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

Luísa Jorge
Teresa Gomes

Uma versão melhorada de um algoritmo de encaminhamento para protecção local com partilha de largura de banda de protecção

No. 8 2006

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

Uma versão melhorada de um algoritmo de encaminhamento para protecção local com partilha de largura de banda de protecção

Luísa Jorge[1,3] e Teresa Gomes[2,3]

[1] Escola Superior de Tecnologia e de Gestão do
Instituto Politécnico de Bragança
Campus de St[a] Apolónia, 5301-857 Bragança, Portugal

[2] Departamento de Engenharia Electrotécnica e de Computadores da FCTUC,
Pólo 2 da Univ. Coimbra, 3030-290 Coimbra, Portugal

[3] INESC-Coimbra, Rua Antero de Quental 199,
3000-033 Coimbra, Portugal.

e-mail: ljorge@inescc.pt, teresa@deec.uc.pt,

April 5, 2007

Abstract

A new improved version of a dynamic routing algorithm of locally restorable QoS guaranteed paths, will be described. The algorithm calculates simultaneously the active and local protection paths, with protection bandwidth sharing. A method for minimal allocation of protection bandwidth in the context of local shared protection will also be proposed. The performance of the original and of the improved version of algorithm, using minimal allocation of protection bandwidth, will be analysed. Experimental results will show the new version has advantages both regarding the total bandwidth used, the number of hops of the active path and the probability of rejection of new requests.

Resumo

O QoS duma rede depende fortemente da sua capacidade de sobrevivência. Por este motivo mecanismos de protecção em redes têm sido objecto de estudo intenso. A protecção pode ser local ou global e pode ser orientada ao recurso ou ao caminho. A protecção local tem vantagens sobre a protecção global, nomeadamente no respeitante ao tempo de recuperação. Os caminhos de protecção e activo podem ser calculados separadamente ou em conjunto. O cálculo conjunto dos caminhos de protecção e do caminho activo é mais difícil mas pode conduzir a melhor utilização dos recursos da rede, especialmente se for considerada a partilha de LB de protecção.

Será descrita uma versão melhorada de um algoritmo de encaminhamento dinâmico para protecção local com garantias de QoS. O algoritmo calcula simultaneamente o caminho activo e os caminhos de protecção local, com partilha da LB de protecção. Será também proposto um método que permite reservar o valor mínimo de LB de protecção, num contexto de protecção local com partilha. Será analisado o desempenho de ambas as versões do algoritmo, implementadas com reserva exacta da LB de protecção. Os resultados simulacionais mostrarão que a nova versão requer menos LB total usada, utiliza um menor número de ramos no caminho activo e conduz a uma menor rejeição de pedidos.

1 Introdução

O *MultiProtocol Label Switching* (MPLS) tem sido proposto como uma solução para oferecer serviços de telecomunicações fiáveis, eficientes e diferenciados. As tecnologias orientadas à ligação, como é o caso do MPLS, permitem a utilização da engenharia de tráfego na selecção dos caminhos, *Label Switched Path*s (LSPs), e são também supostas melhorar a fiabilidade das redes IP.

Os mecanismos de protecção da camada de rede demoram demasiado a recuperar uma falha, o que não é aceitável para muitas aplicações, pelo que um enquadramento para recuperação baseada no MPLS foi proposto em [15]. Os dois modelos básicos de recuperação que aí foram propostos são a recuperação por reencaminhamento e a protecção por comutação. Na recuperação por reencaminhamento o caminho de recuperação (o caminho usado pelo tráfego recuperado depois da ocorrência de uma falha na rede) apenas é sinalizado depois da detecção da falha no caminho activo (o caminho que transporta o tráfego antes da falha ocorrer). A protecção por comutação pré-sinaliza o caminho de protecção (ou um conjunto de caminhos de protecção) antes da detecção da falha no caminho activo (o caminho protegido).

No *Fast Reroute* (FRR) [13] foram normalizados dois esquemas, os quais permitem a redirecção do tráfego em dezenas de milissegundos. O *Facility backup* protege recursos (nós ou ramos) implementando *bypass tunnels*. No esquema *One-to-one backup* para cada caminho protegido é necessário estabelecer caminhos de protecção (designados por *detour* LSP). Esses caminhos de protecção local têm origem em cada nó do caminho responsável pela recuperação (o *Point of Local Repair* (PLR)) e terminam no nó destino. Este esquema tem a desvantagem de necessitar de um largo número de caminhos de recuperação. Para minimizar a Largura de Banda (LB) consumida, é utilizada a fusão de caminhos (entre *detours* de um LSP protegido e o próprio LSP), sempre que tal é possível. Numa rede com FRR qualquer destes esquemas será disponibilizado a LSPs que solicitaram protecção garantindo o *Quality of Service* (QoS) do LSP protegido. A recuperação nas camadas inferiores (do modelo OSI) são geralmente mais rápidas, contudo as redes baseadas em MPLS, com FRR, permitem fazer recuperação diferenciada (ou seja permitem recuperar apenas os fluxos de tráfego que solicitaram protecção) pelo que consomem menos recursos.

Numa rede, com certos esquemas de protecção, a minimização da LB de protecção pode ser conseguida resolvendo o problema *min-sum*: encontrar k caminhos disjuntos, entre dois nós distintos s e t, tal que a soma do custo dos caminhos seja mínima. Em [16] foi proposto um algoritmo que resolve este problema em tempo polinomial. A partilha de LB entre caminhos de protecção é desejável porque permite reduzir o consumo de LB. Esta partilha conduz a redes cujos custos nos ramos são não uniformes, ou seja redes com custos diferentes no caminho activo e nos caminhos de recuperação. A LB reservada em cada ramo de um caminho de protecção, associado a um dado caminho activo, é sempre menor ou igual do que a LB que seria requerida, nesse ramo, pelo caminho activo. Assim encontrar um par de caminhos disjuntos (o caminho activo e o caminho de protecção) conduz ao problema *min-sum* com custos duais ordenados (*Min-Sum with Ordered Dual cost links* (MSOD)). Este problema é NP-completo para redes dirigidas e não dirigidas, segundo [9] e [17], respectivamente. Obter k caminhos disjuntos, de s para t ($s \neq t$), numa rede com k custos diferentes em cada ramo, tal que o custo total dos caminhos é mínimo, foi estudado em [10]. Os ca-

minhos poderão ser disjuntos nos ramos ou nos nós e as redes poderão ser dirigidas ou não dirigidas. Inicialmente provaram que o problema é fortemente NP-completo mesmo para $k = 2$, quando a relação entre os k custos nos ramos (no mesmo ramo) é arbitrária. Duas heurísticas que resolvem, em tempo polinomial, o problema *min-sum* foram propostas e limiares para o pior caso foram também obtidos para essas heurísticas. A abordagem de Ho *et al.* [1] ao problema do encaminhamento resiliente (resistente a falhas) requer a resolução do problema *min-sum* para o par de caminhos formado pelo caminho activo e pelo caminho de protecção, em que o custo do caminho de protecção depende do custo do caminho activo seleccionado. Este problema é semelhante ao abordado em [8]. Em [1] este problema é formulado como um processo de programação linear inteira. Uma vez que este problema é NP-completo foram propostas duas heurísticas: uma aproximação iterativa com dois passos (Iterative Two-Step-Approach – ITSA) e um método de relaxação de máxima verosimilhança (Maximum Likehood Relaxation – MLR). O MLR é um algoritmo de Dijkstra modificado, com complexidade temporal polinomial. A ITSA já tinha sido esboçada em [9] em que foi designada por um melhoramento duma aproximação com dois passos (Enhancement to the Two-Step-Approach – TSA).

Em [5, 6, 7, 8] são apresentados três cenários para o cálculo de LB de protecção partilhada: sem informação – *No Information* (NI), com informação parcial – *Partial Information* (PI) e com informação completa – *Complete Information* (CI). No modelo NI apenas é conhecida em cada ramo a LB total reservada e a LB residual (livre); o modelo CI requer o conhecimento completo de cada caminho activo e de protecção; no modelo PI devem ser conhecidas a LB total usada pelos caminhos activos e de protecção em cada ramo (apenas a informação agregada, sem ser discriminada para cada fluxo) e também a LB livre em cada ramo. O único modelo com interesse prático é o PI, podendo os outros dois ser utilizados para gerar limites inferiores e superiores para a LB requerida pelo PI (embora o modelo CI seja uma aproximação mais precisa torna-se impraticável devido à quantidade enorme de informação necessária à sua implementação).

Algoritmos de recuperação global para estes três modelos de informação são apresentados em [5, 7, 8], considerando apenas falhas isoladas dos ramos, embora possam ser adaptados para incluírem falhas isoladas dos nós. Qiao and Xu [14] propuseram um modelo PI para protecção global o qual afirmam ser mais eficiente que a aproximação semelhante em [5]. A característica mais relevante dos esquemas propostos em [14] é a sua capacidade para atribuir (libertar) LB de protecção mínima (máxima). Esta característica também está presente em [7] em que é designada por " PI com reservas exactas".

Têm sido publicados muitos outros trabalhos no contexto de encaminhamento com capacidades de sobrevivência, para redes MPLS e ópticas, entre os quais [2, 11, 12]. Em [3] é apresentado um resumo de esquemas de protecção nas redes MPLS. Liu et al. [11] propuseram um método matricial de cálculo da LB de reserva (protecção), dados os caminhos activo e de protecção para cada fluxo de tráfego. Propuseram também um algoritmo designado por encaminhamento com capacidade de sobrevivência sucessivo – Successive Survivable Routing (SSR).

A contribuição principal deste relatório é uma versão melhorada de um algoritmo [6] de encaminhamento para protecção local com partilha de largura de banda de protecção, e um método para reserva mínima de LB de protecção em todos os ramos dos caminhos de protecção local. O método proposto para reserva mínima de LB de protecção é uma adaptação

da aproximação em [14] para protecção global (o qual é semelhante às aproximações propostas por [11, 7]). A versão original dos algoritmos em [4, 6] contém algumas gralhas e imprecisões as quais serão justificadamente corrigidas. Resultados experimentais mostrarão que a versão proposta neste relatório usa menos LB, conduz a caminhos activos com menor número de ramos e a uma menor probabilidade de rejeição de novos pedidos.

O relatório é organizado da seguinte forma. Na secção 2 será revisto o algoritmo proposto em [6] e será introduzida a notação utilizada. A secção 3 encontra-se dividida em duas subsecções. Na subsecção 3.1 serão apresentados os algoritmos original e alterado, lado a lado, para evidenciar as diferenças entre as duas versões. Na subsecção 3.2, serão ilustradas as correcções e as alterações propostas através de exemplos que demonstram a pertinência das correcções propostas e justificam as alterações introduzidas. A secção 4 apresentará uma proposta para que permite fazer reserva exacta de LB de protecção. Finalmente na secção 5, serão apresentados resultados simulacionais que mostram as vantagens da nova versão. Por último, na secção 6, serão apresentadas algumas conclusões.

2 Revisão de LOCAL_EDGE_DISJOINT e notação

Seja $G' = (V, E')$, o grafo da rede, em que V é o conjunto dos nós e E' o conjunto dos ramos dirigidos. Seja $G = (V, E)$, o grafo da rede com todos os ramos invertidos. LOCAL_-EDGE_DISJOINT(s, t) [6] é um algoritmo que calcula simultaneamente o caminho activo e os caminhos de protecção locais, de s para t ($s, t \in V$). O primeiro passo do algoritmo é a inversão de todos os ramos do grafo da rede – isto torna difícil a referência sem ambiguidades a uma dado ramo. Neste texto os ramos pertencerão sempre a G (o grafo da rede com todos os ramos invertidos), excepto quando indicado em contrário. O algoritmo passa iterativamente por todos os ramos da rede, calculando o seu peso como o custo de os proteger usando um caminho de protecção local [4, 6]. Este algoritmo chama o sub-algoritmo ALT_-PAHT_COST(k, j) o qual calcula o custo mínimo de proteger o ramo (k, j). Isto é conseguido através de chamadas sucessivas ao sub-algoritmo SHORT_PRED_PATH(k, u, j), em que u é um nó no caminho activo candidato, de t para k, o que permite a ALT_PAHT_COST(k, j) escolher o caminho local de protecção de custo mínimo.

LOCAL_EDGE_DISJOINT(s, t) minimiza o custo de proteger o caminho activo, levando em conta a partilha de LB entre LSPs de protecção para o LSP activo calculado (partilha intra-pedido) e também entre LSPs de protecção de diferentes LSPs activos (partilha inter-pedido), no modelo PI para protecção de falhas isoladas de ramos.

A notação introduzida em [6] será utilizada com pequenas alterações. Um pedido é uma solicitação de um LSP protegido de s para t. O pedido r solicita b_r unidades de LB. Designa-se por k o último nó etiquetado permanentemente em LOCAL_EDGE_DISJOINT(s, t) e por j um nó adjacente a k, o qual é o nó candidato corrente a integrar o caminho activo candidato que contém k. O custo mínimo de proteger o ramo (k, j) é calculado por ALT_PAHT_-COST(k, j). Este valor é obtido chamando SHORT_PRED_PATH(k, u, j) para cada nó u no caminho mais curto de t para k (incluindo k e t e com início em k). Isto porque o caminho de protecção do ramo (k, j) pode começar em qualquer nó no caminho mais curto de t para k. A lista de símbolos usados nos algoritmos é apresentada de seguida (assumindo que todos os nós pertencem a V e todos os ramos a E):

- A_{ij}: conjunto dos pedido que usam o ramo (i, j) no caminho activo.
- C_{ij}: capacidade do ramo (i, j).
- F_{ij}: total da LB reservada para os pedidos que usam o ramo (i, j) no caminho activo, $F_{ij} = \sum_{k \in A_{ij}} b_k$.
- B_{ij}, G_{ij}: total da LB reservada para proteger ramos que utilizam o ramo (i, j) nos respectivos caminhos de protecção.
- R_{ij}: LB residual no ramo (i, j): $R_{ij} = C_{ij} - F_{ij} - B_{ij}$.
- λ^u_{mn}: LB que seria reservada pelo conjunto dos caminhos que protegem os ramos no caminho activo candidato do nó t até u, para o pedido corrente.
- δ_{mn}: $\delta_{mn} = F_{kj} + b - B_{mn} - \lambda^k_{mn}$
- l_{mn}: custo do ramo (m, n) na determinação do caminho de protecção mais curto.
- T (T'): conjunto dos nós temporariamente etiquetados no algoritmo de Dijkstra.
- P (P'): conjunto dos nós permanentemente etiquetados no algoritmo de Dijkstra.
- ϕ (γ): etiquetas dos nós no algoritmo de Dijkstra em LOCAL_EDGE_DISJOINT(s, t) (SHORT_PRED_PATH(k, u, j)). ϕ_i (γ_a) custo do caminho mais curto do nó t (u) para o nó i (a), se $i \in P$ ($a \in P'$). ϕ_i (γ_a) custo do caminho corrente do nó t (u) para o nó i (a), se $i \in T$ ($a \in T'$).
- Q (Q'): predecessores dos nós no algoritmo de Dijkstra em LOCAL_EDGE_DISJOINT(s, t) (SHORT_PRED_PATH(k, u, j)). $Q(i)$ ($Q(a)$) predecessor do nó i (a) no caminho mais curto de t (u) no algoritmo de Dijkstra, se $i \in P$ ($a \in P'$). $Q(i)$ ($Q(a)$) predecessor do nó i (a) no caminho corrente se $i \in T$ ($a \in T'$).
- λ^u: vector formado por λ^u_{ij}. Vector de dimensão igual ao número de ramos na rede. Se u é um nó permanentemente etiquetado no algoritmo de Dijkstra, então λ^u atingiu o seu valor final.
- β: vector auxiliar que armazena temporariamente λ^u.

3 Alterações ao algoritmo de Kodialam and Lakshman

Pretendendo tornar claras as referências às instruções dos vários algoritmos e sub-algoritmos, sempre que for utilizada a palavra **"linha"** (de um algoritmo) estamos a referir-nos a uma instrução da nova versão do algoritmo, e sempre que for utilizada a palavra **"passo"** (de um algoritmo) estamos a referir-nos a uma instrução da versão original.

3.1 Algoritmos original e alterado

Na coluna do lado esquerdo é apresentado o algoritmo original (conforme apresentado em [6]), e na coluna do lado direito é apresentado o algoritmo com as correcções e alterações que propomos e também com as gralhas corrigidas.

LOCAL_EDGE_DISJOINT(s,t) como em [6]

- INITIALIZATION
  1: Reverse all links in the network.
  2: $T = V$, $P = \emptyset$, $\phi_t = 0$, $\phi_j = \infty$, $\forall j \neq t$,$\lambda^d_{mn} = 0\ \forall (m,n) \in E$, $Q(t) = \emptyset$.
- ITERATIVE STEP
  2: $k = \arg\min_{j \in T} \phi_j$. If $k = s$ goto Step 6.
  3: $T = T \backslash \{k\}$ and $P = P \cup \{k\}$.
  4: For each $j \in T, (k,j) \in E$
  w_{kj} =ALT_PATH_COST(k,j)
  if $(\phi_j \geq w_{kj} + \phi_k)$
  $\phi_j = \phi_k + w_{kj}$
  $Q(j) = k$
  5: Go to Step 2.
- TERMINATION
  6: Exit.

LOCAL_EDGE_DISJOINT(s,t) – nova versão

1: Reverse all links in the network
2: $T \leftarrow V$, $P \leftarrow \emptyset$, $\phi_t \leftarrow 0$, $\phi_j \leftarrow \infty$, $\forall j \neq t$, $\lambda^t_{mn} \leftarrow 0\ \forall (m,n) \in E$, $Q(t) \leftarrow \emptyset$
3: $k \leftarrow \arg\min_{j \in T} \phi_j$
4: **while** $k \neq s$ **do**
5: $T \leftarrow T \backslash \{k\}$ and $P \leftarrow P \cup \{k\}$
6: **for** $j \in T, (k,j) \in E$ **do**
7: **if** $R_{kj} \geq b$ **then**
8: $w_{kj} \leftarrow$ALT_PATH_COST$(k,j)+b$
9: **if** $\phi_j > w_{kj} + \phi_k$ **then**
10: $\phi_j \leftarrow \phi_k + w_{kj}$
11: $Q(j) \leftarrow k$
12: $\lambda^j_{mn} \leftarrow \tau_{mn} \forall (m,n) \in E$
13: **end if**
14: **end if**
15: **end for**
16: $k \leftarrow \arg\min_{j \in T} \phi_j$
17: **end while**

ALT_PATH_COST(k,j) como em [6]

- INITIALIZATION
  1: $u = k$, $MIN = \infty$.
- ITERATIVE STEP
  2: If $u = \emptyset$ go to Step 6.
  3: $\alpha =$ SHORT_PRED_PATH(k,u,j).
  4: if $(\alpha \leq MIN)$
  $MIN = \alpha$.
  $\lambda^j_{mn} = \beta_{mn} \forall (m,n)\ \in\ E$.
  5: $u = Q(u)$ Go to Step 2.
- TERMINATION
  6: Exit.

ALT_PATH_COST(k,j) – nova versão

1: $u \leftarrow k$, $MIN \leftarrow \infty$
2: **while** $u \neq \emptyset$ **do**
3: $\alpha \leftarrow$SHORT_PRED_PATH(k,u,j)
4: **if** $\alpha \leq MIN$ **then**
5: $MIN \leftarrow \alpha$
6: $\tau_{mn} \leftarrow \beta_{mn} \forall (m,n) \in E$
7: **end if**
8: $u \leftarrow Q(u)$
9: **end while**

SHORT_PRED_PATH(k, u, j) como em [6]

- INITIALIZATION

1: $\delta_{mn} = F_{kj} + b - B_{mn} - \lambda^u_{mn}$, $\forall (m,n) \in E$.

2: $l_{mn} =$

$$\begin{cases} 0 & \text{if } \delta_{mn} \leq 0 \\ \delta_{mn} & \text{if } 0 \leq \delta_{mn} \leq b \text{ and } R_{mn} \geq \delta_{mn} \\ & \text{and } (m,n) \neq (k,j) \\ \infty & \text{otherwise} \end{cases}$$

3: $T' = V$, $P' = \emptyset$, $\gamma_u = 0$, $\gamma_j = \infty \forall j \neq u$, $\lambda^d_{mn} = 0 \, \forall \, (m,n) \in E$

- ITERATIVE STEP

4: $w = \arg\min_{j \in T} w_j$.
If $w = k$ go to Step 9.

5: $T' = T' \backslash \{w\}$ and $P' = P' \cup \{w\}$.

6: For each $i \in T'$, $(w,i) \in E$
if $(\gamma_i \geq l_{wi} + \gamma_w)$
$\gamma_i = l_{wi} + \gamma_w$
$Q'(i) = w$

7: Go to Step 2.

- TERMINATION

8: Set $\beta_{mn} = \lambda^u_{mn} \forall (mn) \in E$, if arc (mn) is on the shortest path from u to j set $\beta_{mn} = \lambda^u_{mn} + l_{mn}$.

9: Exit.

SHORT_PRED_PATH(k, u, j) – nova versão

1: $\delta_{mn} = \min(F_{kj} + b - B_{mn}, b) - \lambda^k_{mn}, \forall (m,n) \in E$.

2: $l_{mn} =$

$$\begin{cases} 0 & \text{if } \delta_{mn} \leq 0 \text{ and } (m,n) \neq (k,j) \\ \delta_{mn} & \text{if } 0 \leq \delta_{mn} \text{ and } R_{mn} \geq \delta_{mn} \\ & \text{and } (m,n) \neq (k,j) \\ \infty & \text{otherwise} \end{cases}$$

3: $T' \leftarrow V$, $P' \leftarrow \emptyset$, $\gamma_u \leftarrow 0$, $\gamma_h \leftarrow \infty \, \forall \, h \neq u$, $Q'(u) \leftarrow \emptyset$

4: $w \leftarrow \arg\min_{h \in T'} \gamma_h$

5: **while** $w \neq j$ **do**

6: $T' \leftarrow T' \backslash \{w\}$ and $P' \leftarrow P' \cup \{w\}$

7: **for** $i \in T'$, $(w,i) \in E$ **do**

8: **if** $\gamma_i > l_{wi} + \gamma_w$ **then**

9: $\gamma_i \leftarrow l_{wi} + \gamma_w$

10: $Q'(i) \leftarrow w$

11: **end if**

12: **end for**

13: $w \leftarrow \arg\min_{h \in T'} \gamma_h$

14: **end while**

15: **for** $(m,n) \in E$ **do**

16: **if** (m,n) is on the shortest path from u to j **then**

17: $\beta_{mn} \leftarrow \lambda^k_{mn} + l_{mn}$

18: **else**

19: $\beta_{mn} \leftarrow \lambda^k_{mn}$

20: **end if**

21: **end for**

Apresenta-se a seguir uma lista com as gralhas detectadas no algoritmo original em [6] (no algoritmo principal e num sub-algoritmo) e respectivas correcções.

No algoritmo LOCAL_EDGE_DISJOINT

- Substituir $\lambda^d_{m,n}$ (no passo 2) por $\lambda^t_{m,n}$ (como é visível na linha 2).
- A numeração dos passos deveria ser "1:, 2:, 3:, ..., 7:" e não "1:, 2:, 2:, 3:, ..., 6:".

No sub-algoritmo SHORT_PRED_PATH

- Substituir $\gamma_j \leftarrow \infty \forall \, j \neq u$ (no passo 3) por $\gamma_h \leftarrow \infty \forall \, h \neq u$ (como é visível na linha 3), para evitar confusões com o parâmetro de entrada.
- Retirar $\lambda^d_{mn} = 0 \, \forall \, (m,n) \in E$ (no passo 3).
- Acrescentar $Q'(u) = \emptyset$ (no passo 3), ver linha 3.
- Substituir $j \in T$ (no passo 4) por $h \in T'$ (como é visível na linha 4).
- Substituir w_j (no passo 4) por γ_h (como é visível na linha 4).

- Substituir "Step 9" (no passo 4) por "Step 8".
- Substituir $w = k$ (no passo 4) por $w = j$ (ver linha 5).
- Substituir "if $(\gamma_i \geq l_{wi} + \gamma_w)$" (no passo 6) por "**if** $(\gamma_i > l_{wi} + \gamma_w)$" (como visível na linha 8), pois não vemos necessidade da alteração em caso de igualdade.
- Substituir "Step 2" (no passo 7) por "Step 4".

Encontramos também algumas incorrecções no algoritmo original (no algoritmo principal e sub-algoritmos), as quais se não forem corrigidas conduzirão a resultados indesejados, como se mostrará na secção 3.2. As incorrecções detectadas são:

1. No sub-algoritmo **SHORT_PRED_PATH**(k, u, j), alterar a condição para atribuir um custo nulo a l_{mn}, de $\delta_{mn} \leq 0$ (passo 2) para $\delta_{mn} \leq 0$ and $(m, n) \neq (k, j)$ (linha 2). Só assim é garantido que ramos do caminho activo não sejam seleccionados para fazer parte dos caminhos de recuperação correspondentes.
2. No passo 4, do algoritmo **LOCAL_EDGE_DISJOINT**(s, t), a iteração j só deve ser efectuada se a LB residual do ramo (k, j), R_{kj}, for superior ou igual a b (como visível na linha 7). Ou seja, um ramo só poderá fazer parte do caminho activo se tiver LB residual suficiente para o pedido em causa.
3. O valor de λ^j_{mn} não deve ser actualizado no passo 4 do sub-algoritmo **ALT_PATH_COST**(k, j) mas sim no passo 4 do algoritmo **LOCAL_EDGE_DISJOINT**(s, t). Para tal, no algoritmo **ALT_PATH_COST**(k, j) a instrução $\lambda^j_{mn} = \beta_{mn} \forall (m, n) \in E$ (no passo 4) foi substituída pela instrução $\tau_{mn} \leftarrow \beta_{mn} \forall (m, n) \in E$ (linha 6), e foi acrescentada a instrução $\lambda^j_{mn} \leftarrow \tau_{mn} \forall (m, n) \in E$ (linha 12) no algoritmo **LOCAL_EDGE_DISJOINT**(s, t). Assim evita-se a alteração de λ^j_{mn} quando a etiqueta, ϕ_j, e o predecessor, $Q(j)$, de j não sofrem alteração. Essa alteração poderia levar a que λ^j_{mn} tome valores incorrectos os quais poderão ser usados pelo algoritmo.
4. No sub-algoritmo **SHORT_PRED_PATH**(k, u, j), no cálculo dos β_{mn} (no passo 8), onde está λ^u deveria estar λ^k (ver linha 17 e 19). Caso contrário poderá acontecer que o nó origem do caminho activo fique com informação errada acerca da LB requerida nos arcos dos caminhos de protecção.

Uma versão mais eficiente de LOCAL_EDGE_DISJOINT(s, t) (considerando que todas as incorrecções e gralhas detectadas foram corrigidas) é obtida fazendo as seguintes alterações:

1. No sub-algoritmo **SHORT_PRED_PATH**(k, u, j) (no passo 1 – cálculo de δ_{mn}) onde está λ^u surge agora λ^k na nova versão (ver linha 1). Desta forma utiliza-se informação mais actualizada o que poderá conduzir a uma redução da LB de protecção, pois é utilizada toda a informação disponível para partilha intra-pedido.
2. No algoritmo **LOCAL_EDGE_DISJOINT**(s, t), alterar o valor a atribuir ao custo w_{kj} de **ALT_PATH_COST**(k, j) (passo 4) para **ALT_PATH_COST**$(k, j) + b$ (linha 8). Desse modo esse custo toma não só em consideração o custo dos ramos do caminho de protecção mas também o custo do ramo do caminho activo.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	0	0	0	0	0	0	0	0	0	0
B_{nm}	0	0	0	0	0	0	0	0	0	0	0
R_{nm}	250	250	75	250	250	250	250	250	250	250	250

Tabela 1: Estado da rede antes do estabelecimento de qualquer pedido.

3. No sub-algoritmo **SHORT_PRED_PATH**(k, u, j) (no passo 1) onde está $\delta_{mn} = F_{kj} + b - B_{mn} - \lambda^u_{mn} \forall (m, n) \in E$ passa a estar $\delta_{mn} = \min(F_{kj} + b - B_{mn}, b) - \lambda^k_{mn}, \forall (m, n) \in E$ na nova versão (ver linha 1). Dado um pedido de LB b, nenhum ramo dos caminhos de protecção do caminho activo correspondente necessitará de um aumento na reserva da LB de protecção superior a b. Assim a expressão presente na linha 1 da nova versão é uma estimativa mais precisa do valor mínimo da LB de protecção requerida no ramo (m, n) para proteger o ramo (k, j).

   Esta alteração implica que δ_{mn} será no máximo b e por conseguinte o teste $\delta_{mn} \leq b$, no passo 2, torna-se redundante pelo que foi removido na linha 2 da nova versão.

   Na realidade, no algoritmo original, a atribuição do valor δ_{mn} ao custo l_{mn}, no sub-algoritmo **SHORT_PRED_PATH**(k, u, j) (no passo 2), apenas se $\delta_{mn} \leq b$ faz com que possam existir situações em que um pedido é rejeitado mesmo havendo LB disponível na rede.

Em [6] o seguinte texto "the cost of using link (k, j) in the active path is the sum of bandwidth currently being routed and the cost of backing up link (k, j)" parece indicar que a alteração 2 teria estado sob consideração por Kodialam e Lakshman. No entanto em LOCAL_EDGE_-DISJOINT(s, t) [4, 6] escolheram uma abordagem diferente: a minimização da LB de protecção.

A alteração em 1 (combinada com a correcção 4) contribuirá para aumentar a partilha de LB de protecção, e a alteração em 2 poderá melhorar os tempos de propagação do tráfego nos caminhos activos e possivelmente também para reduzir a LB total necessária. A alteração 3 fará com que os custos estimados para os ramos de protecção sejam mais adequados.

3.2 Ilustração da pertinência das modificações propostas

Nesta subsecção recorreremos a exemplos para ilustrar a pertinência das correcções e alterações propostas. Nesses exemplos recorremos à rede ilustrada na Figura 1 e a pedidos particulares nessa rede. Daqui em diante o caminho activo será designado por *Active Path* (AP) e os caminhos de protecção usados para recuperar um dado AP serão designados por *Recovery Path* (RP), para uma descrição mais sucinta dos exemplos e dos resultados apresentados. A Tabela 1 corresponde ao estado inicial da rede.

Em cada um dos vários exemplos seguintes começa-se por estabelecer, em geral, um determinado número de pedidos. Os caminhos para estes pedidos serão determinados com a versão alterada do algoritmo. Para cada exemplo, os estados após esses pedidos iniciais serão os pontos de partida para ambos os algoritmos (original e alterado).

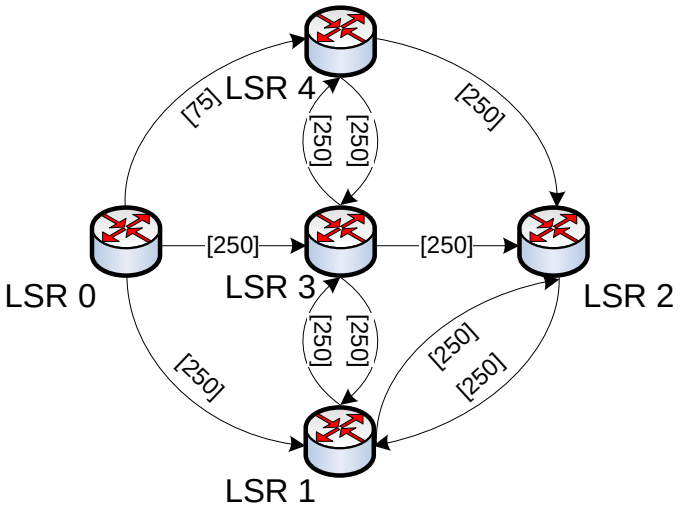

Figura 1: Rede exemplo usada para ilustrar as correcções e alterações propostas ao algoritmo.

Pedido	Origem-Destino	LB
1º	LSR0-LSR2	60
2º	LSR0-LSR2	50

Tabela 2: Pedidos considerados no exemplos para a Correcção 1.

Lembra-se que o algoritmo começa por inverter os ramos da rede e a partir daí determina o AP e juntamente os RP do destino até à origem. No entanto, nesta subsecção, nas tabelas representam-se os ramos da rede original e quando são referidos ramos da rede são também apresentados de acordo com o sentido original (e não com o sentido inverso que o algoritmo usa). Perceberá os exemplos mais facilmente se tiver presente que o indice k corresponde ao último nó que foi etiquetado definitivamente, o índice j corresponde ao nó que se está a tentar re-etiquetar e o índice u ao nó destino do caminho de protecção.

3.2.1 Correcção 1

Deve ser notado que, no algoritmo **SHORT_PRED_PATH**(k, u, j) no teste do passo 2 (versão original), a condição apresentada para atribuir um custo nulo a l_{mn} arrisca-se a escolher para os RP troços pertencentes ao AP, se nestes o valor de LB para protecção ($B_{mn} + \lambda^k_{mn}$) ultrapassar a LB necessária para AP (F_{kj}) somada à LB para o pedido em causa (b).

Os pedidos para o exemplo desta correcção foram os apresentados na Tabela 2.

A Tabela 3 corresponde ao estado da rede após o estabelecimento do primeiro pedido. Na determinação dos caminhos para o segundo pedido (Tabela 2), quando o algoritmo **SHORT_PRED_PATH**(k, u, j) procura um RP para o ramo LSR3-LSR2 ($j = 3$ e $k = u = 2$) começa por determinar os valores dos vectores δ e l. Os valores que obtém são apresentados na Tabela 4 (determinam-se facilmente a partir da Tabela 3), que mostra que foi atribuído um custo zero para a utilização do ramo LSR3-LSR2 como protecção do ramo LSR3-LSR2. Como é óbvio,

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	60	0	0	60	0	0	0	0	0	0	0
B_{nm}	0	60	0	0	60	0	0	60	0	0	0
R_{nm}	190	190	75	190	190	250	250	190	250	250	250

Tabela 3: Estado da rede após o estabelecimento do primeiro pedido.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
δ_{nm}	50	-10	50	50	-10	50	50	-10	50	50	50
l_{nm}	50	0	50	50	0	50	50	0	50	50	50

Tabela 4: Valores dos vectores δ e l numa invocação particular de **SHORT_PRED_PATH**(k, u, j) na versão original do algoritmo.

com esses valores para os custos o algoritmo de Dijkstra (código da linha 3 à linha 14) obtém para RP o ramo LSR3-LSR2. O que finalmente resultará na escolha dos RP LSR0-LSR3-LSR2 e LSR3-LSR2 e do AP LSR0-LSR3-LSR2 (não disjuntos).

Este problema é evitado, na nova versão, através da correcção da condição que atribuir um custo nulo a l_{mn} (linha 2). Deste modo, um ramo que pertença ao AP será sempre impedido de ser utilizado em RP (através da atribuição de um custo infinito).

3.2.2 Correcção 2

A necessidade desta correcção parece-nos demasiado óbvia para necessitar de um exemplo.

3.2.3 Correcção 3

Como é sabido o vector λ^k guarda a quantidade de LB necessária para os RP de todos os ramos no caminho (na árvore de caminhos mais curtos) do nó k até ao nó destino, para o pedido actual.

Para o exemplo seguinte acrescentámos o ramo LSR3-LSR0 (com capacidade 250) à rede apresentada na Figura 1. Foi tentado estabelecer, com o algoritmo original (com as Correcções 1, 2 e 4 feitas), o pedido entre o LSR0 (origem) e o LSR2 (destino) com 5 unidades de LB. Para o pedido em causa o algoritmo **LOCAL_EDGE_DISJOINT**(s, t) começa por etiquetar definitivamente o LSR2 ($k = 2$) e temporariamente todos os restantes com etiquetas ∞, após o que vai re-etiquetar temporariamente todos os nós adjacentes a LSR2 (LSR1, LSR3 e LSR4), quando $j = 1, 3$ e 4 (passo 4 desse algoritmo), e calcular os λ^j, indirectamente pela chamada do algoritmo **ALT_PATH_COST**(k, j) (no passo 4 deste), correspondentes a esses nós (ver Tabela 5). As etiquetas temporárias desses nós passam todos a ter o valor 10 ($\phi_j = 10$). O algoritmo escolhe etiquetar definitivamente o LSR1 ($k = 1$) após o que na análise das suas adjacências (LSR0 e LSR3) o valor de λ^3 é alterado para reflectir o custo da eventual inclusão do ramo LSR1-LSR3 na árvore de caminhos mais curtos. No entanto, como o custo mínimo dessa inclusão - protecção do ramo LSR3-LSR1 com o ramo LSR3-LSR2 – somado à etiqueta

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
λ^1	0	0	0	0	5	0	0	0	5	0	0	0
λ^3	0	0	0	5	0	0	0	5	0	0	0	0
λ^4	0	0	0	0	0	0	0	0	5	0	0	5

Tabela 5: Valores no vector λ^3 após a execução da primeira iteração do código identificado como "ITERATIVE STEP" da versão original do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) .

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
λ^3	0	0	0	0	5	0	0	0	10	0	0	0

Tabela 6: Valores no vector λ^3 após a execução da segunda iteração do código identificado como "ITERATIVE STEP" da versão original do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) .

do LSR1 ($w_{13}+\phi_1 = 5+10$ – passo 4 do algoritmo **LOCAL_EDGE_DISJOINT**(s,t)) é superior à etiqueta actual do LSR3 ($\phi_3 = 10$) esta não é alterada bem como o não é o predecessor do LSR3, porém nos cálculos subsequentes λ^3 é utilizado após ter sido incorrectamente (indevidamente) alterado (ver Tabela 6).

Este problema foi resolvido, no algoritmo corrigido, utilizando uma variável auxiliar para guardar temporariamente o vector λ^j no algoritmo **ALT_PATH_COST**(k,j) (linha 6), e actualizando o vector λ^j apenas no algoritmo **LOCAL_EDGE_DISJOINT**(s,t) (linha 12) quando é alterada a etiqueta e o predecessor do nó j.

3.2.4 Correcção 4

Para o exemplo seguinte utilizamos a mesma rede e tentamos estabelecer, com o algoritmo original (com todas as correcções anteriores feitas), o mesmo pedido (entre o LSR0 e o LSR2 com 5 unidades de LB) usado no exemplo da correcção anterior.

O algoritmo etiqueta definitivamente o LSR4 a partir do LSR2, o que significa que o ramo LSR4-LSR2 pertence à árvore de caminhos mais curtos. Tendo sido escolhido LSR4-LSR3-LSR2 para caminho de protecção desse ramo. Por isso o vector λ^4 possui nessa altura os valores indicados da Tabela 7.

Posteriormente, LSR0 é etiquetado permanentemente pelo algoritmo a partir de LSR4 ($k = 4$). O algoritmo escolheu para caminho de protecção do ramo LSR0-LSR4 o caminho LSR0-LSR1-LSR2 ($u = 2$). O vector λ^0 possui, no final da execução do algoritmo, os valores indi-

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
λ^4	0	0	0	0	0	0	0	0	5	0	0	5

Tabela 7: Valores no vector λ^4 quando o LSR4 é etiquetado permanentemente.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-0**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
λ^0	5	0	0	5	0	0	0	0	0	0	0	0

Tabela 8: Valores no vector λ^0 quando o LSR0 é etiquetado permanentemente.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-0**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
λ^0	5	0	0	5	0	0	0	0	5	0	0	5

Tabela 9: Valores que o vector λ^0 deveria ter quando LSR0 é etiquetado permanentemente.

cados da Tabela 8, pois λ^0 foi calculado somando os custos dos dois ramos do caminho de protecção nas posições respectivas de λ^2 (este vector possui zeros em todas as posições uma vez que corresponde ao nó destino).

No entanto, no final da execução do algoritmo, o vector λ^0 deveria possuir os valores indicados da Tabela 9, uma vez que o vector λ^0 deve guardar a quantidade de LB necessária para os RP de todos os ramos do AP, para o pedido actual.

Este problema foi resolvido, no algoritmo corrigido, utilizando λ^k em vez de λ^u no passo 8 do sub-algoritmo **SHORT_PRED_PATH**(k, u, j). Para o exemplo em causa, esta correcção implicava que o cálculo de λ^0 tinha em consideração o vector λ^4 em vez do vector λ^2.

3.2.5 Alteração 1

Para este exemplo acrescentámos também o ramo LSR3-LSR0 (com capacidade 250) à rede apresentada na Figura 1. Começámos por estabelecer um pedido entre o LSR3 (origem) e o LSR4 (destino) com 5 unidades de LB. De seguida foi estabelecido, com o algoritmo alterado e com o algoritmo original (este com todas as Correcções feitas), o pedido entre o LSR0 (origem) e o LSR2 (destino) com 60 unidades de LB, os resultados respectivos são apresentados na Tabela 10. Regista-se que no exemplo apresentado a alteração 3 não foi considerada no algoritmo modificado, no entanto se ela tivesse sido considerada o resultado final, em termos de AP e RP, seria exactamente o mesmo.

A Tabela 11 corresponde ao estado da rede após o estabelecimento do primeiro pedido.

As Figuras 2 e 3 ilustram as quatro iterações do código identificado como "ITERATIVE STEP" da versão original do algoritmo **LOCAL_EDGE_DISJOINT**(s, t) e as quatro iterações

Pedido	*Origem-Destino*	*LB*	AP	RP	Algoritmo
1º	*LSR3-LSR4*	*5*	LSR3-LSR4	LSR3-LSR0-LSR4	-
2º	*LSR0-LSR2*	*60*	LSR0-LSR1-LSR2	LSR0-LSR3-LSR2, LSR1-LSR3-LSR2	Alterado
2º	*LSR0-LSR2*	*60*	LSR0-LSR3-LSR2	LSR0-LSR4-LSR2, LSR3-LSR1-LSR2	Original

Tabela 10: Pedidos considerados no exemplo para a Alteração 1.

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
F_{jk}	0	0	0	0	0	0	0	0	0	5	0	0
B_{nm}	0	0	5	0	0	0	5	0	0	0	0	0
R_{nm}	250	250	70	250	250	250	245	250	250	245	250	250

Tabela 11: Estado da rede após o estabelecimento do primeiro pedido.

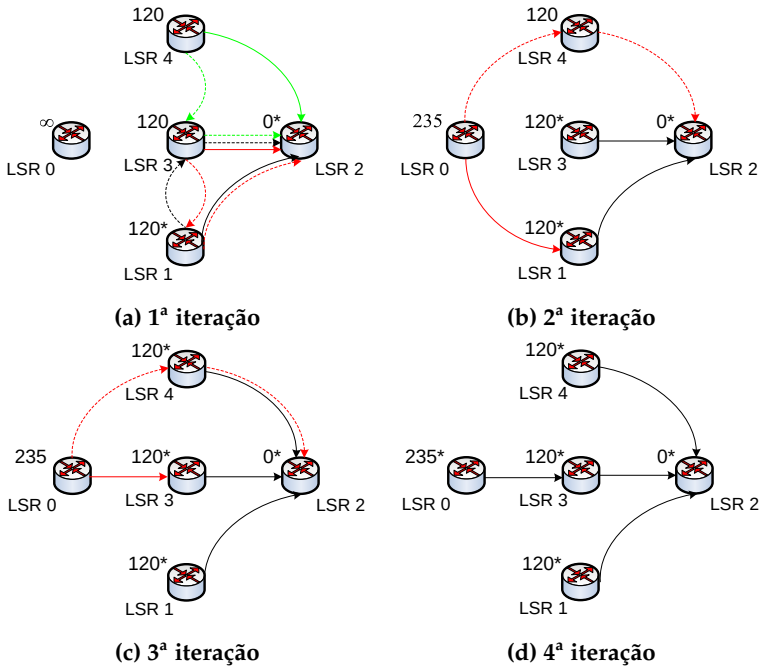

Figura 2: Iterações principais do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) - versão original.

do ciclo "WHILE" (linha 4 a linha 17) da versão alterada do algoritmo **LOCAL_EDGE_DISJOINT**(s,t), respectivamente. Nessas figuras, em cada uma das iterações, os ramos candidatos a entrar na árvore de caminhos mais curtos são representados com uma linha contínua e os caminhos de protecção escolhidos para esses ramos são representados por linhas descontínuas. Sendo que, o caminho de protecção e o ramo protegido por este possuem a mesma cor, e a cor preta é usada para o próximo ramo a entrar na árvore de caminhos mais curtos. Para simplificar as figuras, em cada iteração apenas se mantêm das iterações anteriores a representação dos ramos que fazem parte da árvore.

Ambos os algoritmos começam por etiquetar definitivamente o nó 2 com etiqueta zero (e todos os outros com etiquetas ∞) após o que re-etiquetam temporariamente os nós adjacentes a esse (nós 1, 3 e 4), como é visível nas Figuras 2(a) e 3(a). Com o algoritmo alterado as etiquetas temporárias obtidas são 180 para os três nós, enquanto que com o algoritmo original são 120 (cada um destes valores é igual ao correspondente anterior menos a LB do pedido em causa). Ambos os algoritmos etiquetam definitivamente o nó 1 após o que re-etiquetam temporariamente o nó 0 (o nó 3 embora adjacente mantém o valor), no algoritmo original passa a ter o valor 235 (etiqueta de LSR1 mais o custo da protecção do ramo

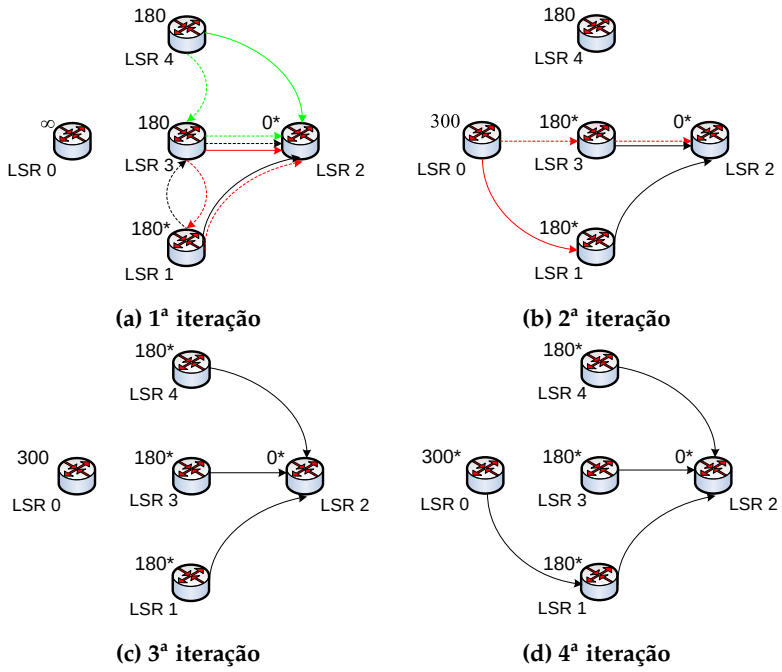

Figura 3: Iterações principais do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) - versão alterada.

LSR0-LSR1) enquanto que no alterado é re-etiquetado para 300, 180 (etiqueta de LSR1) + 60 (custo da protecção do ramo LSR0-LSR1) + 60 (LB do pedido). Tanto o algoritmo original como o alterado etiquetam definitivamente o LSR3, como é ilustrado nas Figuras 2(b) e 3(b), respectivamente. O algoritmo original na terceira iteração, ilustrada na Figura 2(c), re-etiqueta o LSR0 com 235, 120 (etiqueta de LSR3) + 115 (custo da protecção do ramo LSR0-LSR3). Note que o LSR0 foi re-etiquetado, desnecessariamente, com o mesmo valor que já possuía, sendo isso resultado da condição, $(\phi_j \geq w_{kj} + \phi_k)$, no passo 4 do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) . O LSR4 mantém a etiqueta tornando-se essa definitiva. Na mesma iteração o algoritmo alterado (ver Figura 3(c)) não efectua nenhuma alteração no valor das etiquetas mas torna definitiva a etiqueta do LSR4. Na última iteração tanto o algoritmo original (ver Figura 2(d)) como o algoritmo alterado (ver Figura 3(d)) apenas tornam definitiva a etiqueta de LSR0. Estes resultados confirmam as escolhas apresentadas na Tabela 10 para os AP e RP. Os estados da rede, após o estabelecimento deste pedido, são representados nas Tabelas 13 e 14, respectivamente para a versão original e alterada do algoritmo. Neste exemplo, embora ambos os algoritmos tenham partido do mesmo estado da rede, é visível que, para satisfazer o 2º pedido, a LB de protecção é superior na versão original. Na solução do algoritmo alterado o ramo LSR3-LSR2 é utilizado tanto para protecção do ramo LSR1-LSR2 como para protecção do ramo LSR0-LSR1. Este tipo de partilha não ocorreu no algoritmo original porque na segunda iteração quando re-etiquetou temporariamente o LSR0, o custo da protecção do ramo LSR0-LSR1 115 (55 do ramo LSR0-LSR4 (recorrendo a partilha inter-pedido) +60 do ramo LSR4-LSR2) não teve em consideração a informação mais recente relativa à partilha intra-pedido, utiliza o λ^2 em vez do λ^1 que é mais actualizado (ver Tabela 12), no cálculo de δ_{mn} a partir do qual se obtém os custos, l_{mn}, dos ramos no algoritmo **SHORT_PRED_PATH**(k,u,j)). Como na versão alterada é utili-

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
λ^1	0	0	0	0	60	0	0	0	60	0	0	0
λ^2	0	0	0	0	0	0	0	0	0	0	0	0

Tabela 12: Valores nos vectores λ^1 e λ^2 na 2ª iteração do algoritmo **LOCAL_EDGE_DISJOINT**(s, t).

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
F_{jk}	0	60	0	0	0	0	0	0	60	5	0	0
B_{nm}	0	0	60	60	0	0	5	60	0	0	60	0
R_{nm}	250	190	15	190	250	250	245	190	190	245	190	250

Tabela 13: Estado da rede após o estabelecimento do segundo pedido - versão original.

zada a informação do λ^1 este consegue partilha intra-pedido no ramo LSR3-LSR2 (este ramo já estava a ser usado para protecção do ramo LSR1-LSR2).

O algoritmo mesmo com as alterações mantém-se uma heurística "greedy" uma vez que as escolhas inicias podem impedir conseguir uma maior partilha no futuro. Como exemplo podemos utilizar o caso anterior. Vamos considerar que no segundo pedido, na primeira iteração, o algoritmo escolhia proteger o ramo LSR3-LSR2 com o caminho LSR3-LSR4-LSR2 (neste caso este caminho tem o mesmo custo que o que foi escolhido pelo algoritmo, embora isto não seja sempre condição necessária, ou seja pode mesmo haver situações em que a escolha inicial de um caminho de custo superior poderá conduzir a um consumo de LB final inferior). Se isso tivesse acontecido então na segunda iteração o LSR0 seria re-etiquetado a partir do LSR3 e não do LSR1 como aconteceu, pois neste caso estaria a usar simultaneamente partilha intra-pedido e inter-pedido. O estado da rede após o estabelecimento do segundo pedido seria o apresentado na Tabela 15, o que mostra que a LB total necessária para o segundo pedido foi de 295 unidades (ver também Tabela 11) enquanto que para o mesmo pedido tinha anteriormente sido 300 unidades (ver Tabela 11 e Tabela 14). Isto aconteceu porque a escolha da protecção do ramo LSR3-LSR2 foi feita sem ter em consideração as futuras possibilidades de partilha inter-pedido.

3.2.6 Alteração 2

A sequência de pedidos iniciais bem como os caminhos obtidos, para esses pedidos, são apresentados na Tabela 16. De novo regista-se que no exemplo apresentado a alteração 3

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-0	3-1	3-2	3-4	4-2	4-3
F_{jk}	60	0	0	60	0	0	0	0	0	5	0	0
B_{nm}	0	60	5	0	60	0	5	0	60	0	0	0
R_{nm}	190	190	70	190	190	250	245	250	190	245	250	250

Tabela 14: Estado da rede após o estabelecimento do segundo pedido - versão alterada.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-0**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	60	0	0	0	0	0	0	60	5	0	0
B_{nm}	0	0	60	0	0	0	5	0	0	60	60	0
R_{nm}	250	190	15	250	250	250	250	245	190	185	190	250

Tabela 15: Estado da rede após o estabelecimento do segundo pedido - com escolha de caminho de protecção alternativo.

Pedido	Origem-Destino	LB	AP	RP
1º	LSR3-LSR2	5	LSR3-LSR2	LSR3-LSR1-LSR2
2º	LSR0-LSR4	60	LSR0-LSR4	LSR0-LSR3-LSR4
3º	LSR0-LSR3	80	LSR0-LSR3	LSR0-LSR1-LSR3
4º	LSR1-LSR2	80	LSR1-LSR2	LSR1-LSR3-LSR2

Tabela 16: Pedidos considerados no exemplo para a Alteração 2.

não foi considerada no algoritmo modificado, no entanto se ela tivesse sido considerada o resultado final, em termos de AP e RP, seria exactamente o mesmo.

As Tabelas 17, 18, 19 e 20 correspondem ao estado após o estabelecimento do primeiro pedido, do segundo pedido, do terceiro pedido e do quarto pedido, respectivamente.

Vamos considerar que o quinto pedido é entre o LSR0 e o LSR2 com 80 de LB e vamos mostrar, para esse pedido, quais são as escolhas feitas para o AP e para os RP, pelos algoritmos alterado e original (com a alteração anterior e todas as correcções feitas). As Figuras 4 e 5 ilustram as quatro iterações do código identificado como "ITERATIVE STEP" da versão original do algoritmo **LOCAL_EDGE_DISJOINT**(s, t) e as quatro iterações do ciclo "WHILE" (linha 4 a linha 17) da versão alterada do algoritmo **LOCAL_EDGE_DISJOINT**(s, t), respectivamente. Nessas figuras, em cada uma das iterações, os ramos candidatos a entrar na árvore de caminhos mais curtos são representados com uma linha contínua e os caminhos de protecção escolhidos para esses ramos são representados por linhas descontínuas. Sendo que, o caminho de protecção e o ramo protegido por este possuem a mesma cor, e a cor preta é usada para o próximo ramo a entrar na árvore de caminhos mais curtos. Para simplificar as figuras, em cada iteração apenas se mantêm das iterações anteriores a representação dos ramos que fazem parte da árvore.

Ambos os algoritmos começam por etiquetar definitivamente o nó 2 com etiqueta zero (e todos os outros com etiquetas ∞) após o que re-etiquetam temporariamente os nós adjacentes a esse (nós 1, 3 e 4), como é visível nas Figuras 4(a) e 5(a). Com o algoritmo alterado as eti-

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	0	0	0	0	0	0	5	0	0	0
B_{nm}	0	0	0	5	0	0	5	0	0	0	0
R_{nm}	250	250	75	245	250	250	245	245	250	250	250

Tabela 17: Estado da rede após o estabelecimento do primeiro pedido.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	0	60	0	0	0	0	5	0	0	0
B_{nm}	0	60	0	5	0	0	5	0	60	0	0
R_{nm}	250	190	15	245	250	250	245	245	190	250	250

Tabela 18: Estado da rede após o estabelecimento do segundo pedido.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	80	60	0	0	0	0	5	0	0	0
B_{nm}	80	60	0	5	80	0	5	0	60	0	0
R_{nm}	170	110	15	245	170	250	245	245	190	250	250

Tabela 19: Estado da rede após o estabelecimento do terceiro pedido.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	80	60	80	0	0	0	5	0	0	0
B_{nm}	80	60	0	5	80	0	5	80	60	0	0
R_{nm}	170	110	15	165	170	250	245	165	190	250	250

Tabela 20: Estado da rede após o estabelecimento do quarto pedido.

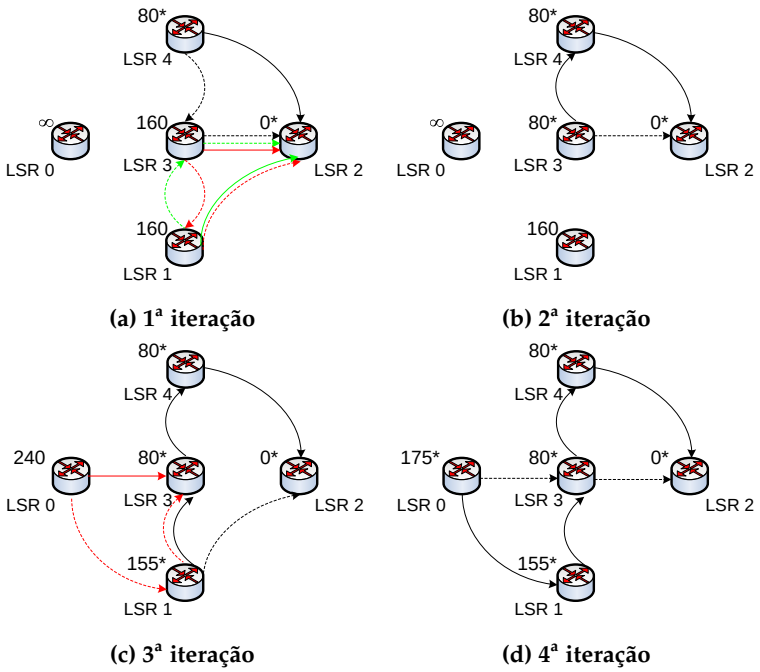

Figura 4: Iterações principais do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) - versão original.

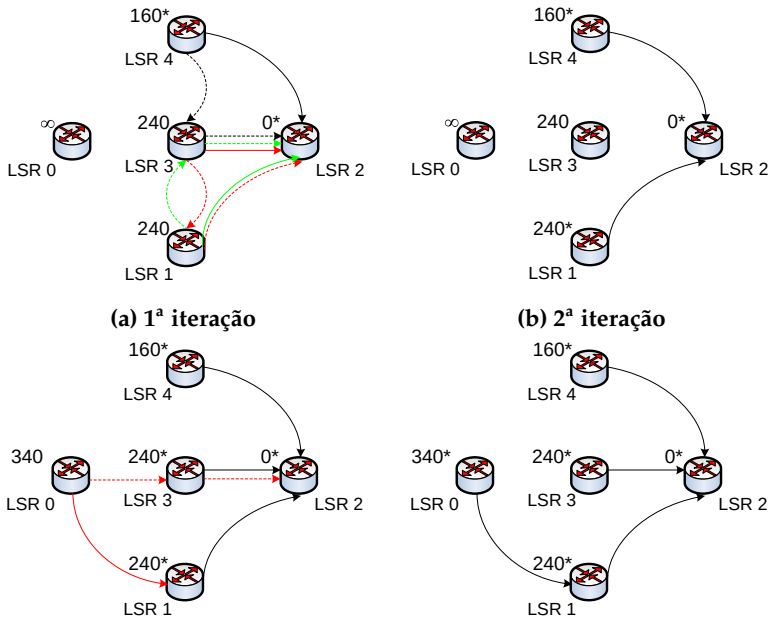

Figura 5: Iterações principais do algoritmo **LOCAL_EDGE_DISJOINT**(s,t) - versão alterada.

quetas temporárias obtidas são 240, 240 e 160, respectivamente para os nós 1, 3 e 4, enquanto que com o algoritmo original são 180, 180 e 80 (cada um destes valores é igual ao correspondente anterior menos a LB do pedido). Ambos os algoritmos etiquetam definitivamente o nó 4 após o que re-etiquetam temporariamente o nó 3 (o nó 0 embora também seja adjacente não é re-etiquetado porque não possui LB livre suficiente para o pedido em causa), no algoritmo original passa a ter o valor 80 (etiqueta de LSR4 mais o custo da protecção do ramo LSR3-LSR4) enquanto que no alterado mantém o valor 240; pois o valor que obtém é também 240[1] (160 (etiqueta de LSR4) $+$ 0 (custo da protecção do ramo LSR3-LSR4) $+$ 80 (LB do pedido)). O algoritmo original etiqueta definitivamente o LSR3 e o algoritmo alterado o LSR1 (embora também pudesse ter sido etiquetado definitivamente o LSR3[2]) como é ilustrado nas Figuras 4(b) e 5(b), respectivamente. O algoritmo original na terceira iteração, ilustrada na Figura 4(c), re-etiqueta o LSR1 com 155, 80 (etiqueta de LSR3) $+$ 75 (custo da protecção do ramo LSR1-LSR3), e o LSR0 com 240 ($80 + 160$) e torna a etiqueta de LSR1 definitiva. Na mesma iteração o algoritmo alterado (ver Figura 5(c)) re-etiqueta o LSR0 com 340, 240 (etiqueta de LSR1) $+$ 20 (custo da protecção do ramo LSR0-LSR1)$+$ 80 (LB do pedido) e torna definitiva a etiqueta de LSR3. Na última iteração o algoritmo original (ver Figura 4(d)) re-etiqueta o LSR0 com 175, 155 (etiqueta de LSR1) $+$ 20 (custo da protecção do ramo LSR0-LSR1), e torna a sua etiqueta definitiva. O algoritmo alterado apenas torna definitiva a etiqueta de LSR0 (ver Figura 5(d)).

[1]Como o novo valor é igual ao anterior – valores alternativos – o algoritmo poderia passar a utilizar o novo, o que significava que o LSR3 passava e ser etiquetado a partir de LSR4, como na versão original, no entanto o resultado final em termos de AP e RP seria o mesmo se em vez de manter o valor anterior se passasse a utilizar o novo.

[2]No entanto, também aqui o resultado final em termos de AP e RP seria o mesmo, ou seja é independente qual dos dois é etiquetado definitivamente em primeiro lugar.

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-1	3-2	3-4	4-2	4-3
F_{jk}	80	80	60	80	80	0	0	5	80	80	0
B_{nm}	80	80	0	80	80	0	5	80	60	0	80
R_{nm}	170	110	15	165	170	250	245	165	190	250	250

Tabela 21: Estado da rede após o estabelecimento do quinto pedido - versão original.

Ramo	0-1	0-3	0-4	1-2	1-3	2-1	3-1	3-2	3-4	4-2	4-3
F_{jk}	80	80	60	160	0	0	0	5	0	0	0
B_{nm}	80	80	0	5	160	0	5	160	60	0	0
R_{nm}	90	90	15	85	90	250	245	85	190	250	250

Tabela 22: Estado da rede após o estabelecimento do quinto pedido - versão alterada.

Quando se utiliza o algoritmo alterado, este escolhe para AP o caminho LSR0-LSR1-LSR2, para RP do ramo LSR1-LSR2 o caminho LSR1-LSR3-LSR2 e para RP do ramo LSR0-LSR1 o caminho LSR0-LSR3-LSR2. Quando se utiliza o algoritmo original, este escolhe para AP o caminho LSR0-LSR1-LSR3-LSR4-LSR2, para RP do ramo LSR4-LSR2 o caminho LSR4-LSR3-LSR2, para RP do ramo LSR3-LSR4 o caminho LSR3-LSR2, para RP do ramo LSR1-LSR3 o caminho LSR1-LSR2 e para RP do ramo LSR0-LSR1 o caminho LSR0-LSR3-LSR2. Neste exemplo, embora ambos os algoritmos tenham partido do mesmo estado da rede, com o algoritmo alterado o AP obtido é formado por dois ramos enquanto que com o algoritmo original o AP obtido é formado por quatro ramos. As razões destas escolhas são o algoritmo original minimizar o custo dos RP e o algoritmo alterado minimizar o custo total do AP e dos RP.

Os estados da rede, após o estabelecimento deste pedido, são representados nas Tabelas 21 e 22, respectivamente para a versão original e alterada do algoritmo. É visível que, para satisfazer o 5º pedido, a LB de protecção aumentou 180 na versão alterada e apenas 175 na versão original, no entanto a quantidade LB necessária para os AP foi de 160, 80 (LB do pedido) * 2 (número de ramos do AP), na versão alterada e de 320, 80 (LB do pedido) * 4 (número de ramos do AP), na versão original, o que mostra que neste exemplo a LB total para satisfazer o pedido foi superior na versão original.

3.2.7 Alteração 3

Para simplificar o exemplo vamos considerar que na rede da Figura 1 a capacidade dos ramos LSR0-LSR1 e LSR4-LSR2 é 50 unidades em vez das 250 lá indicadas. Nesta rede, tentamos estabelecer os pedidos apresentados na Tabela 23.

Na Tabela 24 é apresentado o estado da rede após o estabelecimento do primeiro pedido. Na determinação dos caminhos para o segundo pedido (Tabela 23), quando o algoritmo **SHORT_PRED_PATH**(k, u, j) procura um RP para o ramo LSR0-LSR3 ($j = 0$ e $k = u = 3$) começa por determinar os valores dos vectores δ e l.

Na versão original (este com todas as Correcções feitas) os valores obtidos são apresentados

Pedido	Origem-Destino	LB
1º	LSR0-LSR3	10
2º	LSR0-LSR3	60

Tabela 23: Pedidos considerados no exemplo para a Alteração 3.

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
F_{jk}	0	10	0	0	0	0	0	0	0	0	0
B_{nm}	10	0	0	0	10	0	0	0	0	0	0
R_{nm}	40	240	75	250	240	250	250	250	250	50	250

Tabela 24: Estado da rede após o estabelecimento do primeiro pedido (ver Tabela 23).

na Tabela 25 (determinam-se facilmente a partir da Tabela 24), que mostra que foi atribuído um custo "Infinito" a todos os ramos com excepção do ramo LSR1-LSR3. Com esses valores para os custos de utilização desses ramos o algoritmo de Dijkstra (código da linha 3 à linha 14) não encontra nenhum RP para protecção do ramo LSR0-LSR3, como é óbvio. Repare que, os valores de δ_{mn} são todos 70 ($F_{kj} + b$ = 60 + 10) e portanto maiores que a LB do pedido (60 unidades), excepto para os ramos LSR0-LSR1 e LSR1-LSR3 uma vez que nestes o valor de B_{mn} é não nulo (pois estes são os ramos usados na protecção do primeiro pedido). Tendo sido essa a razão para quase a totalidade dos valores "Infinito" dos custos. O que levou a que o pedido tenha sido rejeitado apesar de haver LB disponível para o estabelecer.

Este problema é evitado, na versão modificada, através da alteração da atribuição a δ_{mn} (linha 1). Deste modo, o custo de um ramo com capacidade residual não inferior à LB do pedido (b) será quando muito igual a b (com excepção do ramo que se está a tentar proteger).

No exemplo, mas agora com a versão modificada, na determinação dos caminhos para o segundo pedido, quando o algoritmo **SHORT_PRED_PATH**(k, u, j) procura um RP para o ramo LSR0-LSR3 ($j = 0$ e $k = u = 3$) os valores que obtém para os vectores δ e l são apresentados na Tabela 26. O que resulta na escolha do caminho LSR0-LSR3 para AP e do caminho LSR0-LSR4-LSR3 para RP.

4 Proposta para reserva exacta

Em [7, 11, 14] foram propostos métodos que, no contexto da protecção global, permitem fazer reserva mínima e libertação máxima de LB protecção, numa rede com o modelo PI.

Tal como proposto em [14] para cada ramo $e = (u, v)$ deve ser conhecido o conjunto completo de ramos (i, j) (dos caminhos activos) e a quantidade de LB em (i, j) que é protegida pelo ramo (u, v) – $P_{\mathcal{B}}(e)$ em [14].

Dado um caminho activo e os seus caminhos de protecção, considere os pares formados por cada ramo de protecção (u, v) e o conjunto de ramos por ele protegido P_{uv}. O método que permite fazer reserva mínima de LB de protecção proposto em [14] para protecção global, pode ser aplicado a cada um desses pares tomando o ramo (u, v) como um caminho de pro-

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
δ_{nm}	60	70	70	70	60	70	70	70	70	70	70
l_{nm}	Inf.	Inf.	Inf.	Inf.	60	Inf.	Inf.	Inf.	Inf.	Inf.	Inf.

Tabela 25: Valores dos vectores δ e l numa invocação de **SHORT_PRED_PATH**(k, u, j) na versão original do algoritmo, no estabelecimento do segundo pedido (ver Tabela 23).

Ramo	**0-1**	**0-3**	**0-4**	**1-2**	**1-3**	**2-1**	**3-1**	**3-2**	**3-4**	**4-2**	**4-3**
δ_{nm}	60	60	60	60	60	60	60	60	60	60	60
l_{nm}	Inf.	Inf.	60	60	60	60	60	60	60	Inf.	60

Tabela 26: Valores dos vectores δ e l numa invocação de **SHORT_PRED_PATH**(k, u, j) na versão modificada do algoritmo, no estabelecimento do segundo pedido (ver Tabela 23).

tecção global *virtual*, que protege o caminho activo *virtual* formado pelos ramos em P_{uv} (o qual em geral não coincidirá com nenhum caminho topológico). O mesmo procedimento pode ser utilizado para libertar caminhos de protecção (quando o caminho activo correspondente for libertado). Um aspecto relevante desta aproximação é permitir que coexistam, numa mesma rede, esquemas de protecção local e global partilhando LB de protecção. Isto é possível porque ambos os métodos requerem o mesmo tipo de informação e podem partilhar as estruturas de dados que mantêm essa informação e por conseguinte as reservas.

A utilização do modelo (preparado para protecção global) num contexto de protecção local, implica em primeiro lugar obter o conjunto, L_{RP}, de todos os ramos pertencentes aos caminhos de protecção do caminho activo do pedido corrente. Em segundo lugar, dado L_{RP}, é preciso, para cada $(u, v) \in L_{RP}$, obter o conjunto, P_{uv}, dos ramos do caminho activo que são protegidos por (u, v). Seguidamente o modelo de reserva exacta para protecção global, já conhecido, pode ser aplicado a cada caminho de protecção global *virtual*, o ramo (u, v) ($\forall (u, v) \in L_{RP}$), considerando que este protege o caminho activo *virtual* definido pelo conjunto P_{uv}. De notar que a LB adicional que o ramo (u, v) passa a proteger será igual à LB do pedido seja qual for o número de ramos que o caminho *activo virtual* possua.

Na Figura 6 é apresentada a rede usada para ilustrar o método proposto nesta secção. A Figura 7 descreve o caminho activo e os caminhos de protecção, através das respectivas sequências de ramos. Com base nesta figura é possível obter o conjunto de todos os ramos de protecção, L_{RP}. Para cada ramo, $(u, v) \in L_{RP}$, é obtido o conjunto dos ramos protegidos, P_{uv}, como se pode ver na Figura 8. Com base nesta informação pode então ser aplicado o método em [14] a cada par de caminhos virtuais (P_{uv}, (u, v)), $\forall (u, v) \in L_{RP}$. Por exemplo, quando (u, v) é igual a (LSR5,LSR3), $P_{uv} = \{(LSR1, LSR2), (LSR2, LSR3)\}$.

5 Resultados Simulacionais

Em **SHORT_PRED_PATH**(k, u, j) (passo 2) o teste $\delta_{mn} \leq b$ poderá originar a rejeição de pedidos apesar de existir LB suficiente para os satisfazer. Este comportamento será ilustrado numa rede com capacidade infinita, em que esta condição leva a uma taxa de rejeição extre-

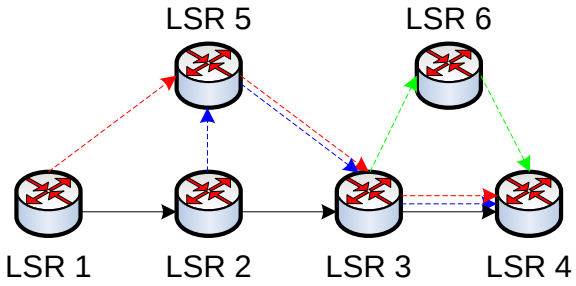

Figura 6: Rede para ilustrar a estrutura usada na reserva exacta.

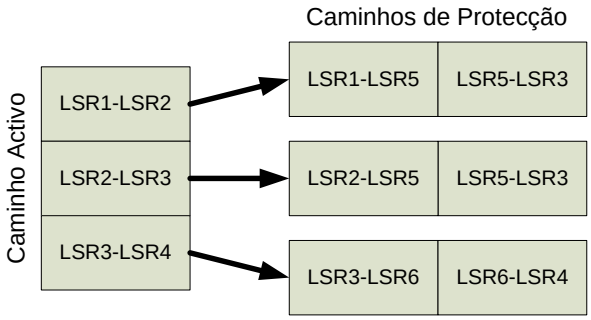

Figura 7: Estrutura activo-protecção.

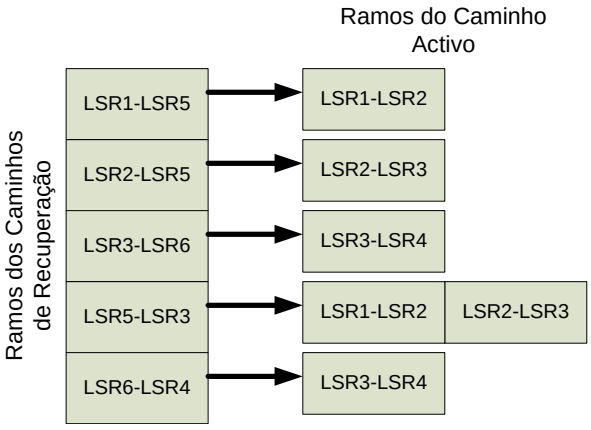

Figura 8: Estrutura protecção-activo.

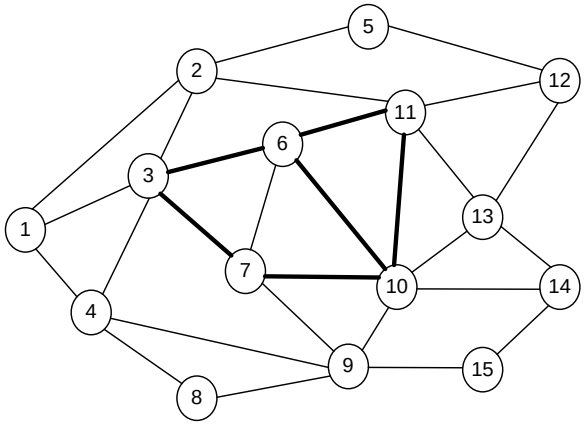

Figura 9: Topologia da rede [adaptada de 5].

mamente elevada! No contexto de reserva exacta esta restrição pode ser removida pois não influenciará a LB reservada para os RP.

A topologia de rede considerada aqui é apresentada na Figura 9. Esta rede é usada em vários trabalhos [5, 6, 7] dos autores do algoritmo em análise.

Trata-se de uma rede com 15 nós e 28 ramos bidireccionais (cada ramo não dirigido representa dois ramos dirigidos em sentidos opostos).

São apresentados resultados para duas situações: uma em que a capacidade dos ramos é infinita e outra em que todos os ramos têm capacidades finitas. Para a segunda situação, a capacidade de cada ramo fino é de 60 unidades e de cada ramo grosso é de 240 unidades, em ambos os casos em cada direcção. Em cada experiência chegam à rede pedidos (um de cada vez) de LB uniformemente distribuída no intervalo [6, 9], como em [7]. A origem e o destino dos LSP são escolhidos aleatoriamente. Foram executadas dez corridas para cada uma das experiências seguintes em que a reserva de LB de protecção foi feita de forma exacta de acordo com o método proposto na secção 4.

O algoritmo **LOCAL_EDGE_DISJOINT**(s,t) na sua forma original, com as correcções propostas será designado por LED; o algoritmo **LOCAL_EDGE_DISJOINT**(s,t) na sua forma original, com as correcções propostas e com o teste $\delta_{mn} \leq b$ removido no passo 2 de **SHORT_PRED_PATH**(k,u,j) será designado por LEDb; a versão modificada do algoritmo **LOCAL_EDGE_DISJOINT**(s,t), com todas as correcções e alterações propostas será designado por ILED.

Resumindo, vão ser apresentados resultados relativos às seguintes experiências:

1. Capacidade dos ramos infinita
   (a) Algoritmo original – LED
   (b) Algoritmo original com a supressão em **SHORT_PRED_PATH**(k,u,j) (passo 2) do teste $\delta_{mn} \leq b$ – LEDb
   (c) Algoritmo modificado – ILED

2. Capacidade dos ramos finita
   (a) Algoritmo original – LED
   (b) Algoritmo original com a supressão em **SHORT_PRED_PATH**(k, u, j) (passo 2) do teste $\delta_{mn} \leq b$ – LEDb
   (c) Algoritmo modificado – ILED

As medidas de desempenho seleccionadas foram:

- Comprimento médio dos AP.
- Número médio de ramos diferente utilizados pelos RP.
- Número médio de ramos do RP mais longo de cada AP.
- LB utilizada pelos AP.
- LB utilizada pelos RP.
- Percentagem de pedidos recusados (= probabilidade de bloqueio) versus número de LSP pedidos.

Para tal foi necessário registar:

- Número de pedidos tentados.
- Número de pedidos estabelecidos.
- Para cada pedido:
  - Comprimento do AP.
  - Largura de banda do AP.
  - Largura de banda dos RP.
  - Número total de ramos diferentes nos RP de cada AP.
  - Número de ramos do RP mais longo de cada AP.

Cada corrida de simulação, para o caso do tráfego incremental, termina quando forem solicitados 10001 pedidos. Na Figura 10 é claramente visível o efeito da remoção ou não da restrição $\delta_{mn} \leq b$: LEDb aceita todos os pedidos e LED apresenta uma taxa de aceitação de pedidos muito baixa. O comportamento de LED face aos pedidos solicitados pode ser visto em mais detalhe na Figura 11.

Numa rede de capacidade infinita e usando tráfego incremental não é necessário comparar, no que concerne à utilização de LB, ILED (ou LEDb) com LED, pois apenas mostraria que este último utiliza menos LB devido à rejeição de pedidos, como é visível nas Figuras 11 e 10. Assim sempre que for usada uma rede de capacidade infinita, ILED será comparado apenas com LEDb.

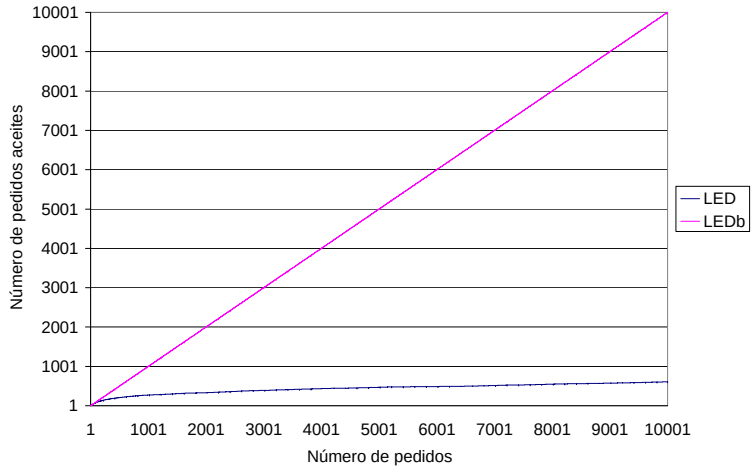

Figura 10: Pedidos aceites, usando tráfego incremental numa rede de capacidade infinita.

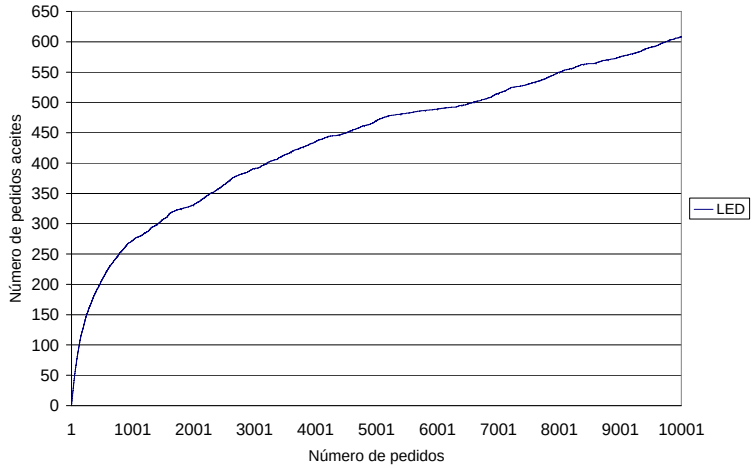

Figura 11: Pedidos aceites, usando tráfego incremental numa rede de capacidade infinita, para melhor visualização do comportamento de LED.

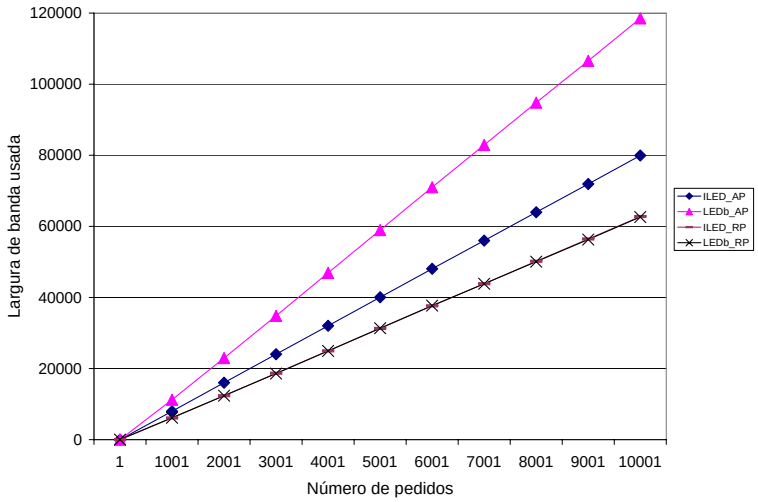

Figura 12: Largura de banda usada por AP e RP, usando tráfego incremental numa rede de capacidade infinita.

Na Figura 12 apresenta-se a LB usada por ILED e LEDb na rede com capacidade infinita usando tráfego incremental. Pode observar-se que ambos os algoritmos utilizam quantidade semelhantes de LB de protecção mas ILED apresenta melhor desempenho quanto à LB requerida pelos AP.

No final das corridas de simulação o comprimento médio dos AP é cerca de 2,3 para ILED e 3,5 para LEDb. O número médio de ramos diferentes necessários para proteger cada AP é 5 para ILED e 9 para LEDb. Por conseguinte ILED reserva menos LB do que LEDb porque utiliza AP mais curtos (com menor número de ramos) e embora ILED use nos RP menos ramos (diferentes) do que LEDb, ambos os algoritmos apresentam consumo identico de LB de protecção.

Para o caso do tráfego dinâmico, cada corrida de simulação termina após 10001 pedidos, depois de terminado um período de aquecimento com 2500 pedidos, o qual garante que foi atingido o regime de estacionaridade para todos os valores de tráfego usados. As chegadas seguem um processo de Poisson de intensidade λ, e a duração da ligações são exponencialmente distribuídas com valor médio igual a $1/\mu$. Foram igualmente utilizadas 10 corridas de simulação, assim como pedidos de LB uniformemente distribuídos no intervalo [6, 9]. Para avaliar o desempenho dos algoritmo em diferentes situações de carga tomou-se $\lambda/\mu = 100, 110, 120, 130, 140, 150$.

Os resultados na Tabela 27 mostram que ILED tem um bloqueio médio significativamente inferior a LEDb e LED. Por sua vez LEDb apresenta melhor desempenho do que LED o que vem confirmar a pertinência da remoção da restrição $\delta_{mn} \leq b$ no passo 2 de **SHORT_PRED_PATH**(k, u, j). ILED utiliza em média menos ramos nos AP, do que LEDb ou LED, como se pode ver na Tabela 28, o que tende a reduzir o tempo de propagação do tráfego na rede. Por outro lado o número de ramos diferentes utilizados por ILED nos RP é em média também inferior aos requeridos por LEDb e LED. Averiguou-se ainda qual número médio de ramos do RP mais longo de cada AP, tendo-se verificado que ILED apresenta consistentemente o menor valor das três versões do algoritmo.

λ/μ	100	110	120	130	140	150
ILED	0,2%	0,4%	0,7%	1,1%	1,8%	2,6%
LED	25,5%	27,3%	28,9%	29,4%	30,1%	32,2%
LEDb	2,9%	5,1%	7,7%	10,4%	13,0%	15,7%

Tabela 27: Bloqueio médio para $\lambda/\mu = 100, 110, 120, 130, 140, 150$.

λ/μ	100	110	120	130	140	150
ILED	2,4	2,4	2,4	2,4	2,4	2,4
LED	3,9	3,9	3,9	3,9	3,9	3,9
LEDb	3,8	3,8	3,8	3,8	3,8	3,7

Tabela 28: Número médio de ramos dos AP para $\lambda/\mu = 100, 110, 120, 130, 140, 150$.

Observamos que é frequente o algoritmo obter RP longos. Este aspecto poderá ser melhorado: (a) através da escolha do caminho com menor número de ramos, entre caminhos alternativos (de igual custo) no algoritmo **ALT_PATH_COST**(k, j); (b) permitindo em **SHORT_PRED_PATH**(k, u, j) a alteração da etiqueta e do predecessor de um nó, quando os custos são iguais, garantindo desta forma que o caminho obtido até ao nó j é um caminho de custo mínimo com o menor número de ramos possível.

6 Conclusões

Foram corrigidas algumas gralhas e imprecisões de um algoritmo de encaminhamento dinâmico com protecção local e foi proposta uma nova versão (mais eficiente) para esse algoritmo. Foi também apresentado um método para reserva mínima de LB de protecção num contexto da protecção local. Uma característica relevante deste método consiste em permitir a co-existência de esquemas de protecção local e global, partilhando indistintamente LB de protecção.

O desempenho do algoritmo original e da versão melhorada do algoritmo, utilizando reserva mínima de LB de protecção foi analisada usando tráfego incremental e dinâmico. Em ambos os casos, o número médio de ramos utilizado nos AP é menor em ILED do que em LEDb ou LED. Usando tráfego incremental, os resultados experimentais mostraram que a nova versão utiliza menos LB, nomeadamente nos AP. No estudo experimental utilizando tráfego dinâmico o bloqueio médio na rede foi significativamente menor quando se utiliza ILED. As conclusões anteriores relativamente ao número médio de ramos utilizado nos AP podem neste caso verificar-se também no comprimento médio do RP mais longo de cada AP.

As várias versões do algoritmo obtêm com frequência RP longos, embora tal seja menos visível em ILED. Foi sugerida uma estratégia para mitigar essa característica.

Em resumo, pode afirmar-se que as as alterações propostas tornam o algoritmo mais eficiente em termos de consumo de LB e também relativamente aos tempos de propagação do tráfego

λ/μ	100	110	120	130	140	150
ILED	6,2	6,1	6,0	6,0	6,0	6,0
LED	9,4	9,5	9,4	9,7	10,2	10,1
LEDb	9,4	9,7	9,7	9,8	10,0	9,9

Tabela 29: Número médio de ramos diferentes usados por todos os RP de um dado AP.

λ/μ	100	110	120	130	140	150
ILED	4,3	4,3	4,3	4,3	4,3	4,2
LED	6,8	7,0	7,1	7,3	8,0	7,9
LEDb	6,7	7,1	6,9	7,1	7,1	7,0

Tabela 30: Número médio de ramos do RP mais longo de cada AP.

nos AP. O algoritmo mesmo com as alterações mantém-se uma heurística "greedy" uma vez que as escolhas inicias podem impedir conseguir uma maior partilha no futuro. O algoritmo, apresentado em [6] é bastante complexo por esta razão pensamos que as correcções e as alterações que propusemos não lhe retiram mérito.

A Acrónimos

AP *Active Path*

CI *Complete Information*

FRR *Fast Reroute*

IP *Internet Protocol*

LB Largura de Banda

LSP *Label Switched Path*

MPLS *MultiProtocol Label Switching*

MSOD *Min-Sum with Ordered Dual cost links*

NI *No Information*

PI *Partial Information*

PLR *Point of Local Repair*

QoS *Quality of Service*

RP *Recovery Path*

Referências

[1] P.-H. Ho, J. Tapolcai, and H. T. Mouftah. On achieving optimal survivable routing for shared protection in survivable next-generation internet. *IEEE Transactions on Reliability*, 53(2):216–225, June 2004.

[2] D. W. Hong, C. S. Hong, and W.-S. Kim. A segment-based protection scheme for MPLS network survivability. In *Network Operation and Management Symposium (10th IEEE/IFIP)*, pages 1–4, 2006. ISBN 1-4244-1429.

[3] Luísa Jorge and Teresa Gomes. Survey of Recovery Schemes in MPLS Networks. In *Proceedings of the International Conference on Dependability of Computer Systems (DepCos 2006)*, Wroclaw University of Technology, Poland, May 2006. To be presented.

[4] M. Kodialam and T. V. Lakshman. Dynamic backup routing of network tunnel paths for local restoration in a packet network, Patent US 2002/0067693A1, June 2002.

[5] M. Kodialam and T. V. Lakshman. Dynamic Routing of Bandwidth Guaranteed Tunnels with Restoration. In *Proceedings of IEEE INFOCOM 2000*, pages 902–911, 2000.

[6] M. Kodialam and T. V. Lakshman. Dynamic Routing of Locally Restorable Bandwidth Guaranteed Tunnels using Aggregated Link Usage Information. In *Proceedings of IEEE INFOCOM 2001*, pages 376–385, April 2001.

[7] M. Kodialam and T. V. Lakshman. Restorable Dynamic Quality of Service Routing. *IEEE Communications Magazine*, 40(6):72–81, June 2002.

[8] M. Kodialam and T. V. Lakshman. Dynamic routing of restorable bandwidth-guaranteed tunnels using aggregated network resource usage information. *IEEE/ACM Transactions on Networking*, 11(3):399–410, 2003. ISSN 1063-6692. doi: http://dx.doi.org/10.1109/TNET.2003.813044.

[9] 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, October 7-10 2001.

[10] C.-L. Li, S. T. McCormick, and D. Simchi-Levi. Finding disjoint paths with different path costs: complexity and algorithms. *Networks*, 22:653–667, 1992.

[11] Y. Liu, D. Tipper, and P. Siripongwutikorn. Approximating optimal spare capacity allocation by successive survivable routing. In *INFOCOM 2001*, pages 699–708, 2001.

[12] H. T. Mouftah and P.-H. Ho. *Optical networks – arquitecture and survivability*. Kluwer Academic Publishers, 2003.

[13] Ping Pan, George Swallow, and Alia Atlas. Fast reroute extensions to RSVP-TE for LSP tunnels. IETF RFC 4090, May 2005.

[14] Chunming Qiao and Dahai Xu. Distributed Partial Information Management (DPIM) Schemes for Survivable Networks - Part I. In *Proceedings of IEEE INFOCOM 2002*, pages 302–311, 2002.

[15] V. Sharma, F. Hellstrand, B. Mack-Crane, S. Makam, K. Owens, C. Huang, J. Weil, B Cain, L. Anderson, B. Jamoussi, A. Chiu, and S Civanlar. Framework for multi-protocol label switching (MPLS)-based recovery. IETF RFC 3469, February 2003.

[16] J. W. Suurballe. Disjoint paths in networks. *Networks*, 4:125–145, 1974.

[17] Dahai Xu, Yang Chen, Yizhi Xiong, Chunming Qiao, and Xin He. On finding disjoint paths in single and dual link cost networks. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 705–715, 2004. doi: 10.1109/INFCOM.2004.1354541.