se em tempo polinomial utilizando os algoritmos de Suurballe ou de Bhandari [1]. Quando se deseja partilhar largura de banda de proteção pode utilizar-se uma aproximação do tipo min-min, ou seja, procura determinar-se para caminho activo (*Active Path* – AP) o caminho mais curto que é possível proteger, e para caminho de protecção (*Backup Path* – BP) o caminho mais curto na rede em que foram removidos os arcos (ou nós) do caminho a proteger. Este problema é NP-completo [14]. No contexto de partilha de largura de banda, a utilização de recursos da rede pelo AP, pode ser considerada mais importante do que a do BP. Isto pode conduzir ao problema min-sum com pesos assimétricos, em que o custo do AP é considerado ω vezes mais relevante do que o custo do BP [8]. Este problema é também NP-completo [14].

Os problemas min-sum e min-min podem ser formulados, considerando também a necessidade dos caminhos serem disjuntos nos SRLG. Neste caso, o problema min-sum passa a ser NP-completo [7]. Assim, têm sido propostas várias heurísticas para a sua

[1]Por capacidade de computação entende-se a capacidade de processamento juntamente com a memória RAM disponível.

resolução. As heurísticas *"Trap Avoidance"* (TA) [15] e CoSE [11] resolvem de forma bastante eficiente o problema min-min, considerando SRLG. Por outro lado, as heurísticas *"Iterative Two Step Approach"* (ITSA) [6], IMSH [12] e CoSE-MS [4] procuram resolver o problema min-sum de cálculo de um par de caminhos disjuntos nos nós e nos SRLG com custo aditivo mínimo.

A determinação de um conjunto de caminhos disjuntos nos SRLG foi abordada por [10]. Nesse trabalho são atribuídos pesos aos arcos, como base nos SRLG a que pertencem, de forma a tentar maximizar a possibilidade de encontrar o maior número possível de caminhos disjuntos, entre um par de nós origem-destino.

Neste trabalho propõem-se duas heurísticas para a determinação de k de caminhos disjuntos nos nós e nos SRLG, de custo aditivo mínimo.

2 Notação

É apresentada aqui a notação utilizada neste texto.

- **Grafo dirigido:** O grafo $G = (V, A)$ é definido por um conjunto de nós V, $V = \{v_1, \ldots, v_n\}$ e por um conjunto de arcos A, $A = \{a_1, \ldots, a_m\}$.

  Um arco liga dois vértice por uma dada ordem, sendo um par ordenado de elementos pertencentes a V. Se $a, b \in V$, com $a \neq b$ e $(a, b) \in A$, diz-se que a é a **cauda** (ou origem) do arco e b a sua **cabeça** (ou destino).

- **Grafo não dirigido:** O grafo $G = (V, E)$, é definido por um conjunto de nós V, $V = \{v_1, \ldots, v_n\}$ e por um conjunto de arestas E, $E = \{e_1, \ldots, e_m\}$, em que uma aresta é um par de nós (por qualquer ordem).

  Se $a, b \in V$, $a \neq b$, a aresta que liga esse dois nós pode ser representada por $(a, b) \in E$ ou $(b, a) \in E$, em que a e b são os **extremos** da aresta.

- **Arco simétrico:** Considere-se o arco (i, j). O simétrico do arco (i, j) é o arco (j, i).

- **Custo de um arco:** Indica o custo de utilização desse arco num caminho. É representado por $l(a, b)$, onde a é o nó origem e b é o nó destino do arco (a, b).

- **Rede dirigida:** Uma rede dirigida é representada por um grafo dirigido a cujos arcos foram associados atributos, como por exemplo o custo de utilizar um dado arco.

- **Rede não dirigida:** Uma rede não dirigida é representada por um grafo não dirigido a cujas arestas foram associados atributos, como por exemplo o custo de utilizar uma dada aresta.

Uma rede não dirigida pode ser computacionalmente representada por uma rede dirigida, em que cada aresta é substituída por dois arcos, de sentidos opostos, entre o mesmo par de nós (com os mesmos atributos).

- **Caminho:** Sequência contínua de nós (todos diferentes) desde um nó de **origem**, i, até um nó de **destino**, j ($i, j \in V$). Um caminho é representado por $p = \langle i \equiv v_1, v_2, \ldots, j \equiv v_u \rangle$, onde $(v_k, v_{k+1}) \in A, \forall k \in \{1, \ldots, u-1\}$, sendo u o número de nós do caminho.

  Seja A_p o conjunto de arcos que formam o caminho $p = \langle i \equiv v_1, v_2, \ldots, j \equiv v_u \rangle$: $A_p = \cup_{\forall k \in \{1,\ldots,u-1\}}(v_k, v_{k+1})$. Seja V_p o conjunto dos nós do caminho p.

- **Custo (aditivo) de um caminho:** Somatório do custo dos arcos que constituem o caminho p, $C_p = \sum_{(i,j)\in A_p} l(i,j)$. Se um caminho entre um dado par de nós não existe, é representado pelo conjunto vazio ($\emptyset$), e o seu custo é infinito.

- **Par de caminhos:** Conjunto de dois caminhos com o mesmo nó de **origem** e o mesmo nó de **destino**. Um par de caminhos é representado por (p, q), sendo p e q os caminhos constituintes do par.

- **Par de caminhos disjuntos nos nós:** Seja (p, q) um par de caminhos de um nó origem s para um nó destino t. (p, q) é um par de caminhos disjuntos nos nós, se e só se $V_p \cap V_q = \{s, t\}$.

- **Conjunto de k caminhos:** Conjunto de k caminhos com o mesmo nó de **origem** e o mesmo nó de **destino**. Um conjunto de k caminhos é representado por S, onde $k = |S|$.

  Os caminhos no conjunto S, do nó s para o nó t são todos disjuntos entre si, se e só se: $\cap_{p\in S} V_p = \{s, t\}$.

- **Segmento:** Sequência contínua de arcos que constituem parte de um caminho.

- **$C_{(p,q)}$:** Custo (aditivo) de um par de caminhos (p, q), dado pela soma do custo dos caminhos que constituem o par, $C_{(p,q)} = C_p + C_q$.

  Se $(p, q) = (\emptyset, \emptyset)$, o custo do par de caminhos é infinito ($C_{(\emptyset,\emptyset)} = \infty$).

- **C_S:** Custo de um conjunto de caminhos S, dado pelo somatório do custo dos caminhos nesse conjunto, $C_S = \sum_{p\in S} C_p$.

- **$d(i)$:** Distância do nó i ao nó origem, ou seja, a soma do custo de todos os arcos que formam o caminho mais curto desde o nó origem até ao nó i (o qual pode ser determinado usando o algoritmo de Dijkstra, desde que os custos sejam não negativos e não haja custos fixos associados à passagem em nós).

- $P(i)$**:** Nó predecessor do nó i num dado caminho.
- **Nó adjacente:** Um nó, i, diz-se adjacente de um outro nó, j, quando o nó i se encontra ligado através de um arco ao nó j. Desta definição resulta que nó i é adjacente ao nó j se existir o arco (i, j) ou o arco (j, i) (ou ambos).
- **Nó vizinho:** Um nó, j, diz-se vizinho de um outro nó, i, se e só se o nó j estiver ligado ao nó i através do arco (i, j).
- Γ_i**:** Conjunto de nós vizinhos do nó i.
- Y**:** Conjunto dos riscos de falha que podem afetar os arcos da rede. $Y = \{y_1, y_2, \ldots, y_r\}$ em que r é o número de riscos na rede.
- $g_i, i = 1, \ldots, r$**:** g_i é o conjunto de arcos da rede que ficam indisponíveis quando ocorre a falha associada ao risco y_i ($y_i \in Y$). Ou seja g_i é um SRLG.
- R'**:** Conjunto de todos os SRLG existentes na rede. $R' = \{g_1, g_2, \ldots g_r\}$ em que r é o número de SRLG diferentes que existem numa rede.
- $R(i, j)$ ou $R(a) = \{g_k : a_k \in g_k\}$, com $a = (i, j)$: Conjunto dos SRLG a que o arco $a = (i, j)$ pertence.

  Das definições anteriores resulta que $R' = \cup_{(i,j)\in A} R(i, j)$, e $r = |R'|$.
- R**:** Conjunto dos conjuntos de SRLG de todos os arcos da rede: $R = \{R(i, j) : (i, j) \in A\}$, sendo $G = (V, A)$ o grafo associado a essa rede ($|R| = |A|$).
- R_p**:** Conjunto dos SRLG que afetam um caminho p, ou seja, $R_p = \cup_{(i,j)\in p} R(i, j)$.
- $\mathcal{P}_{st}$**:** Conjunto de todos os caminhos de s para t na rede.
- (p^*, q^*)**:** Par de caminhos disjuntos nos nós, ou nos nós e nos SRLG, dependendo do problema a resolver, de menor custo aditivo, ou seja, uma solução ótima para o problema em causa.

Neste trabalho os termos grafo e rede serão usados indistintamente, sendo o termo grafo utilizado preferencialmente quando se pretende fazer referência à topologia de uma rede.

3 Heurísticas para determinação de k caminhos disjuntos nos nós e nos SRLG

Nesta secção iremos descrever duas heurísticas propostas para a determinação de de k caminhos disjuntos nos nós e nos SRLG, de custo aditivo mínimo. Estas heurísticas denotadas por kIMSHd e kCoSE-MScd envolvem essencialmente três passos:

1. determinação de um conjunto de pares de caminhos disjuntos nos nós e nos SRLG, que funciona como conjunto semente;

2. para cada elemento do conjunto semente (de pares de caminhos disjuntos nos nós e nos SRLG) determinado no ponto anterior, determina-se o conjunto de caminhos disjuntos nos nós e nos SRLG de dimensão k, ou se essa dimensão não tiver sido atingida, guarda-se o conjunto (ou conjuntos) de maior dimensão entre os calculados;

3. selecionar o melhor conjunto (o de menor custo total) entre todos os conjuntos de dimensão k (de caminhos disjuntos nos nós e nos SRLG), ou entre os conjuntos de maior dimensão encontrados (caso a dimensão máxima seja inferior a k).

As heurísticas kIMSHd e kCoSE-MScd diferem entre si no passo 1,a a primeira utiliza uma variante da heurística IMSH e segunda uma variante da heurística CoSE-MS.

O passo 2 é conseguido utilizando uma generalização da MSH definida pelo **Algoritmo kMSH**, descrito de seguida. A heurística kMSH corresponde a aplicação da MSH a um conjunto de caminhos (de forma semelhante ao algoritmo de Bhandari [1] para a obtenção de k caminhos disjuntos nos nós).

A função `K-Remove_Interlacing` (S_c,p') na linha 10 do **Algoritmo kMSH** tem dois passos:

1. Remove todos os arcos *comuns* a p' e S_c, ou seja cria um grafo formado pelos arcos de p' e S_c e em seguida remove desse grafo os pares de arcos simétricos que surjam, ou seja os arcos de p' com simétrico em S_c (e os de S_c com simétrico em p');

2. Determina nesse grafo $|S_c| + 1$ caminhos disjuntos nos nós. Isto pode ser feito de duas formas:

   (a) Em cada iteração i ($i = 1, 2, |S_c| + 1$) calcula sucessivamente o caminho mais curto p_i (usando o algoritmo de Dijsktra [2]); em seguida remove os arcos de p_i do grafo antes de passar à iteração seguinte (onde será calculado o caminho mais curto no grafo remanescente);

   (b) Seleciona sequencialmente cada um dos $|S_c| + 1$ arcos de cauda s, e dada a sua cabeça obter o arco emergente desse nó, e assim sucessivamente até seleccionar um dos $|S_c| + 1$ arcos de cabeça t, definindo assim cada um dos caminhos.

Algoritmo kMSH: Heurística kMSH que dado um conjunto semente de caminhos disjuntos nos nós e nos SRLG de dimensão k devolve um conjunto de $k+1$ caminhos disjuntos, ou um conjunto vazio se isso não foi possível.

Dados: Grafo da rede dirigida $G = (V, A)$; nó origem s; nó destino t; o custo $l(i, j)$, e os SRLG, $R(i, j)$, associados a cada arco $(i, j) \in A$; Conjunto semente de caminhos disjuntos nos nós e nos SRLG S_c

Resultado: Conjunto de caminhos disjuntos nos nós e nos SRLG S, obtido usando o conjunto semente S_c, com $|S| = |S_c| + 1$ ou $S = \emptyset$ se não foi possível estender S_c.

1 $\forall p \in S_c$ Remove temporariamente os arcos (i, j) da rede $G, \forall (i, j) \in p$
2 $M \leftarrow \sum_{(i,j)\in A} l(i, j)$
3 $R_{S_c} \leftarrow \cup_{p\in S_c} R_p$ `// Conjunto de SRLG que afectam os caminhos em` S_c
4 **Para cada** *arco* (i, j) *da rede, tal que* $R(i, j) \cap R_S \neq \emptyset$ **faz**
  `// O arco` (i, j) `pertence a pelo menos um dos SRLG que afetam` $p \in S$
5 $l(i, j) \leftarrow l(i, j) + M$
6 **fim**
7 Finaliza a transformação da rede: divide os nós intermédios e inverte os arcos dos caminhos $p \in S_c$ e atribui-lhes o custo nulo.
  `// Dado que a rede não tem arcos com custo negativo, pode ser usado o algoritmo de Dijkstra [2]`
8 $p' \leftarrow$`Dijkstra` (s,t)
9 **Se** *existe* p' **então**
10 $S \leftarrow$ `K-Remove_Interlacing` (S_c,p')
11 **Se** $\cap_{p\in S} R_p \neq \emptyset$ `// Avalia se os` $p \in S$ `são disjuntos nos SRLG`
12 **então**
13 $S \leftarrow \emptyset$ `// Não há solução`
14 **fim**
15 **fim**
16 **Caso contrário**
17 $S \leftarrow \emptyset$ `// Não há solução`
18 **fim**
19 Repõe a rede no estado inicial

Na nossa implementação optámos pela segunda opção, por ser mais eficiente.

3.1 Heurística kIMSHd

Na heurística kIMSHd o conjunto semente é obtido recolhendo todos os pares de caminhos disjuntos nos SRLG gerados usando a heurística IMSHd, que corresponde à heurística IMSH [12], na qual heurística auxiliar MSH foi substituída pela nova versão proposta em [5][2].

[2]Neste relatório a heurística MSHd devolve sempre um par disjunto nos SRLG ou $(\emptyset, \emptyset)$. A heurística MSHcd difere de MSHd pelo facto de, caso exista um par de caminhos disjuntos nos nós, com base no caminho semente fornecido, devolver sempre esse par, assim como o conjunto X de SRLG comum ao par.

Esta heurística, dado um limite de iterações, $i_{\max}$, guarda todos os pares de caminhos disjuntos nos SRLG obtidos nas iterações $i = 1, \ldots, i_{\max}$, numa pilha. Seguidamente é usado cada um dos pares guardados na pilha como par semente na heurística kMSH, e usando esta heurística $k-2$ vezes, até obter o conjunto de dimensão k desejado, ou verificar que isso já não é possível. Por fim, devolve o maior conjunto que for conseguido (de dimensão menor ou igual a k), e entre esses o conjunto de menor custo total aditivo. O pseudo-código da heurística kIMSHd encontra-se no **Algoritmo kIMSHd**,formalizado em seguida, onde MPS representa o algoritmo de enumeração de caminhos mais curtos por ordem crescente do seu custo proposto em [9].

3.2 Heurística kCoSE-MScd

Na heurística kCoSE-MScd o conjunto semente é obtido recolhendo todos os pares de caminhos disjuntos nos SRLG gerados usando a heurística CoSE-MScd, a variante da heurística CoSE-MS proposta em [5].

Esta heurística é em tudo semelhante à kIMSHd, com a diferença de que a recolha dos pares de caminhos é feita com base na heurística auxiliar AllPairs, cujo pseudo-código se encontra no **Algoritmo AllPairs** em anexo. Este algoritmo corresponde a utilizar, a nova variante do algoritmo CoSE-MS proposta em [5], mas recolhendo todos os pares de caminhos disjuntos nos nós e nos SRLG encontrados.

Todos os pares de caminhos disjuntos nos nós e nos SRLG calculados ficam em P (ver linha 1). O ciclo "Enquanto" na linha 13 do **Algoritmo kIMSHd** e na linha 2 do **Algoritmo kCoSE-MScd** é comum a ambas as heurísticas.

4 Análise de resultados

São aqui apresentados os resultados obtidos, no que toca à qualidade das soluções obtidas e aos tempos de execução das heurísticas cujo pseudo-código se encontra no **Algoritmo kIMSHd** e no **Algoritmo kCoSE-MScd**.

Começa-se por se enunciar as condições de experimentação usadas para analisar a qualidade das soluções e para determinar os tempos de execução. Depois são apresentados os resultados que permitem avaliar a qualidade das soluções obtidas, seguidas dos tempos de execução das heurísticas desenvolvidas.

4.1 Condições de experimentação

Esta secção visa introduzir o modo como foram efectuados os testes, quer ao nível da qualidade das soluções óptimas quer ao nível dos tempos de execução de cada uma das heurísticas.

Algoritmo kIMSHd: Heurística kIMSHd que permite obter um conjunto de caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo.

Dados: Grafo da rede dirigida $G = (V, A)$; nó origem s; nó destino t; o custo $l(i, j)$, e os SRLG, $R(i, j)$, associados a cada arco $(i, j) \in A$, dimensão do conjunto a obter, k $(k > 2)$, número máximo de iterações $i_{\max}$.

Resultado: Conjunto de k caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo, ou entre os maiores conjuntos de caminhos disjuntos nos nós e nos SRLG o de menor custo aditivo.

```
1  i ← 0
2  Δ = (min_{(i,j)∈A} l(i,j))/(2|V|)          // Assume que min_{(i,j)∈A} l(i,j) > 0
3  P ← ∅ // Pilha de pares
4  S ← ∅ // Melhor conjunto de caminhos C_S = ∞
5  Enquanto i ≤ i_max faz
6  |  i ← i + 1
   |  // Obtém um caminho semente a cada iteração usando o algoritmo de
   |     MPS
7  |  p_i ←MPS (s,t)                          // O i-ésimo caminho mais curto
8  |  (p'_i, p''_i) ←MSHd (G,s,t,l,R,p_i,Δ,verdade)
9  |  Se (p'_i, p''_i) ≠ (∅, ∅) então
10 |  |  push(P, (p'_i, p''_i))                        // Guarda o par disjunto
11 |  fim
12 fim
13 Enquanto ¬ empty(P) faz
14 |  i = 2
15 |  S_i ← top(P)
16 |  pop (P)
17 |  Enquanto i < k ∧ S_i ≠ ∅ faz
18 |  |  S_{i+1} =kMSH(G, s, t, l, R, S_i)          // Tenta juntar mais um caminho
19 |  |  i ← i + 1
20 |  fim
21 |  Se S_i = ∅ então
22 |  |  i ← i − 1 // S_i é o maior conjunto não vazio na iteração corrente
23 |  fim
   |  // Verifica se precisa de actualizar o melhor conjunto
24 |  Se {i = k ∧ [|S| < k] ∨ (|S| = k ∧ C_{S_i} < C_S)]} ∨ {i < k ∧ [(|S| = |S_i| ∧ C_{S_i} <
   |  C_S)) ∨ |S| < |S_i|]} então
25 |  |  S ← S_i                    // Actualiza o melhor conjunto encontrado
26 |  fim
27 fim
```

Para se poder estudar a qualidade das soluções obtidas por cada uma das heurísticas (**Algoritmo kIMSHd** e **Algoritmo kCoSE-MScd**), foram associadas 10 distribuições distintas de SRLG à mesma rede. A rede utilizada corresponde ao maior componente bi-conectado de uma rede SDH com 231 nós e 471 arestas (representadas por 942 arcos

Algoritmo kCoSE-MScd: Heurística kCoSE-MScd que permite obter um conjunto de caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo.

Dados: Grafo da rede dirigida $G = (V, A)$; nó origem s; nó destino t; o custo $l(i, j)$, e os SRLG, $R(i, j)$, associados a cada arco $(i, j) \in A$, dimensão do conjunto a obter, k ($k > 2$), custo residual Δ ($\Delta > 0$), e usa_Δ (verdade se usa Δ), , número máximo de iterações $i_{\max}$.

Resultado: Conjunto de k caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo, ou entre os maiores conjuntos de caminhos disjuntos nos nós e nos SRLG o de menor custo aditivo.

```
1  P ← AllPairs(G, s, t, l, R, Delta, usa_Δ, i_max)    // Pares de caminhos semente
2  Enquanto ¬ empty(P) faz
3  |  i = 2
4  |  S_i ← top(P)
5  |  pop (P)
6  |  Enquanto i < k ∧ S_i ≠ ∅ faz
7  |  |  S_{i+1} =kMSH(G, s, t, l, R, S_i)            // Tenta junta mais um caminho
8  |  |  i ← i + 1
9  |  fim
10 |  Se S_i = ∅ então
11 |  |  i ← i − 1  // S_i é o maior conjunto não vazio na iteração corrente
12 |  fim
   |  // Verifica se precisa de actualizar o melhor conjunto
13 |  Se  {i = k ∧ [|S| < k] ∨ (|S| = k ∧ C_{S_i} < C_S)]} ∨ {i < k ∧ [(|S| = |S_i| ∧ C_{S_i} <
   |  C_S)) ∨ |S| < |S_i|]} então
14 |  |  S ← S_i                    // Actualiza o melhor conjunto encontrado
15 |  fim
16 fim
```

dirigidos), fornecida pela PT Inovação. Posteriormente foi adicionada a cada aresta a informação relativa aos SRLG a que pertencem. Cada aresta foi associada de forma aleatória a um número de SRLG, que varia entre zero e quatro. Convém também referir que a cada SRLG foram associados no máximo a quatro arestas. Deste modo, foi possível obter 10 variantes da rede que permitiram obter os resultados pretendidos.

Nos resultados apresentados, o número máximo de iterações consideradas (**Algoritmo kIMSHd**) ou número máximo de problemas resolvidos (**Algoritmo kCoSE-MScd**) foram $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$. Já o número de caminhos determinado foi $k = 3, 4$, pois para $k > 4$ as soluções ótimas que era possível obter nas variantes da rede criadas eram muito reduzidas e os tempos de execução eram demasiado elevados.

4.1.1 Qualidade das soluções

Para se poder estudar a qualidade das soluções obtidas por cada uma das heurísticas (**Algoritmo kIMSHd** e **Algoritmo kCoSE-MScd**), utilizando cada uma das 10 va-

riantes da rede, procurou-se obter uma solução (conjunto de k caminhos disjuntos nos nós e nos SRLG, de custo aditivo mínimo) para todos os pares origem destino de cada imagem, considerando o número máximo de problemas/iterações permitidos(as) enunciado na sub-secção 4.1. O passo seguinte foi comparar as soluções obtidas por cada uma das heurísticas desenvolvidas com as soluções ótimas obtidas através de um "software" de "Programação Linear Inteira" (PLI) – CPLEX (versão 12.3). A formulação do problema de PLI para o problema de cálculo de um par de caminhos disjuntos nos nós e nos SRLG, adaptada de [7], encontra-se no apêndice A.

4.1.2 Tempos

Os tempos foram obtidos em duas plataformas distintas: um "Path Computation Element" (PCE)[3] e um "Desktop"[4], pois havia o interesse em avaliar o impacto das limitações computacionais de um PCE (que são significativas) no desempenho dos algoritmos. A obtenção das soluções pelo CPLEX e a recolhas dos tempos de execução num Desktop permite avaliar o compromisso entre a eficiência computacional e a precisão das soluções, possível com a heurística.

Para obter os tempos no PCE foram escolhidos de forma aleatória 1000 pares origem destino distintos, com sementes diferentes para cada uma das variantes da rede utilizadas. Os tempos de execução, para cada imagem, foram medidos para 1000 execuções de cada uma das heurísticas desenvolvidas, considerando o número máximo de problemas/iterações permitidos(as) enunciado na sub-secção 4.1. Foram também obtidos os tempos no "Desktop" anteriormente referido, para se poder comparar o desempenho no PCE relativamente ao desempenho no "Desktop". Nestes testes foi sempre utilizada a biblioteca partilhada.

Para se poder ter uma percepção dos tempos de execução de cada uma das heurísticas, face ao tempo que demora a resolver o mesmo problema utilizando o CPLEX, foram medidos os tempos de execução em cada uma das variantes da rede, para 5000 pares origem destino ($\sim 10\%$ dos pares origem destino da rede), utilizando uma semente diferente (do gerador de números aleatórios) para cada uma das variantes – neste caso sem utilizar a biblioteca partilhada em qualquer das heurísticas.

4.2 Qualidade das soluções

Nesta sub-secção são apresentados os resultados que permitem avaliar os desempenho, no que toca à qualidade das soluções obtidas, para cada heurística.

[3] O PCE utilizado foi uma UNICOM-V5 fornecida pela PT Inovação com as seguintes características: CPU G2_LE *core clock* 330MHz com 128 MB de memória RAM.

[4] O "Desktop" utilizado tem as seguintes características: Intel(R) Core(TM) i7 CPU 870 *core clock* 2.93GHz com 4 GB de memória RAM.

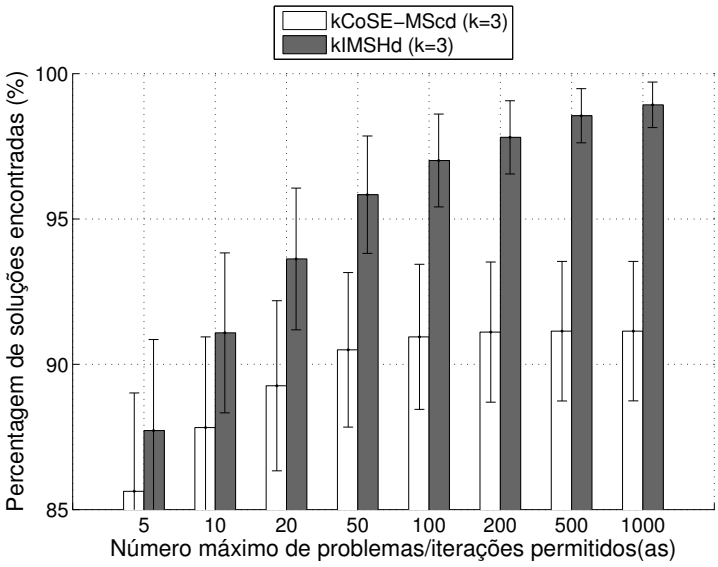

Figura 1: Percentagem média de soluções encontradas pelas heurísticas kIMSHd e kCoSE-MScd, com um número máximo de iterações (kIMSHcd) ou problemas resolvidos (kCoSE-MScd) $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$ e um número de caminhos determinado igual a 3 ($k = 3$), usando o número de soluções obtidas pelo CPLEX como referência.

Relativamente à percentagem de soluções encontradas (figuras 1 e 2) e à percentagem de soluções ótimas encontradas (figuras 3 e 4), é possível identificar três tendências:

- quando o número de caminhos k, a gerar, aumenta, a percentagem de soluções encontradas e a percentagem de soluções ótimas encontradas diminui, porque as soluções para $k = i + 1$ são obtidas a partir das encontradas para $k = i$.

- quando o número de problemas/iterações permitidos(as) aumenta, a percentagem de soluções encontradas e a percentagem de soluções ótimas encontradas também aumenta, em particular para heurística kIMSHd.

- em todos os casos a heurística kIMSHd encontra mais soluções e mais soluções ótimas que a heurística kCoSE-MScd, o que advém da forma mais exaustiva como kIMSHd gera os caminhos semente.

No que toca à percentagem de soluções encontradas pela heurística kCoSE-MScd é possível verificar que esse valor estabiliza para $i_{\max} = 50$ problemas permitidos (cerca de 91% para $k = 3$ e 83% para $k = 4$). Já usando a heurística kIMSHd a percentagem de soluções encontradas aumenta sempre com o número de iterações permitidas, sendo que é cerca de 96% ($k = 3$) e 93% ($k = 4$) para $i_{\max} = 50$ iterações permitidas. Convém referir que as semi-amplitudes dos Intervalos de Confiança (IC) a 95% para ambas as heurísticas, tanto para $k = 3$ como para $k = 4$, são inferiores em 10% às médias correspondentes[5].

[5]Para a kCoSE-MScd o IC máximo para $k = 3$ é aproximadamente 6,8% e para $k = 4$ é aproxima-

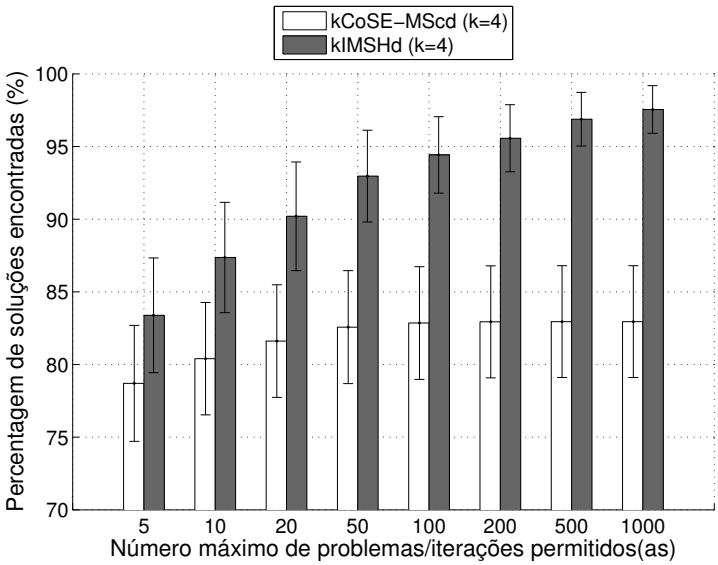

Figura 2: Percentagem média de soluções encontradas pelas heurísticas kIMSHd e kCoSE-MScd, com um número máximo de iterações (kIMSHcd) ou problemas resolvidos (kCoSE-MScd) $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$ e um número de caminhos determinado igual a 4 ($k = 4$), usando o número de soluções obtidas pelo CPLEX como referência.

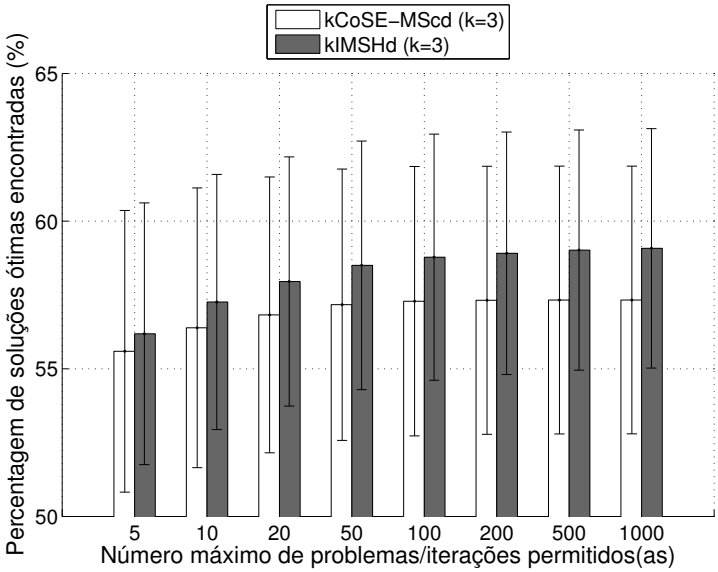

Figura 3: Percentagem média de soluções ótimas encontradas pelas heurísticas kIMSHd e kCoSE-MScd, com um número máximo de iterações (kIMSHcd) ou problemas resolvidos (kCoSE-MScd) $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$ e um número de caminhos determinado igual a 3 ($k = 3$), usando o custo das soluções ótimas obtidas pelo CPLEX como referência.

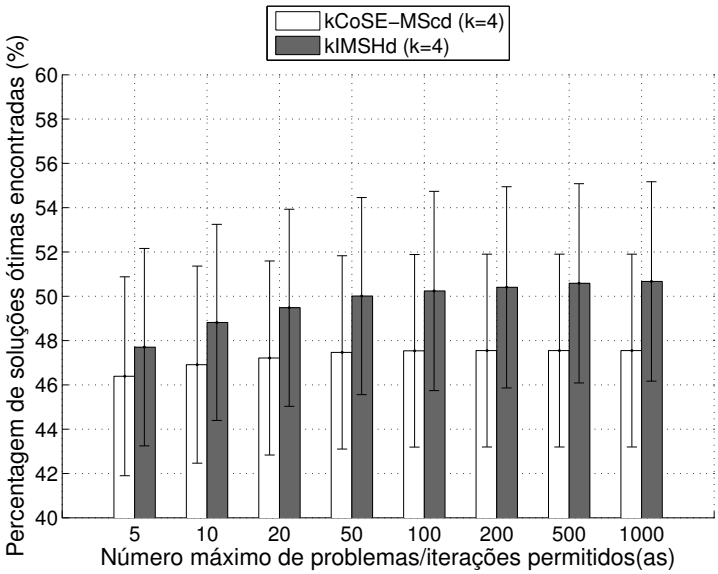

Figura 4: Percentagem média de soluções ótimas encontradas pelas heurísticas kIMSHd e kCoSE-MScd, com um número máximo de iterações (kIMSHcd) ou problemas resolvidos (kCoSE-MScd) $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$ e um número de caminhos determinado igual a 4 ($k = 4$), usando o custo das soluções ótimas obtidas pelo CPLEX como referência.

Relativamente à percentagem de soluções ótimas encontradas, pode-se verificar o mesmo que sucedeu com a percentagem de soluções encontradas. No entanto, para $i_{\max} = 50$ problemas/iterações permitidos(as) a heurística kCoSE-MScd encontrou em média 57% ($k = 3$) e 47% ($k = 4$) e, a heurística kIMSHd encontrou em média 59% ($k = 3$) e 50% ($k = 4$) das soluções ótimas que existem. Um outro aspeto importante é o facto da semi-amplitude dos IC a 95% para ambas as heurísticas, tanto para $k = 3$ como para $k = 4$, ser cerca de 15% da percentagem média de soluções ótimas encontradas[6]. De notar que a heurística kCoSE-MScd apresenta um erro relativo médio inferior à heurística kIMSHd, embora possua uma maior número de soluções sub-ótimas que esta última.

No que toca ao erro relativo médio (figuras 5 e 6) é possível verificar que este varia muito pouco com o número de problemas/iterações permitidos(as) e com a heurística usada.

Ao analisar atentamente o erro relativo médio, é possível verificar que para $k = 3$, o erro relativo médio está compreendido entre 7,0% e 7,4% para a heurística kCoSE-MScd e, 7,2% e 7,9% para a heurística kIMSHd. Já para $k = 4$ o erro relativo médio passa a estar compreendido entre 12,7% e 13,2% para a heurística kCoSE-MScd, e entre 13,4% e 15,0% para a heurística kIMSHd. É importante referir que os IC estão contidos nos

damente 8,0%. Já no caso da kIMSHd o IC máximo para $k = 3$ é 6,3% e para $k = 4$ o IC máximo é 7,9%.

[6]Para a heurística kCoSE-MScd o IC máximo para $k = 3$ é aproximadamente 9,6% e para $k = 4$ é aproximadamente 9,0%. Já no caso da heurística kIMSHd o IC máximo para $k = 3$ é 8,7% e para $k = 4$ o IC máximo é 9,0%.

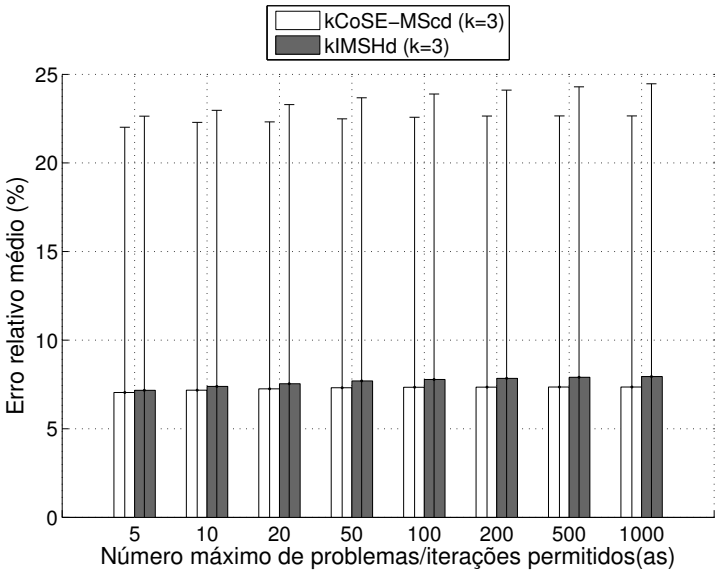

Figura 5: Erro relativo médio (tendo como referência as soluções ótimas obtidas pelo CPLEX) para as heurísticas kIMSHd e kCoSE-MScd, com um número máximo de iterações (kIMSHd) ou problemas resolvidos (kCoSE-MScd) $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$ e um número de caminhos determinado igual a 3 ($k = 3$).

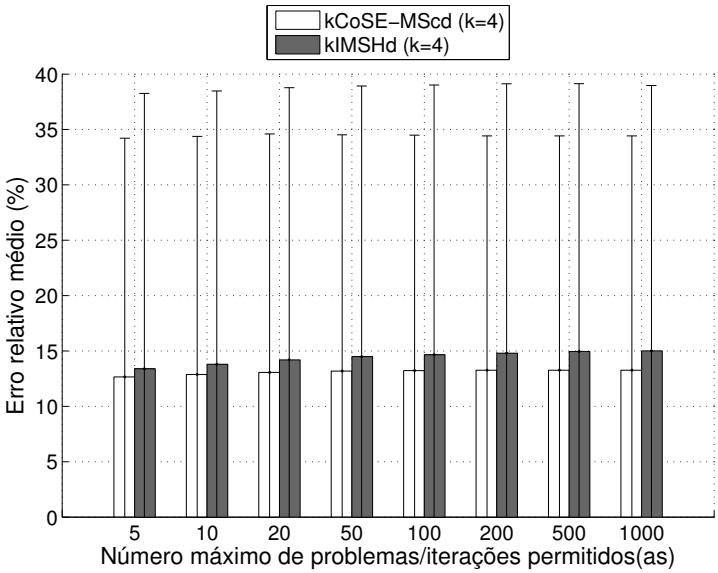

Figura 6: Erro relativo médio (tendo como referência as soluções ótimas obtidas pelo CPLEX) para as heurísticas kIMSHd e kCoSE-MScd, com um número máximo de iterações (kIMSHd) ou problemas resolvidos (kCoSE-MScd) $i_{\max} = 5, 10, 20, 50, 100, 200, 500, 1000$ e um número de caminhos determinado igual a 4 ($k = 4$).

intervalos de 30,0% a 30,6% (kCoSE-MScd para $k = 3$), de 42,3% a 49,1% (kCoSE-MScd para $k = 4$), de 31,0% a 33,0% (kIMSHd para $k = 3$) e de 47,9% a 49,7% (kIMSHd para $k = 4$). Facilmente se pode verificar a grande variabilidade do erro relativo, pois os IC são aproximadamente quatro vezes superiores aos valores médios.

4.3 Tempos

Esta sub-secção visa fornecer a informação relativa ao desempenho computacional das heurísticas implementadas. Assim são fornecidos os tempos relativos a cada heurística para o cálculo de 2 ($k = 2$) e 3 ($k = 3$) caminhos. Os tempos apresentados dividem-se em quatro tabelas:

- tabela 1 que contém os tempos para 1000 execuções das heurísticas implementadas (todos os pares origem destino são diferentes), na UNICOM-V5, usando uma biblioteca dinâmica;
- tabela 2 que contém os tempos para 1000 execuções das heurísticas implementadas (todos os pares origem destino são diferentes), no "Desktop", usando uma biblioteca dinâmica;
- tabela 3 que contém os tempos para 5000 execuções das heurísticas implementadas (todos os pares origem destino são diferentes), no "Desktop", não usando uma biblioteca dinâmica;
- tabela 4 que contém os tempos para 5000 execuções do CPLEX (todos os pares origem destino são diferentes), no "Desktop", não usando uma biblioteca dinâmica.

Estas tabelas permitem que se compare:

- o desempenho das heurísticas num PCE face ao desempenho que é conseguido num "Desktop" (tabelas 1 e 2);
- o desempenho das heurísticas face ao desempenho que se obtém usando o CPLEX para resolver o mesmo problema (tabelas 3 e 4).

Comparando os tempos da tabela 1 com os tempos da tabela 2 para $i_{\max} = 50$, é possível verificar que a heurística kCoSE-MScd é sempre mais rápida. No entanto os tempos obtidos no PCE para $i_{\max} = 50$ são sempre superiores a 400 ms, o que torna as heurísticas pouco aconselhadas para serem usadas no plano de controlo de uma rede GMPLS.

Através dos resultados apresentados nas tabelas 3 e 4 é possível ver que os tempos obtidos no "Desktop" para as heurísticas, para $i_{\max} = 50$, são cerca de 5% do tempo que

o CPLEX levou a resolver o problema na mesma plataforma. Além disso, com qualquer uma das heurísticas com $i_{\max} = 50$, é possível obter resultados em menos de 23ms.

Embora estas heurísticas não sejam adequadas ao plano de controlo utilizando um valor de $k = 50$, com o qual já se consegue obter soluções para um número elevado de pares origem destino (superior a 95% e a 90% para $k = 3$ e a $k = 4$, respectivamente) são, mesmo assim, uma ordem de grandeza mais rápidas do que o CPLEX à custa de menor qualidade nas soluções obtidas.

5 Conclusão

Foram propostas duas heurísticas para o problema de determinar k caminhos disjuntos nos nós e nos SRLG. Os resultados apresentados na secção 4 mostram que a heurística kIMSHd encontra um número elevado de soluções. Ambas as heurísticas apresentam um número de soluções ótimas entre 55% e 60%, mas o erro relativo das soluções sub-ótimas é inferior a 10%, para $k = 3$. Já para $k = 4$ o número de soluções ótimas encontradas está entre 45% e 50% e o erro relativo das soluções sub-ótimas passa a ser inferior a 15%.

No que concerne ao tempo de execução a heurística kCoSE-MScd foi a que apresentou melhor desempenho, não sendo contudo adequada para utilização no plano de controlo do GMPLS, num PCE com as características da placa UNICOM-v5.

O número de soluções obtidas pela heurística kIMSH indica que poderia ser utilizada no plano de gestão de uma rede GMPLS.

$i_{\max}$	*kIMSHd ($k = 3$)* (segundos)	*kCoSE-MScd ($k = 3$)* (segundos)	*kIMSHd ($k = 4$)* (segundos)	*kCoSE-MScd ($k = 4$)* (segundos)
5	119.366± 3.672	55.964± 0.704	138.136± 3.868	76.404± 0.765
10	168.790± 3.091	104.772± 1.215	207.493± 2.364	145.167± 1.498
20	256.983± 3.684	193.321± 2.217	337.474± 4.198	269.854± 2.716
50	532.430± 7.202	411.002± 5.781	736.473± 10.461	577.644± 7.635
100	1004.407± 13.116	682.339±16.001	1416.268± 20.440	960.767± 21.589
200	1975.215± 28.008	1006.493±37.746	2802.737± 42.465	1416.237± 50.531
500	4977.985± 63.961	1314.054±76.332	7088.324±111.981	1841.568±103.020
1000	10118.374±138.904	1384.289±84.286	14248.716±275.524	1935.250±112.548

Tabela 1: Tabela com os tempos de execução para as heurísticas de cálculo de um conjunto de K caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo (kIMSHcd e kCoSE-MScd), com $i_{\max} = \{5, 10, 20, 50, 100, 200, 500, 1000\}$ e um número de caminhos determinados igual a 3 ($k = 3$) e a 4 ($k = 4$), medidos na *UNICOM-V5* usando a biblioteca partilhada (1000 pares origem destino).

$i_{\max}$	*kIMSHd* ($k = 3$) (segundos)	*kCoSE-MScd* ($k = 3$) (segundos)	*kIMSHd* ($k = 4$) (segundos)	*kCoSE-MScd* ($k = 4$) (segundos)
5	4.893±0.736	2.744±0.427	7.339±0.305	3.375±0.481
10	8.758±0.292	5.294±0.752	11.296±0.106	6.168±0.851
20	14.217±0.192	9.953±1.241	18.717±0.226	11.474±1.536
50	30.097±0.406	18.838±2.518	41.483±0.476	27.063±2.921
100	57.439±0.784	30.000±3.125	80.096±0.866	42.703±3.797
200	113.086±1.880	48.727±4.592	158.431±1.671	63.911±4.356
500	283.319±3.748	58.410±5.464	397.142±4.027	82.336±5.120
1000	575.950±7.271	65.086±6.176	801.683±8.891	85.007±6.175

Tabela 2: Tabela com os tempos de execução para as heurísticas de cálculo de um conjunto de K caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo (kIMSHcd e kCoSE-MScd), com $i_{\max} = \{5, 10, 20, 50, 100, 200, 500, 1000\}$ e um número de caminhos determinados igual a 3 ($k = 3$) e a 4 ($k = 4$), medidos no "Desktop" usando a biblioteca partilhada (1000 pares origem destino).

i_{max}	*kIMSHd* ($k = 3$) (segundos)	*kCoSE-MScd* ($k = 3$) (segundos)	*kIMSHd* ($k = 4$) (segundos)	*kCoSE-MScd* ($k = 4$) (segundos)
5	15.913± 0.284	9.483± 0.163	18.894± 0.215	12.554± 0.164
10	23.258± 0.298	17.865± 0.285	29.356± 0.262	23.931± 0.283
20	37.859± 0.392	33.069± 0.559	50.202± 0.442	44.612± 0.589
50	82.883± 0.731	70.719± 1.266	113.937± 0.964	96.020± 1.429
100	159.635± 1.274	118.607± 2.905	221.808± 1.917	161.210± 3.583
200	316.639± 2.641	176.701± 6.913	440.727± 3.705	239.923± 8.907
500	798.393± 5.958	231.615±14.794	1109.120± 9.643	313.115±19.268
1000	1619.466±12.307	244.393±17.566	2243.048±20.911	329.601±22.681

Tabela 3: Tabela com os tempos de execução para as heurísticas de cálculo de um conjunto de K caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo (kIMSHcd e kCoSE-MScd), com um número de caminhos determinados igual a 3 ($k = 3$) e a 4 ($k = 4$), medidos no "Desktop" não usando a biblioteca partilhada (5000 pares origem destino).

	CPLEX (versão 12.3) (segundos)
$k = 3$	1705.853±77.120
$k = 4$	2122.749±183.772

Tabela 4: Tabela com os tempos de execução para o cálculo de um conjunto de K caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo, usando o CPLEX (versão 12.3), com $i_{\max} = \{5, 10, 20, 50, 100, 200, 500, 1000\}$ e um número de caminhos determinados igual a 3 ($k = 3$) e a 4 ($k = 4$), medidos no "Desktop" não usando a biblioteca partilhada (5000 pares origem destino).

A Formulação de PLI para o poblema

A formulação do problema de PLI para o problema de cálculo de um par de caminhos disjuntos nos nós e nos SRLG adaptada de [7], é apresentada de seguida, sendo primeiro

é introduzida a notação adicional necessária:

- $\delta(i)^+$: conjunto dos arcos emergentes do nó $i \in V$.
- $\delta(i)^-$: conjunto dos arcos incidente do nó $i \in V$.
- $h_{g,(i,j)}$, com $g \in R'$ e $(i,j) \in A$, indica se o SRLG g contém o arco (i,j)

$$h_{g,(i,j)} = \begin{cases} 1 & \text{se } g \in R(i,j), \\ 0 & \text{caso contrário;} \end{cases} \tag{1}$$

- $x_{(i,j),k}$ é a variável binária de decisão do arco $(i,j) \in A$ associado ao caminho p_k $(k = 1, 2)$, onde,

$$x_{(i,j),k} = \begin{cases} 1 & \text{se o arco } (i,j) \text{ pertence ao caminho } p_k, \\ 0 & \text{caso contrário;} \end{cases} \tag{2}$$

- $z_{g,k}$ é a variável binária de decisão dos SRLG que afetam o caminho p_k $(k = 1, 2)$, onde,

$$z_{g,k} = \begin{cases} 1 & \text{se } g \text{ está associado ao caminho } p_k, \text{ ou seja se } g \in R_{p_k}, \\ 0 & \text{caso contrário;} \end{cases} \tag{3}$$

É agora introduzida a formulação do problema de cálculo de um par de caminhos disjuntos nos nós e nos SRLG em PLI.

$$\min \quad \sum_{(i,j)\in A} l(i,j)(x_{(i,j),1} + x_{(i,j),2}) \tag{4}$$

$$\text{s. a:} \quad \sum_{(i,j)\in\delta(i)^+} x_{(i,j),k} - \sum_{(j,i)\in\delta(i)^-} x_{(j,i),k} = \begin{cases} 1 & : \ i = s, \\ -1 & : \ i = t, \\ 0 & : \ i \in V\backslash\{s,t\} \end{cases}, \quad i \in V, k = 1,2 \tag{5}$$

$$\sum_{(i,j)\in A} h_{g,(i,j)} x_{(i,j),k} \leq |A| z_{g,k} \quad g \in R', k = 1,2 \tag{6}$$

$$z_{g,1} + z_{g,2} \leq 1, \quad g \in R' \tag{7}$$

$$\sum_{k\in\{1,2\}} \sum_{(i,j)\in\delta(i)^+} x_{(i,j),k} \leq 1, \quad i \in V\backslash\{s\}, \tag{8}$$

$$\sum_{k\in\{1,2\}} \sum_{(j,i)\in\delta(i)^-} x_{(j,i),k} \leq 1, \quad i \in V\backslash\{t\}, \tag{9}$$

$$x \text{ e } z \text{ são variáveis binárias de decisão} \tag{10}$$

- A restrição 5 garante que os arcos (i, j) selecionadas com base em $x_{(i,j),k}$, constituem o caminho p_k $(k = 1, 2)$ de s para t.

- A restrição 6 implica que se g está contido no caminho p_k, então algum arco do conjunto desse SRLG tem que ser usado. Se um arco está associada a vários SRLG, então só pode ser escolhido para o caminho p_k, se e só se os elementos contidos em R' estiverem contidos em p_k. O coeficiente $|A|$ é usado, pois p_k pode conter vários arcos que estejam associados a um SRLG.

- A restrição 7 garante que nenhum dos elementos do conjunto R' está contido em ambos os caminhos. Assumindo que cada arco pertence a pelo menos um SRLG $(\forall (i, j) \in A : R(i, j) \neq \emptyset)$, a disjunção nos SRLG garante a disjunção nos arcos.

- As restrições 8 e 9 garantem que os caminhos são disjuntos nos nós.

B Recolha de pares de caminhos usando CoSE-MS

A heurística kCoSE-MScd precisa de uma versão da heurística CoSE-MS que recolha todos os pares de caminhos disjuntos nos SRLG que esta possa encontrar. Essa tarefa é feita pela heurística AllPairs, cujo pseudo-código se encontra no **Algoritmo AllPairs**. Em [4] foi proposta a heurística MBH que dado o caminho mais curto na rede procura determinar um par de caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo, devolvendo $(\emptyset, \emptyset)$ se não foi possível obter um par de caminhos ou se o par resultante não é disjunto nos SRLG. A heurística MBHc é semelhante à heurística MBH, mas caso exista um par de caminhos disjuntos nos nós, com base no caminho semente fornecido, devolve sempre esse par, assim como o conjunto de SRLG comum ao par, X (que será vazio se os caminhos forem disjuntos nos SRLG ou se o par de caminhos não existir).

A heurística "Confliting SRLG Exclusion" (CoSE), apresentada em Rostami et al (2007), procura resolver o problema min-min, considerando SRLG. A heurística CoSE-MS, proposta em [4] procura resolver a formulação min-sum para esse mesmo problema. A heurística CoSE-MS (tal como a CoSE), vai dividindo os SRLG na rede em dois conjuntos: o conjunto de inclusão I e o conjunto de exclusão E. O primeiro indica os SRLG que não devem ser excluídos e o segundo os que devem ser excluídos no cálculo do caminho semente (o caminho que vai dar origem a dois caminhos possivelmente disjuntos nos SRLG). Sempre que um caminho semente não pode ser protegido ou não pode dar origem a um par de caminhos disjuntos nos SRLG, é identificado o conjunto dos SRLG conflituosos (aqueles que se usados por um caminho semente impedem que seja obtido um par disjunto nos SRLG). O caminho semente é o caminho mais curto no problema auxiliar corrente. Um problema é representado por $P(I, E, H)$, onde I e E são os conjuntos de inclusão,

Algoritmo AllPairs: Heurística AllPairs (auxiliar de kCoSE-MScd) que procura obter uma pilha de pares de caminhos disjuntos nos nós e nos SRLG de custo aditivo mínimo.

Dados: Grafo da rede dirigida $G = (V, A)$; nó origem s; nó destino t; o custo $l(i, j)$, e os SRLG, $R(i, j)$, associados a cada arco $(i, j) \in A$, custo residual Δ ($\Delta > 0$), número máximo de iterações $i_{\max}$.

Resultado: Pilha de pares de caminhos disjuntos nos nós e nos SRLG, S_p

Considere-se uma pilha de problemas vazia, S, e uma pilha de pares de caminhos vazia, S_p
// (p, q) é o melhor par corrente de custo $C_{(p,q)}$
$(p, q) \leftarrow (\emptyset, \emptyset)$ // Recorde-se que $C_{(\emptyset,\emptyset)} = \infty$
$P_0 \leftarrow (\emptyset, \emptyset, \emptyset)$, push (P_0)
$i \leftarrow 0$ // Para limitar o número de problemas resolvidos
Enquanto $\neg$empty$(S) \wedge i < i_{\max}$ **faz**
    $P_c(I_c, E_c, H_c) \leftarrow$ top (S) // problema corrente
    pop (S) // remove S do topo da pilha de problemas
    $i \leftarrow i + 1$ // Actualiza número de problemas resolvidos
    $p_c \leftarrow$ caminho mais curto para o problema corrente
    // Caso exista um caminho para o problema corrente
    Se $C_{p_c} \neq \infty$ **então**
        Se $P_c = P_0$ **então**
            $(p', q', X) \leftarrow$MBHc(G, s, t, l, R, p_c) **else**
                $(p', q', X) \leftarrow$MSHcd$(G, s, t, l, R, p_c, Delta, Verdade)$
            fim
        fim
        Se $X = \emptyset \wedge (p', q') \neq (\emptyset, \emptyset)$ // q' e p' são disjuntos nos SRLG
        então
            push$(S_p, (p', q'))$ // Guarda par disjunto
        fim
        else
            Se $P_c = P_0 \vee (X = \emptyset \wedge (p', q') = (\emptyset, \emptyset)$ **então**
                // Se é P_0 ou não existe par de caminhos disjuntos nos nós
                $T \leftarrow$SRLG_Exclusion (I_c, p_c)
            fim
            else
                // Existe par de caminhos disjuntos nos nós mas não nos SRLG
                $T \leftarrow X \cap R_{p_c} \backslash I_c$
            fim
            // Seja T o conjunto $\{A_1, A_2, \ldots, A_{|T|}\}$
            $H \leftarrow E_c \cup H_c$
            $P_1(I_1, E_1, H_1) \leftarrow P(\emptyset, \{A_1\}, H)$
            push (S, P_1)
            $i \leftarrow 2$
            Enquanto $i \leq |T|$ **faz**
                $P_i(I_i, E_i, H_i) \leftarrow P(I_{i-1} \cup E_{i-1}, \{A_i\}, H)$
                push (S, P_i)
                $i \leftarrow i + 1$
            fim
        fim
    fim
fim

exclusão, e H é a memória dos problemas anteriores, respectivamente. O conjunto H contém todos os SRLG que foram removidos nos problemas que antecederam o problema em causa. O problema inicial é um problema "vazio", pois $I = H = E = \emptyset$. O problema inicial é representado por $P_\emptyset$. Dado um problema corrente, $P(I_c, E_c, H_c)$, o caminho semente, p_c, é calculado na rede onde foram removidos todos os SRLG contidos em $H_c \cup E_c$. Seguidamente, e dado o caminho semente, p_c, é utilizada a heurística Modified Bhandari's Heuristic (MBH) em [4] ou a Modified Suurballe's Heuristic (MSH) em [12] conforme o problema seja o problema vazio ou não, respectivamente.

Dado um caminho semente, p_c, as heurísticas MBH e MSH procuram determinar um par de caminhos na rede de custo aditivo mínimo. Sempre que a MBH ou a MSH não conseguem obter um par de caminhos disjuntos nos SRLG, a heurística CoSE-MS utiliza a função Conflicting SRLG Exclusion, descrita em [11], para obter o conjunto T dos SRLG conflituosos, ou seja aqueles que se pertencerem ao caminho semente, impedem a obtenção de uma solução. Seja o problema corrente $P(I_c, E_c, H_c)$ e p_c o caminho semente a partir do qual não foi possível obter uma solução. Com base nesse conjunto T são obtidos novos problemas como se descreve de seguida. Se o conjunto de SRLG conflituosos for $T = g_1, g_2, \ldots, g_{|T|}$, os novos sub-problemas gerados são $P = (\emptyset, \{g_1\}, H_c \cup E_c)$, $P = (\{g_1\}, \{g_2\}, H_c \cup E_c)$,...,$P = (\{g_1, g_2, \ldots, g_{|T|-1}\}, \{g_{|T|}\}, H_c \cup E_c)$. Cada um destes problemas é resolvido de forma semelhante ao explicado anteriormente, sendo selecionada a melhor solução entre todas as encontradas. Um problema para o qual não é possível determinar um AP é um problema sem solução (que não dá origem a novos problemas). A heurística CoSE-MS pode facilmente ser adaptada para encontrar pares de caminhos disjuntos nos nós e nos SRLG, que se pretende que sejam de custo aditivo mínimo. Para tal basta modificar as heurísticas MBH e MSH, de forma a dividir os nós intermédios do caminho semente.

Referências

[1] R. Bhandari. *Survivable Networks, Algorithms for Diverse Routing*. Kluwer Academic Publishers, Norwell, Massachusetts, USA, 1999.

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

[3] A. Farrel, J.-P. Vasseur, and J. Ash. A path computation element (PCE)-based architecture. ITEF RFC 4655, 2006.

[4] T. Gomes, C. Simões, and L. Fernandes. Resilient routing in optical networks using SRLG-disjoint path pairs of min-sum cost. *Telecommunication Systems Journal*, pages 1–13, 2011. on-line first.

[5] T. Gomes, M. Soares, J. Craveirinha, P. Melo, L. Jorge, V. Mirones, and A. Brízído. Protecção em redes GMPLS considerando informaçao acerca dos SRLG. In *4º Encontro Nacional de Riscos Segurança e Fiabilidade*. Instituto Superior Técnico, 2012. Abstract accepted. Full paper under revisison.

[6] P.-H. Ho and H. T. Mouftah. Shared protection in mesh WDM networks. *IEEE Communications Magazine*, 42(1):70–76, January 2004.

[7] J. Q. Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, March 2003.

[8] P. Laborczi, J. Tapolcai, P.-H. Ho, T. Cinkler, A. Recski, and H. T. Mouftah. Algorithms for asymmetrically weighted pair of disjoint paths in survivable networks. In T. Cinkler, editor, *Proceedings of Design of Reliable Communication Networks (DCRN 2001)*, pages 220–227, Budapest, Hungary, October 7-10 2001.

[9] E. Martins, M. Pascoal, and J. Santos. Deviation algorithms for ranking shortest paths. *International Journal of Foundations of Computer Science*, 10(3):247–263, 1999.

[10] E. Oki, N. Matsuura, K. Shiomoto, and N. Yamanaka. A disjoint path selection scheme with shared risk link groups in GMPLS networks. *IEEE Communications Letters*, 6(9):406–408, September 2002.

[11] M. J. Rostami, S. Khorsandi, and A. A. Khodaparast. CoSE: A SRLG-disjoint routing algorithm. In *Proceedings of the Fourth European Conference on Universal Multiservice Networks (ECUMN'07)*, Toulouse, France, 2007.

[12] A. Todimala and B. Ramamurthy. IMSH: An iterative heuristic for SRLG diverse routing in WDM mesh networks. In *13th International Conference on Computer Communications and Networks, ICCCN'2004*, pages 199–204, October 2004.

[13] J.-P. Vasseur, M. Pickavet, and P. Demeester. *Network Recovery – Protection and Restoration of Optical, SONET-SDH, IP, and MPLS*. Elsevier, 2004.

[14] D. Xu, Y. Chen, Y. Xiong, C. Qiao, and X. He. On finding disjoint paths in single and dual link cost networks. In *IEEE INFOCOM 2004*, Hong Kong, 2004.

[15] D. Xu, Y. Xiong, C. Qiao, and G. Li. Trap avoidance and protection schemes in networks with shared risk link groups. *Journal of Lightwave Technology*, 21(11):2683–2693, November 2003.