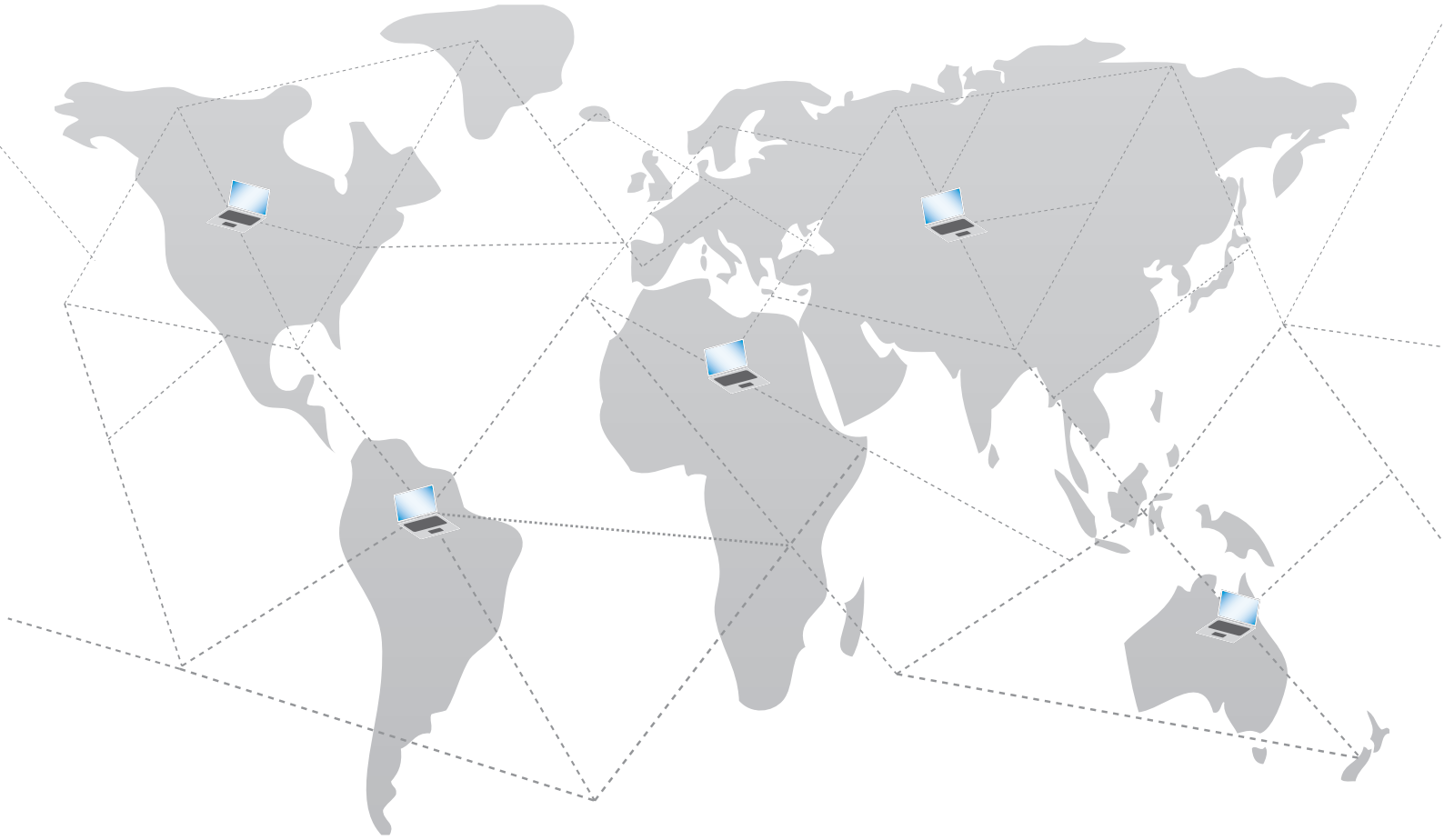


Sofia Marques Ferreira

Determinação do caminho mais curto e respectivo caminho de protecção

Dissertação submetida para a satisfação parcial dos requisitos do grau de Mestre em Engenharia Electrotécnica e de Computadores, Área de Especialização em Telecomunicações

Fevereiro de 2015

· U C ·

Universidade de Coimbra

Faculdade de Ciências e Tecnologia
Mestrado Integrado em Engenharia Electrotécnia e de Computadores

Determinação de um caminho que passe em elementos específicos da rede e respectivo caminho de protecção

Sofia Marques Ferreira

Dissertação submetida para a satisfação parcial dos requisitos do grau de Mestre em Engenharia Electrotécnica e de Computadores, Área de Especialização em Telecomunicações

Júri:
Presidente: Professora Doutora Lúcia Maria dos Reis Albuquerque Martins
Orientadora: Professora Doutora Teresa Martinez dos Santos Gomes
Vogal: Professora Doutora Rita Cristina Girão Coelho da Silva

Fevereiro de 2015

Este trabalho foi suportado pelo projecto QREN 23301 - Panorama II, cofinanciado pelo Fundo Europeu de Desenvolvimento Regional (FEDER), através do Programa Operacional de Competividade (POFC).

Aos meus pais, Carlos e Esmeralda

Agradecimentos

Quero agradecer a todas as pessoas que contribuíram para a concretização deste trabalho.

Agradeço à minha orientadora, Professora Doutora Teresa Martinez dos Santos Gomes por me ter proporcionado a realização desta dissertação, pela sua disponibilidade para esclarecer dúvidas e na revisão da dissertação, bem como pela sua competente orientação e perspicácia na superação de diversos obstáculos que surgiram ao longo do percurso.

Agradeço aos meus amigos pelo suporte e companheirismo ao longo destes meses.

Agradeço aos meus pais que com constante trabalho conseguiram reunir as melhores condições para que eu chegasse ao fim desta etapa e ao meu irmão por o ser.

Muito Obrigada!

Abstract

In routing with protection one seeks to obtain a pair of node or arc disjoint (loopless) paths, depending on the degree of desired protection. When this is not possible, a pair of paths as disjoint as possible may be used. The path that carries information in the absence of failure is referred to as the active path and the path that carries data in case of unavailability of the active path is called the protection path. A shortest loopless path may still have to satisfy the constraint of going through certain nodes and/or arcs. These nodes and/or arcs are called specific network elements.

Firstly, the problem of determining a path of minimal additive cost, which passes through specific elements, was studied - this problem will be referred to as the problem of the specific path (PCE – *Problema do Caminho Específico*). Secondly, the problem of routing with protection in which the active path (additive minimum cost) must pass in specific network elements was tackled - this problem is referred to as the problem of specific active path (PCAE – *Problema do Caminho Activo Específico*). Three different cases were considered: the obtained paths must be node disjoint (problem designated by PCAE-DN), or edge disjoint (problem designated by PCAE-DA), or maximally disjoint (PCAE-MD). Though, PCE problem is NP-hard, an exact resolution approach was considered in this work. It was expected that, with the additional constraint that a protection path must exist for the selected active path, it would be possible to solve the problem in reasonably sized networks.

In the literature references for solving the problem of determining a path of minimum additive cost in which specific elements are uniquely nodes, were found. Thus, a network transformation that allows to represent a specific edge through a specific node is proposed in this work.

Afterwards, the efficiency improvement of Ibaraki's algorithm was considered, and two variants are proposed. Both algorithms build candidate sub-paths for the active path that should include specific elements. A sub-path is only considered admissible if it can be protected. Therefore, it is expected that the resolution of PCAE-DN will be the most efficient because many candidate sub-paths will be discarded due to the inability to protect them. On the other hand PCAE-MD will always return a pair of paths (in the absence of solution it returns the protection path equal to the active path).

Computational experimental results are presented using eleven networks from the SNDlib library with the topology of real networks. An integer linear formulation was adapted to solve the PCE and to validate the solutions obtained by the algorithms for solving the PCAE-DN and PCAE-DA.

This work intended to be a first approach to the resolution of PCAE in which one would acquire the sensitivity for proposing, as future work, an effective heuristic aplicable in the context of larger networks.

Keywords: shortest path, specific nodes, specific arcs, protection path, disjoint paths, maximally disjoint paths.

Resumo

No encaminhamento com protecção procura-se obter um par de caminhos (sem ciclos) disjuntos nos nós ou nos arcos, dependendo do grau de protecção desejado. Quando isso não é possível pode ser permitido a utilização de um par de caminhos tão disjunto quanto possível. O caminho que transporta a informação na ausência de falhas designa-se por caminho activo e o caminho que transporta a informação em caso de indisponibilidade do caminho activo designa-se por caminho de protecção. Um caminho mais curto elementar pode ainda ter que satisfazer a restrição de passar por determinados nós e/ou arcos. Estes nós e/ou arcos são denominados de elementos específicos da rede.

Neste trabalho foi em primeiro lugar abordado o problema da determinação de um caminho, de custo aditivo mínimo, que passe em elementos específicos da rede – este problema será designado por Problema do Caminho Específico (PCE). Em seguida procurou-se resolver o problema do encaminhamento com protecção, em que o caminho activo (de custo aditivo mínimo) precisa de passar em elementos específicos da rede – este problema será designado por Problema do Caminho Activo Específico (PCAE). Foram tratados três casos diferentes: os caminhos obtidos devem ser disjuntos nos nós (problema designado por PCAE-DN), ou disjuntos nos arcos (problema designado por PCAE-DA), ou tão disjuntos entre si quanto possível (PCAE-MD). O problema PCE é NP-difícil. Contudo neste trabalho optou-se por uma abordagem de resolução exacta, pois esperava-se que mesmo com a restrição adicional da existência de um caminho disjunto, fosse possível resolver o problema em redes de dimensão razoável.

Na literatura foram encontradas apenas referências para a resolução do problema da determinação de um caminho, de custo aditivo mínimo em que os elementos específicos são exclusivamente nós. Assim, foi proposta uma transformação de rede que permite representar um arco específico através de um nó específico.

Em seguida, procurou-se melhorar a eficiência do algoritmo de Ibaraki, tendo sido propostas duas variantes. Ambos os algoritmos constroem sub-caminhos candidatos a caminhos activos que devem passar nos elementos específicos seleccionados. Um sub-caminho só é considerado admissível se puder ser protegido. Assim, espera-se que a resolução do problema PCAE-DN seja a mais eficiente uma vez que muitos sub-caminhos candidatos serão descartados devido à impossibilidade de os proteger. No extremo oposto, encontra-se o PCAE-MD que devolverá sempre um par de caminhos (na ausência de solução devolverá o caminho de protecção igual ao caminho activo).

Apresentam-se resultados de experimentação computacional, utilizando onze redes da biblioteca SNDlib com topologias de redes reais. Foi ainda adaptada uma formalização proposta para a resolução do PCE, com o intuito de confirmar as soluções obtidas pelos algoritmos implementados para a resolução dos problemas PCAE-DN e PCAE-DA.

Este trabalho pretendeu ser uma primeira abordagem à resolução do PCAE na qual se adquiriria a sensibilidade necessária para a proposta em trabalho futuro de uma heurística eficaz em redes de maior dimensão.

Palavras-chave: caminho mais curto, nós específicos, arcos específicos, caminho de protecção, caminhos disjuntos, caminho maximamente disjuntos.

Conteúdo

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

O problema que consiste na determinação do caminho mais curto é um problema intensamente estudado e com múltiplas aplicações em diversas áreas nomeadamente nas Telecomunicações. Um algoritmo especializado e amplamente conhecido é o algoritmo de Dijkstra que permite determinar o caminho mais curto entre um dado nó e todos os outros nós de uma rede, com custos não negativos associados aos arcos.

No âmbito desta dissertação pretende-se determinar o caminho mais curto elementar entre um par de nós, com a restrição adicional do caminho ser obrigado a visitar um conjunto de elementos obrigatórios (nós e/ou arcos). Este problema será designado por Problema do Caminho Específico (PCE). Por caminho mais curto elementar entende-se o menor caminho em termos de custo aditivo entre o nó origem e o nó destino que visita uma única vez os nós desse caminho.

Uma forma eficiente, do ponto de vista dos recursos utilizados, de garantir a recuperação de uma conexão ponto a ponto, em caso de falha isolada, é a utilização de um par de caminhos disjuntos entre os nós extremos da conexão. A protecção é uma abordagem proactiva que consiste no estabelecimento com o CA (aquele que transporta o tráfego na ausência de falha) de um ou mais caminhos de protecção (CPs) disjuntos. Quando ocorre uma falha o tráfego no CA é comutado para o CP (ou CPs), sem que haja a percepção de uma interrupção do serviço de comunicação.

O caminho de protecção global pode ser calculado de acordo com os seguintes critérios: disjunto nos nós, disjunto nos arcos, maximamente disjunto nos nós, maximamente disjunto nos arcos. Estes diferentes problemas, considerando sempre que o caminho activo tem de passar em elementos específicos, são designados por: Problema do Caminho Activo Específico com CP Disjunto nos Nós (PCAE-DN), Problema do Caminho Activo Específico com CP Disjunto nos Arcos (PCAE-DA), Problema do Caminho Activo Específico com CP Maximamente Disjunto nos Nós (PCAE-MDN) e Problema do Caminho Activo Específico com CP Maximamente Disjunto nos Arcos (PCAE-MDA), respectivamente.

Optou-se por abordar separadamente os problemas de determinação de pares de caminhos disjuntos e maximamente disjuntos uma vez que os algoritmos que o resolvem deveriam ter

desempenhos diferentes, como aliás se veio a verificar (ver análise de resultados no capítulo 4).

A pesquisa efectuada revelou informação escassa acerca deste problema e foram apenas encontrados quatro trabalhos que tratavam um problema semelhante àquele que se pretendia resolver, são eles: Saksena e Kumar em 1966 [1], Dreyfus em 1969 [2], Ibaraki em 1973 [3], e por fim, Andrade [4] em 2013.

Por ordem cronológica surge primeiro o trabalho de Saksena e Kumar [1] com um método que para uma dada rede procura resolver o problema do caminho mais curto não necessariamente elementar. Em 1969, Dreyfus vem afirmar [2] que o algoritmo proposto por Saksena e Kumar está errado e propõe que o mesmo problema seja resolvido através de uma redução ao problema do caixeiro-viajante. Mais tarde, surge o artigo de Ibaraki [3] sobre o mesmo assunto.

Ibaraki separa o problema do caminho mais curto em dois problemas. À semelhança de Saksena e Kumar considera o problema do caminho mais curto não necessariamente elementar e, por outro lado, o caminho mais curto elementar.

Para resolver este problema é proposto um algoritmo baseado em programação dinâmica e outro em *branch and bound*.

O método da programação dinâmica procura resolver o problema de optimização através da análise de uma sequência de problemas mais simples do que o problema original: a resolução de um problema original de X variáveis é caracterizado pela determinação de uma variável e pela resolução de um problema que possua uma variável a menos $(X-1)$. Este por sua vez é resolvido pela determinação de uma variável e pela resolução de um problema de $(X-2)$ variáveis e assim por diante. Como a intenção deste trabalho é a obtenção do caminho mais curto que seja permitido proteger, foi considerada mais promissora a abordagem da programação dinâmica que vai permitir a cada passo verificar a existência de um caminho de protecção.

Andrade [4] desenvolve duas formalizações para a determinação do caminho mais curto elementar que passa em nós específicos da rede, e apresenta resultados numéricos que ilustram a eficiência computacional na resolução de instâncias aleatórias do problema.

1.1 Motivação

A determinação do caminho mais curto entre um par origem e destino numa rede surge com frequência na área das Telecomunicações. No entanto, tal como já foi referido, na literatura poucas são as referências ao problema do caminho que passa em elementos específicos da rede, o que torna mais aliciante a sua resolução. Para além disso, o tema desta dissertação enquadra-se nos objectivos do projecto QREN 23301 Panorama-II, um consórcio liderado pela PT-Inovação e no qual a Universidade de Coimbra é um dos parceiros envolvidos.

1.2 Objectivos e organização da dissertação

Este trabalho foca-se na implementação de um algoritmo que permita resolver o problema do caminho mais curto que visita nós ou arcos específicos de uma dada rede e a determinação do respectivo caminho de protecção. Os objectivos inerentes a este trabalho subdividiram-se então nas seguintes tarefas:

1. Calcular o caminho mais curto com um conjunto de elementos (nós e/ou arcos) específicos;

2. Calcular o caminho mais curto com um conjunto de elementos (nós e/ou arcos) específicos para o qual é possível encontrar um caminho disjunto nos nós (caminho de protecção).

3. Calcular o caminho mais curto, que passa em elementos (nós e/ou arcos) específicos, para o qual é possível encontrar um caminho disjunto nos arcos: *solução de compromisso*, quando não é possível obter um caminho disjuntos nos nós.

4. Calcular o caminho mais curto, que passa em elementos (nós e/ou arcos) específicos e de um caminho de protecção maximamente disjunto nos nós ou nos arcos, consoante o grau de protecção desejado.

5. Análise do desempenho dos vários algoritmos implementados.

Esta dissertação é composta por 5 capítulos. No capítulo 2 faz-se uma apresentação dos conceitos necessários ao entendimento do problema em conjunto com a descrição do algoritmo proposto em [1] e respectiva crítica feita por Dreyfus [2] acompanhada da nossa análise. Segue-se a descrição do algoritmo de Ibaraki [3] e onde são apresentadas as diversas variantes propostas para a determinação do caminho mais curto com nós/arcos específicos com vista a melhorar os tempos de execução.

No capítulo 3 trata-se da determinação do caminho de protecção. Existem várias variantes para o cálculo deste caminho: disjunto nos nós, disjunto nos arcos, maximamente disjunto nos nós, maximamente disjunto nos arcos.

No capítulo 4 apresenta-se os resultados experimentais relativos aos algoritmos apresentados ao longo do capítulo 3, procurando realizar uma comparação crítica segundo o ponto de vista do desempenho e da complexidade computacional traduzidos nos tempo de execução.

No capítulo 5 apresentaremos as principais conclusões extraídas do trabalho realizado, terminando com algumas sugestões de trabalho futuro.

Capítulo 2

Caminho Mais Curto com Elementos Específicos

2.1 Conceitos

A representação de uma rede de telecomunicações pode ser feita através de um grafo que representa a topologia da rede. Sendo relevante associar um ou mais valores aos arcos de um grafo podemos colocar pesos ou custos como características adicionais dos arcos, ficando com um grafo pesado.

Definição 1. *Seja $G = (N, A, f)$ o grafo não dirigido representativo da rede tal que:*

a) N é o conjunto finito de elementos que se designam por nós.

$$N = \{n_1, n_2, ..., n_p\}$$

b) A é o conjunto finito de elementos que se designam por arestas ou arcos (não dirigidos):

$$A = \{(n_i, n_j) | n_i, n_j \in N \quad \text{e} \quad n_i \neq n_j\}$$

Se $(n_i, n_j) \in A$ diz-se que n_i e n_j são nós adjacentes.

c) Uma aresta (ou arco não dirigido) é representado por um par não ordenado de vértices, (n_i, n_j), sendo n_i e n_j quaisquer nós adjacentes $\in N$.

d) $f : A \to \mathbb{R}^+$ onde $f(n_i, n_j)$ é o custo da aresta $(n_i, n_j) \in A$;

Aos arcos são atribuídos pesos representativos, por exemplo, da capacidade, custo e da probabilidade de falha. Neste trabalho, a cada arco da rede está associado um valor que é o custo do arco que se considerou ser sempre um número real não negativo.

Os grafos surgem como modelos das mais variadas estruturas: as ruas de uma cidade formam um grafo cujas arestas são os troços entre cruzamentos ou rotundas; quando tratamos

de circulação rodoviária, há frequentemente ruas ou partes de ruas de sentido único que podem ser naturalmente consideradas como arcos com sentido associado, e outras com os dois sentidos. Um grafo misto é um modelo óbvio para estudos neste contexto.

Em geral é usada a palavra aresta quando se quer referir a ligação entre dois nós sem sentido associado e a palavra arco surge como sinónimo de arco dirigido, ou seja com sentido associado. Embora este trabalho se debruce sobre redes não dirigidas, haverá um exemplo em que se recorre ao uso de um grafo dirigido. Nesse grafo os arcos são dirigidos, ou seja, cada arco é formado por um par ordenado de vértices (n_i, n_j). Portanto diz-se que n_i é a cauda e n_j é a cabeça do arco.

A Teoria dos Grafos contém um enorme conjunto de definições associadas ao seu estudo, [5] e [6]. Aqui são enunciados alguns dos conceitos necessários à compreensão deste trabalho:

a) O grau de um nó é o número de nós que lhe são adjacentes.

b) Um grafo diz-se ligado, ou conexo, se entre quaisquer dois pontos existir um caminho. Caso contrário diz-se desconexo. Um caminho é uma sequência de nós adjacentes. Se não há nenhum nó repetido o caminho é elementar (isto é, sem ciclos). Se o nó origem e o destino do caminho coincidem e são um único nó repetido, o caminho designa-se por ciclo.

c) Um grafo formado por um só nó, ou seja, cujo conjunto de arcos ou arestas é vazio designa-se por trivial.

d) Um grafo dirigido é constituído por nós e arcos (dirigidos); um grafo não dirigido é constituído por nós e arestas (isto é, arcos não dirigidos) e um grafo misto possui arestas e arcos.

e) Arestas do tipo (a, a) chamam-se laços ou lacetes e grafos que os contenham são chamados *loop-graphs*. Quando pares surgem repetidos, chamam-se multilinhas, linhas múltiplas ou linhas paralelas ou, mais especificamente multiarestas, arestas múltiplas ou paralelas e multiarcos, arcos múltiplos ou paralelos, estas duas últimas para quando estamos a usar a definição geral de aresta e arco – ver [7].

Ao longo do texto, as designações aresta e arco serão utilizados de forma indistinta. Apenas quando for necessário será explicitamente referido o sentido de um arco dirigido. Para ilustrar pode-se observar que uma rede não dirigida pode ser representada por uma rede dirigida em que cada aresta é representada por dois arcos dirigidos (com iguais atributos e sentidos opostos).

No contexto do encaminhamento com restrições vai ser considerada uma métrica aditiva associada ao custo de um caminho. Assim, o custo de um caminho é a soma do custo dos

arcos que constituem esse caminho. De seguida são apresentadas as variáveis e respectivas definições que serão usadas para tratar os problemas estudados.

Definição 2. *Um caminho I é escrito como a sequência dos nós tal que $I = \langle n_{i_1}, n_{i_2}, ..., n_{i_k} \rangle$, com $(n_{i_j}, n_{i_{j+1}}) \in A$, $j = 1, \cdots, k-1$.*

Definição 3. *O custo de um caminho é definido por $f(I) = \sum_{l=2}^{k} f(n_{i_{l-1}}, n_{i_l})$, sendo portanto considerada uma métrica aditiva.*

Ao caminho de menor custo dá-se o nome de caminho mais curto. I é um caminho elementar, sem ciclos, se não existirem nós repetidos.

Ao longo deste trabalho, seguindo a notação utilizada em [1], [3] utilizar-se-á n_1 como nó origem do caminho e n_p como nó destino. Isto contudo não implica qualquer restrição na utilização de qualquer outro par de nós origem e destino na resolução destes problemas. Seja $N_I = \{n_{i_1}, n_{i_2}, \cdots, n_{i_k}\}$ o conjunto dos nós utilizados pelo caminho $I = \langle n_{i_1}, n_{i_2}, ..., n_{i_k} \rangle$ e $P^S_{n_1-n_p}$ o conjunto de todos os caminhos elementares de n_1 para n_p que contêm entre os nós intermédios os nós do conjunto S, sendo o S o conjunto dos nós específicos.

$$P^S_{n_1-n_p} = \{I = \langle n_{i_1} = n_1, n_{i_2}, \cdots, n_{i_k} = n_p \rangle : N_I \subseteq N, N_I \cap S = S\} \tag{2.1}$$

e $(n_{i_j}, n_{i_{j+1}}) \in A, \quad j = 1, 2, \cdots, p-1$.

Assim, o PCE pode escrever-se da seguinte forma:

$$I^* = \arg \min_{I \in P^S_{n_1-n_p}} f(I) \tag{2.2}$$

Algumas vezes será utilizado o símbolo $I_{n_i-n_j}$ para representar um caminho de um nó n_i para n_j.

A concatenação de dois caminhos será representado pelo símbolo $\diamond$. Por exemplo, sendo $I_{n_1-n_d} = \langle n_{i_1} = n_1, n_{i_2}, \cdots, n_{i_u} = n_d \rangle$ e $I_{n_d-n_p} = \langle n_{j_1} = n_d, n_{j_2}, \cdots, n_{j_v} = n_p \rangle$ então, $I_{n_1-n_d} \diamond I_{n_d-n_p}$ será o caminho $I_{n_1-n_p} = \langle n_{i_1} = n_1, n_{i_2}, \cdots, n_{i_u} = n_{j_1} = n_d, n_{j_2} \cdots, n_{j_v} = n_p \rangle$.

Neste trabalho deseja-se garantir que além de nós específicos podem também ser solicitados arcos específicos. A estratégia adoptada foi aplicar uma modificação à rede que transforma um arco num nó. A figura 2.1 ilustra o procedimento para o caso de uma aresta genérica (n_i, n_j), de custo c.

Foi considerado que não é conhecido o sentido preferencial da utilização da aresta (n_i, n_j) no caminho de n_1 para n_p. Se a mesma fosse conhecida bastaria adicionar o nó artificial n_k e dois arcos dirigidos. Desta forma o problema de passagem em elementos específicos (nós e/ou arcos) da rede é transformado no problema de passagem em nós específicos na rede modificada.

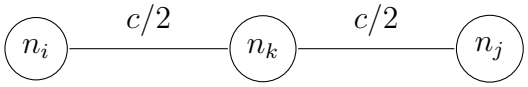


Figura 2.1: Ilustração da construção do nó fictício.

O habitual é visualizar os grafos por meio de diagramas ou desenhos. Certamente, também é familiar descrevê-los pelo conjunto N dos seus nós e pelo conjunto A das suas arestas. A eficiência computacional de um algoritmo para resolver problemas de optimização em redes não depende apenas das suas características intrínsecas, mas também das estruturas de dados utilizadas para representar a rede no programa para resolver o problema (formas de armazenar e manipular os dados associados à rede) e para armazenar os resultados intermédios necessários ao algoritmo. Para representar a rede são necessárias dois tipos de informação:

a) a topologia da rede (estrutura dos nós e dos arcos);

b) o custo e possivelmente outros atributos relevantes associados aos nós e aos arcos.

A representação utilizada foi uma lista de adjacências, [8]. Uma lista de adjacência de arcos $A(i)$ de um nó i corresponde ao conjunto de arcos emergentes desse nó, isto é $A(i) = \{(i,j) \in A : j \in N\}$. Uma lista de adjacência de nós $N(i)$ é o conjunto de nós adjacente a esse mesmo nó, neste caso $N(i) = \{j \in N : (i,j) \in A\}$. Portanto, é natural que sejam omitidos os termos arco e nó e simplesmente referirmo-nos a uma lista de adjacências.

2.2 Algoritmo de Saksena e Kumar

No artigo [1] é apresentado um algoritmo que calcula um caminho *que pode não ser elementar* que passa em k nós específicos, com origem no nó n_1 e destino no nó n_p, sendo $p = |N|$ o número de nós da rede. Neste algoritmo o caminho óptimo é visto como a concatenação de dois sub-caminhos mais curtos, entre a origem n_1 e um nó intermédio n_l (o sub-caminho $I_{n_1 n-n_l}$) e entre n_l e n_p (o sub-caminho $I_{n_l-n_p}$), com $n_l \in S$. Assim, sendo pode-se representar o caminho óptimo como a concatenação dos sub-caminhos $I_{n_1 n-n_l} \diamond I_{n_l-n_p}$.

O algoritmo baseia-se fundamentalmente nestes dois passos:

1. Determina-se a distância óptima num caminho sem restrições entre um par ordenado de nós específicos (n_i, n_l) denominada por $D^r(n_i, n_l)$ em que r indica o número de nós específicos intermédios do caminho considerado. É ainda calculado $D^r(n_1, n_l)$, em que se recorda que n_1 é a origem do caminho.

2. Determina-se $f^{\xi}_{n_i}$, a distância mínima do nó específico intermédio n_l ao destino final n_p, passando pelo menos por ξ nós distintos específicos, excluindo n_l dessa contagem assim como as suas possíveis repetições no caminho.

O princípio da optimalidade, para a programação dinâmica que segundo Saksena e Kumar [1] seria respeitado por este algoritmo, foi definido por Richard Bellman em [9]:

> An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Assim sendo, a ideia fundamental por detrás da teoria da programação dinâmica é a de visualizar uma solução óptima como uma escolha tomada em cada momento em termos do estado actual do sistema. De acordo com este princípio podemos escrever:

$$f_{n_i}^{\xi} = \min_{n_l}[D^r(n_i, n_l) + f_{n_l}^{\xi - r - 1}] \quad \text{para} \quad n_l \in S \quad \text{e} \quad n_l \neq n_i \tag{2.3}$$

O número total de nós específicos do caminho desde n_i até n_p deve ser pelo menos ξ e este facto deve ser verificado em todos os passos do algoritmo. Para elucidar a notação usada, ilustra-se um caso em particular. Por exemplo, $f_{n_i}^0$ de acordo com o ponto 2 corresponde à distância mínima desde o nó n_i ao destino final n_p sem passar[1] por nenhum nó específico. Escreve-se o passo inicial como:

$$f_{n_i}^1 = \min_{n_l}[D(n_i, n_l) + f_{n_l}^0] \tag{2.4}$$

Por observação verifica-se que o autor descarta o valor de r. No passo inicial $\xi = 1$ pelo que n_l é o único nó específico neste sub-caminho e r vale 0.

2.2.1 Exemplo ilustrativo

Para clarificar este método utiliza-se a rede apresentada na figura 2.2 proposta pelo autor em [1], com o intuito de determinar o caminho mais curto de 0 para 9 com nós obrigatórios 2, 4, 6 e 8, ou seja, $S = \{2, 4, 6, 8\}$ – tal como é feito em [1].

O primeiro passo é a determinação dos valores $D^r(n_i, n_l)$. Estes valores são apresentados na tabela 2.1.

Os valores entre parêntesis em todas as tabelas são os nós intermédios constituintes do caminho desde n_i até n_l. O valor à esquerda é o custo do caminho total (entre n_i e n_p) e a ausência de valores entre parêntesis indica que não há nós intermédios entre n_i e n_l, ou seja, o caminho é simplesmente o arco (n_i, n_l). O expoente x junto a um custo significa que o sub-caminho correspondente não é admissível porque não contém um número mínimo de nós específicos requerido nesse passo. Isto é válido para as tabelas 2.1 a 2.6. Assim, por exemplo na coluna $f_{n_i}^2$ da tabela 2.6 o custo 8 refere-se ao caminho desde 2 até 9,

Observando a tabela 2.1 verifica-se que os valores de r variam entre 0 e 2 consoante o número de nós intermédios presentes nesses caminhos. A tabela 2.2 contém o caminho mais

[1] O autor afirma "*without passing*" (sem passar), mas mais à frente verifica-se que deverá ser *without having to pass* (sem ter de passar) para respeitar a resolução do exemplo proposto.

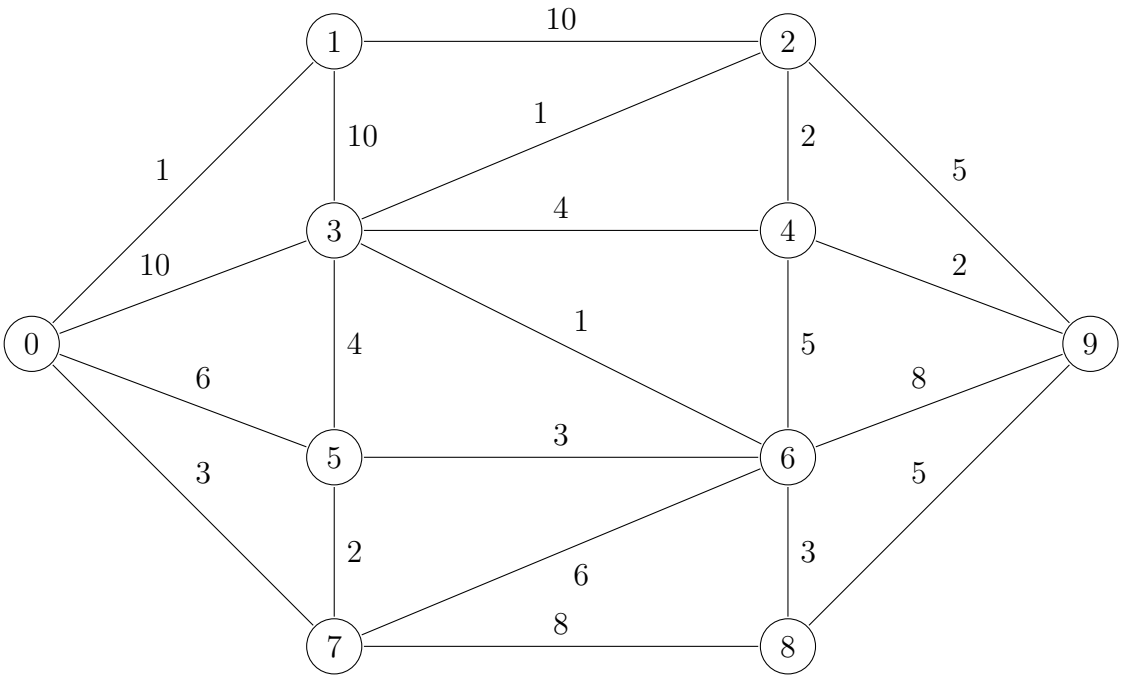


Figura 2.2: Rede usada no artigo [1]. Os nós obrigatórios são 2,4,6 e 8.

curto de cada nó específico até ao nó destino. Nessa tabela r_{n_i,n_p} indica o número de nós específicos intermédios de cada caminho.

As tabelas 2.3 a 2.5 apresentam os valores $f^1_{n_i}$, $f^2_{n_i}$, $f^3_{n_i}$ respectivamente. Note-se que cada tabela precisa do cálculo prévio da anterior.

Na tabela 2.6 estão agrupados os valores $f^{\xi-1}_{n_i}$, necessários ao cálculo de $f^{\xi}_{n_i}$ para $\xi = 1, 2, 3$, com base na equação (2.3).

O passo final corresponde ao cálculo de $f^4_0 = \min_{n_l}[D^r(0, n_l) + f^{\xi-1}_{n_l}], \quad n_l = 2, 4, 6, 8$. Note-se que r em $D^r(0, n_l)$ toma o valor 0 para $n_l = 2, 6$ e toma o valor 1 para $n_l = 4, 8$ – ver tabela 2.1.

$$f^4_0 = \min \begin{cases} D(0,2) + f^3_2 = 10 + 14 = 24 \\ D(0,4) + f^3_4 = 12 + 12 = 24 \\ D(0,6) + f^3_6 = 8 + 12 = 20 \\ D(0,8) + f^3_8 = 11 + 9 = 20 \end{cases} \tag{2.5}$$

Os caminhos óptimos são $\langle 0, 7, 5, 6, 8, 6, 3, 2, 4, 9\rangle$ que contém um ciclo e o caminho elementar $\langle 0, 7, 8, 6, 3, 2, 4, 9\rangle$, e encontram-se representados na figura 2.3.

De notar que o primeiro caminho contém um ciclo e o segundo é um caminho elementar.

Para ilustrar o funcionamento das tabelas, observe-se como foi construído o caminho $\langle 0, 7, 8, 6, 3, 2, 4, 9\rangle$. Este caminho corresponde à solução de custo $D^0(0, 8) + f^3_8 = 11 + 9 = 20$:

- $D^0(0, 8)$ é o custo do caminho mais curto sem restrições de 0 até 8, ou seja, $\langle 0, 7, 8\rangle$ (ver

$D^r(n_i, n_l)$				
n_i	$n_l = 2$	$n_l = 4$	$n_l = 6$	$n_l = 8$
0	10 (7,5,3)	12 (7,5,3,2)	8 (7,5)	11 (7) ou (7,5,6)
2	-	2	2 (3)	5 (3,6)
4	2	-	4 (2,3)	7 (2,3,6)
6	2 (3)	4 (3,2)	-	3
8	5 (6,3)	7 (6,3,2)	3	-

Tabela 2.1: Tabela com os dados $D^r(n_i, n_l)$ referente à rede da figura 2.2.

n_i	$f^0_{n_i}$	$r_{n_i n_p}$
2	4 (4)	1
4	2	0
6	6 (3,2,4)	2
8	5	0

Tabela 2.2: Tabela com os dados $f^0_{n_i}$ referente à rede da figura 2.2.

1ª linha da tabela 2.1). Neste sub-caminho não existem nós intermédios específicos (o nó 8 não conta pois não é intermédio ao caminho). Por conseguinte, f^3_8 deve representar o custo de um caminho com 3 nós específicos (excluindo o nó 8).

- f^3_8 tem o custo 9 – ver tabela 2.5 ou 2.6 – e o nó que sucede ao nó 8 é o nó 6 (ver coluna $f^3_{n_i}$ e linha $n_i = 8$ na tabela 2.6). f^3_8 escreve-se tal que $f^3_8 = D^0(8, 6) + f^2_6 = 3 + 6 = 9$.

- O sub-caminho devido a f^3_8 é então $\langle 8, 6\rangle$ concatenado com o caminho de custo f^2_6 que corresponde a $\langle 6, 3, 2\rangle$ – ver tabela 2.6, linha com $n_i = 6$ e coluna $f^2_{n_i}$, – para $n_l = 2$ e é preciso identificar o sub-caminho de custo $f^1_2 = 4$.

- $f^1_2 = D(2, 4) + f^0_4$, $D(2, 4)$ é o custo do sub-caminho $\langle 2, 4\rangle$ e $f^0_4 = 2$ representa o sub-caminho $\langle 4, 9\rangle$.

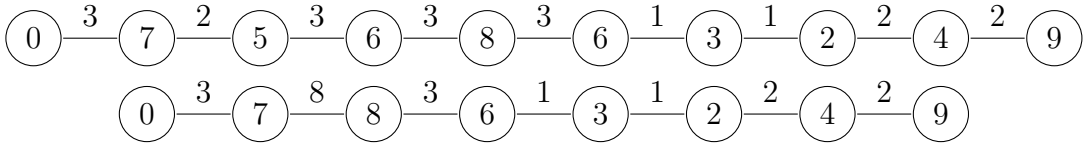


Figura 2.3: Identificação dos caminhos mais curtos para a rede da figura 2.2 com nós obrigatórios $S = \{2, 4, 6, 8\}$.

$f^1_{n_i}$					
n_i	$n_l = 2$	$n_l = 4$	$n_l = 6$	$n_l = 8$	$\min(n_l = 2, n_l = 4, n_l = 6, n_l = 8)$
2	-	4	8 (3)	10 (3,6)	4 (4)
4	6	-	10 (2,3)	12 (2,3,6)	6 (2)
6	6 (3)	6 (3,2)	-	8	6 (3,2) ou 6 (3,2,4)
8	9 (6,3)	9 (6,3,2)	9	-	9 (6,3,2) ou (6,3,2,4) ou (6)

Tabela 2.3: Tabela com os dados $f^1_{n_i}$ referente à rede da figura 2.2.

$f^2_{n_i}$					
n_i	$n_l = 2$	$n_l = 4$	$n_l = 6$	$n_l = 8$	$\min(n_l = 2, n_l = 4, n_l = 6, n_l = 8)$
2	-	8^x	8 (3)	10 (3,6)	8 (3,6)
4	6^x	-	10 (2,3)	12 (2,3,6)	10 (2,3,6)
6	6 (3)	10 (3,2)	-	12	6 (3,2)
8	9 (6,3)	9 (6,3,2)	9	-	9 (6,3,2) ou (6,3,2,4) ou (6)

Tabela 2.4: Tabela com os dados $f^2_{n_i}$ referente à rede da figura 2.2.

- Logo o caminho completo é: $\langle 0, 7, 8\rangle \diamond \langle 8, 6\rangle \diamond \langle 6, 3, 2\rangle \diamond \langle 2, 4\rangle \diamond \langle 4, 9\rangle$, ou seja, $\langle 0, 7, 8, 6, 3, 2, 4, 9\rangle$.

2.2.2 Análise do algoritmo de Saksena e Kumar

Após leitura e análise do artigo verificam-se alguma inconsistências entre a descrição apresentada na secção 2.2 e o exemplo replicado na sub-secção 2.2.1.

Na expressão (2.3) existe a possibilidade de $\xi - r - 1$ assumir valores negativos o que não tem qualquer significado para o entendimento do problema. Pode-se observar isto se por exemplo: no cálculo de $f^1_{n_i}$, $D(n_i, n_l)$ tem um ou mais nós específicos intermédios.

Seguindo com rigor a definição de $f^0_{n_i}$ dada pelos autores em [1], $f^0_{n_i}$ seria o custo do

$f^3_{n_i}$					
n_i	$n_l = 2$	$n_l = 4$	$n_l = 6$	$n_l = 8$	$\min(n_l = 2, n_l = 4, n_l = 6, n_l = 8)$
2	-	12^x	8^x	$14(3, 6)$	14 (3,6)
4	10^x	-	$10^x(2, 3)$	12 (2,3,6)	12 (2,3,6)
6	$10^x(3)$	$14^x(3, 2)$	-	12	12 (8)
8	9 (6,3)	9 (6,3,2)	9	-	9 (6)

Tabela 2.5: Tabela com os dados $f^3_{n_i}$ referente à rede da figura 2.2.

n_i	$f^0_{n_i}$	$f^1_{n_i}$	$f^2_{n_i}$	$f^3_{n_i}$
2	4 (4)	4 (4)	8 (3,6)	14 (3,6,8)
4	2	6 (2)	10 (2,3,6)	12 (2,3,6,8)
6	6 (3,2,4)	6 (3,2) ou (3,2,4)	6 (3,2)	12 (8)
8	5	9 (6,3,2) ou (6,3,2,4)	9 (6) ou (6,3,2)	9 (6)

Tabela 2.6: Tabela com os dados resultantes das tabelas 2.2, 2.3, 2.4 e 2.5.

caminho óptimo de n_i para n_p independentemente do número de nós intermédios que possa ter e estes não poderiam ser nós específicos. Considera-se que no final da secção 2.2 aquando da definição de f^0_i, o autor quis dizer *sem ter de passar* em vez de *sem passar*, ou seja:

> $f^0_{n_i}$ é a distância mínima desde o nó n_i até ao destino final n_p sem ter de passar através de nenhum nó específico.

Para verificar a necessidade desta correcção basta observar, por exemplo, o caso f^0_2 em que na tabela 2.2 do exemplo surge $\langle 2, 4, 9\rangle$ como o caminho de menor custo ficando f^0_2 com o valor 4. De igual forma surge nessa tabela o caminho $\langle 6, 3, 2, 4, 9\rangle$ ficando f^0_6 com o valor 6. Este último passa por dois nós específicos.

Em resumo, uma interpretação do algoritmo consistente com a própria resolução do exemplo proposto pelos autores, é a seguinte:

a) Se a contagem dos nós específicos no caminho com custo $f^\xi_{n_i}$ for inferior a ξ o caminho é não admissível.

b) A contagem dos nós deverá ser feita da seguinte forma:

   i) O nó origem n_i ou suas repetições não é contabilizado;

   ii) Os nós específicos não incluem a origem n_i do sub-caminho a calcular nem o destino do caminho inicialmente procurado, n_p.

   iii) Repetições de nós específicos não são contabilizadas.

2.2.3 Crítica à análise realizada por Dreyfus

No artigo [2], Dreyfus afirma que o método proposto em [1] está errado. Segundo Dreyfus:

[2] 1 corresponde a n_1.
[3] N corresponde a n_p.
[4] p corresponde a ξ de [1].
[5] i corresponde a n_i.
[6] j corresponde a n_l.

> The falacy (...) is the assertion (subject to a proviso to follow) that the shortest path from a specified node $^2 1$ to $^3 N$ passing through at least $^4 p$ of the specified nodes enroute is composed of the shortest path from $^5 i$ to some specified node $^6 j$, followed by the shortest path from $^5 j$ to $^3 N$ passing through $^4 p - r - 1$ specified nodes, where r is the number of specified nodes that lie on the shortest unrestricted path from $^5 i$ to $^6 j$.

O problema reside no seguinte: o facto de um sub-caminho mais curto do nó n_i para o nó n_j conduzir a um caminho candidato que posteriormente é descartado porque não contém o número mínimo de nós específicos necessários, impedindo a utilização do nó n_j como um nó intermédio a partir do nó n_i.

No contra-exemplo dado por Dreyfus é usado um grafo representado em anexo na figura A.1. Dreyfus pretendeu ilustrar que ao procurar o caminho mais curto de 1 para 5 passando no mínimo por dois nós intermédios, ou seja, considerando dois nós do conjunto $S = \{2, 3, 4\}$ como os nós específicos que o caminho obtido não seria a solução óptima. Na descrição detalhada no anexo A, e traduzida pelas tabelas A.2 a A.3, pode confirmar-se que o contra-exemplo dado por Dreyfus está errado: o algoritmo de Saksena e Kumar encontra neste caso a solução óptima.

Apresenta-se na figura 2.4 uma rede que corresponde à rede da figura apresentada em A.1 com um arco adicional $(0, 1)$, na qual se irá obter o caminho específico do nó 0 para o nó 5 considerando $S = \{1, 2, 3\}$. Este exemplo irá conduzir ao caminho $\langle 0, 1, 2, 1, 3, 5\rangle$ de custo 6, demonstrando assim o fundamento da observação feita por Dreyfus, pois o caminho óptimo é $\langle 0, 1, 2, 3, 5\rangle$ de custo 5, como se pode verificar por observação da figura 2.4.

Tal como na secção anterior, nas tabelas 2.7 a 2.10 o valor à esquerda é o custo do caminho e os valores entre parêntesis são os novos nós intermédios constituintes do caminho desde n_i até n_l.

$D^r(n_i, n_l)$			
n_i	$n_l = 1$	$n_l = 2$	$n_l = 3$
0	1	2 (1)	3 (1)
1	-	1	2
2	1	-	2
3	∞	∞	-

Tabela 2.7: Tabela com os dados $D^r(n_i, n_l)$ referente à rede da figura 2.4.

Seguem-se os cálculos que permitem o preenchimento da tabela 2.9.

$$f^1_{n_i=1} \quad (n_l = 2): \quad D(n_i = 1, n_l = 2) + f^0_{n_l=2} = 1 + 2 = 3 \qquad \langle 1, 2, 1, 5\rangle \tag{2.6}$$

$$(n_l = 3): \quad D(n_i = 1, n_l = 3) + f^0_{n_l=3} = 2 + 1 = 3 \qquad \langle 1, 3, 5\rangle \tag{2.7}$$

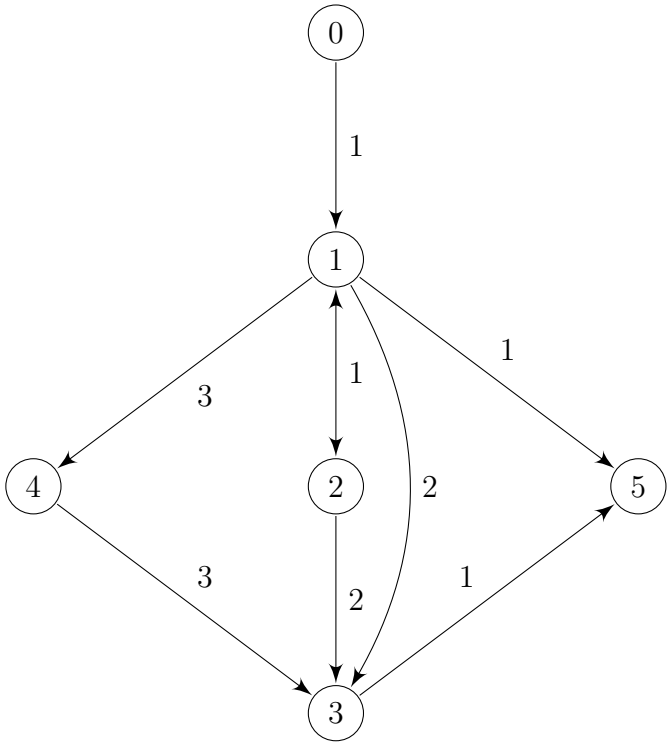


Figura 2.4: Rede baseada na figura A.1. Os nós obrigatórios são 1, 2 e 3.

n_i	$f^0_{n_i}$
1	1
2	2 (1)
3	1

Tabela 2.8: Tabela com os dados $f^0_{n_i}$ referente à rede da figura 2.4.

$$f^1_{n_i=2} \quad (n_l = 1): \quad D(n_i = 2, n_l = 1) + f^0_{n_l=1} = 1 + 1 = 2 \qquad \langle 2, 1, 5\rangle \tag{2.8}$$

$$(n_l = 3): \quad D(n_i = 2, n_l = 3) + f^0_{n_l=3} = 2 + 1 = 3 \qquad \langle 2, 3, 5\rangle \tag{2.9}$$

Não é realizado o cálculo $f^1_{n_i=3}$ porque $D(3, n_l) = \infty$ para $n_l = 1, 2$.

Tal como no caso anterior, seguem-se os cálculos detalhados que permitem o preenchimento da tabela 2.10.

$$f^2_{n_i=1} \quad (n_l = 2): \quad D(n_i = 1, n_l = 2) + f^1_{n_l=2} = 1 + 2 = 3^x \qquad \langle 1, 2, 1, 5\rangle \tag{2.10}$$

$$(n_l = 3): \quad D(n_i = 1, n_l = 3) + f^1_{n_l=3} = 2 + \infty = \infty \tag{2.11}$$

O caminho $\langle 1, 2, 1, 5\rangle$ é não admissível porque contém apenas um nó intermédio obrigatório (o nó 2) quando $\xi = 2$, uma vez que o nó 1 não é contabilizado porque é a origem de $f^2_{n_i=1}$.

$$f^2_{n_i=2} \quad (n_l = 1): \quad D(n_i = 2, n_l = 1) + f^1_{n_l=1} = 1 + 3 = 4^x \qquad \langle 2, 1, 2, 1, 5\rangle \qquad (2.12)$$

$$(n_l = 1): \quad D(n_i = 2, n_l = 1) + f^1_{n_l=1} = 1 + 3 = 4 \qquad \langle 2, 1, 3, 5\rangle \qquad (2.13)$$

$$(n_l = 3): \quad D(n_i = 2, n_l = 3) + f^1_{n_l=3} = 2 + \infty = \infty \qquad (2.14)$$

De igual forma o caminho $\langle 2, 1, 2, 1, 5\rangle$ não é admissível porque contém apenas um nó intermédio obrigatório (o nó 1) – o nó 2 não é contabilizado porque é a origem de $f^2_{n_i=2}$. Tal como no caso $f^1_{n_i=3}$, não é realizado o cálculo $f^2_{n_i=3}$ porque $D(3, n_l) = \infty$ para $n_l = 1, 2$.

$f^1_{n_i}$				
n_i	$n_l = 1$	$n_l = 2$	$n_l = 3$	$\min(n_l = 1, n_l = 2, n_l = 3)$
1	-	3	3	3
2	2	-	3	2
3	∞	∞	∞	∞

Tabela 2.9: Tabela com os dados $f^1_{n_i}$ referente à rede da figura 2.4.

Seguem-se os cálculos detalhados de $f^2_{n_i}$ e a tabela 2.10 resultante.

$f^2_{n_i}$				
n_i	$n_l = 1$	$n_l = 2$	$n_l = 3$	$\min(n_l = 1, n_l = 2, n_l = 3)$
1	-	3^x	∞	∞
2	4	-	∞	4
3	∞	∞	-	∞

Tabela 2.10: Tabela com os dados $f^2_{n_i}$ referente à rede da figura 2.4.

Finalmente pode calcular-se f^2_1 e obter-se o caminho procurado do nó 1 para o nó 5.

n_i	$f^0_{n_i}$	$f^1_{n_i}$	$f^2_{n_i}$
1	1	3	∞
2	2 (1)	2	4
3	1	∞	∞

Tabela 2.11: Tabela com os dados resultantes das tabelas 2.8, 2.9 e 2.10.

$$f_0^3 = \min \begin{cases} D(0,1) + f_{n_l=1}^2 = 1 + \infty = \infty \\ D(0,2) + f_{n_l=2}^2 = 2 + 4 = 6 \qquad \langle 0,1,2,1,3,5 \rangle \\ D(0,3) + f_{n_l=3}^2 = 3 + \infty = \infty \end{cases} \tag{2.15}$$

Como já tinha sido referido, o caminho final é $\langle 0,1,2,1,3,5 \rangle$ de custo 6.

Note-se que a rede da figura 2.4 foi criada para ser possível obter um sub-caminho $I_{n_i - n_l}$ (no caso presente $\langle 2,1,5 \rangle$) de custo mínimo $f_{n_i=2}^{\xi=1} = 2$, que irá dar origem ao caminho descartado $\langle 1,2,1,5 \rangle$, devido ao facto do nó 1 ter sido contabilizado duas vezes durante a sua construção.

A detecção da dupla contagem do nó 1 é feita demasiado tarde, posteriormente à escolha de $\langle 2,1,5 \rangle$ (de custo 2) como sub-caminho no passo que minimiza $f_{n_i=2}^{\xi=1}$. Essa escolha impede, tal como Dreyfus previu, a escolha de $\langle 2,3,5 \rangle$ de custo 3, o qual permitiria obter a solução óptima.

Para resolver o mesmo problema assumindo tal como em [1] que os caminhos com ciclos são admissíveis, Dreyfus sugere o seguinte:

1. Resolver o problema do caminho mais curto para toda a rede com todos os pares de nós iniciais e finais (ou seja para $n_i, n_j \in S \cup \{n_1, n_p\}$).);

2. $f'(n_i, n_j)$ representa a distância de todos os caminhos mais curtos do nó n_i para n_j com $n_i, n_j \in S \cup \{n_1, n_p\}$.

3. Resolver o problema do caixeiro-viajante com $k+1$ cidades para o caminho mais curto de n_1 para n_p passando através de 2,3,$\cdots$,k nós na rede auxiliar onde a distância de n_i para n_j é $f'(n_i, n_j)$. Em [10] são discutidos métodos para resolver o problema.

2.2.4 Conclusão

Foi apresentado o algoritmo proposto por Saksena e Kumar [1]. Este algoritmo, como foi identificado por Dreyfus [2] não satisfaz o princípio de optimalidade uma vez que decisões intermédias impedem a obtenção da solução óptima. Mostra-se no anexo A que o contra-exemplo escolhido por Dreyfus não é adequado pois o algoritmo consegue, nesse caso particular, obter a solução óptima. Por conseguinte, foi apresentado na subsecção 2.2.3 um exemplo que efectivamente demonstra a falha deste algoritmo, com base na observação feita por Dreyfus.

Dreyfus considera que não existe uma forma simples de corrigir este algoritmo. De facto, julga-se que seria possível corrigir o seu funcionamento desde que fosse permitido recuar para uma iteração anterior ou iterações anteriores sempre que fosse detectada uma solução não admissível. Contudo, isto conduziria a um aumento significativo da complexidade do algoritmo.

Tratando-se de um problema NP-difícil [4] considera-se que a abordagem proposta por Saksena e Kumar resulta numa heurística. A ideia subjacente ao algoritmo de Saksena e Kumar poderá ser explorada em trabalho futuro no desenvolvimento de uma heurística que permita obter caminhos elementares, desenvolvendo uma estratégia para limitar a geração de soluções não admissíveis e também evitar a obtenção de soluções com ciclos.

2.3 Algoritmo de Ibaraki

2.3.1 Descrição do algoritmo de Ibaraki

O algoritmo discutido nesta secção foi proposto em [3] e resolve o PCE. Nesse texto o problema é representado por $\langle G, n_1, S, n_p\rangle$. Segue-se alguma notação adicional requerida à descrição deste algoritmo:

a) $\bar{N} = N - \{n_1, n_p\}$ e $S \subseteq \bar{N}$, sendo S o conjunto de nós obrigatórios;

b) Os arcos com custo ∞ representam arcos não existentes;

c) Os conjuntos L e M são os conjuntos de nós tais que $L \subseteq S$, $M \subseteq \bar{S}$ sendo $\bar{S} = \hat{N}$ - S e o trio (L, M, n_l) com $n_l \in L \cup M$ corresponde ao conjunto de caminhos elementares que começam no nó n_1, visitam o conjunto de nós $L \cup M \setminus \{n_l\}$ (e não outros) e terminam em n_l;

d) (S, n_p) corresponde ao conjunto de caminhos elementares que começam no nó n_1 visitam o conjunto de nós denominado S (e, possivelmente outros nós em $\bar{S}$) e terminam em n_p;

e) $g(L, M, n_l)$ e $g(S, n_p)$ representam o custo do caminho mais curto dos conjunto (L, M, n_l) e (S, n_p) respectivamente;

f) $I' = \langle n_1, n_{i1}, n_{i2}, ..., n_{ir}, n_k, n_l\rangle \in (L, M, n_l)$ identifica inequivocamente o caminho entre n_1 e n_l, sendo n_k o nó que precede n_l;

g) O caminho óptimo, ou seja, o caminho mais curto encontra-se no conjunto (S, n_p) e tem custo $g(S, n_p)$.

O método de resolução proposto em [3] é uma generalização do método proposto por [11] para o problema do caixeiro-viajante, e é descrito pelas devidas equações de recorrência de $\langle G, n_1, S, n_p\rangle$, que se seguem:

(i) $|L \cup M| = 1$,

$$\begin{cases} g(\{n_l\}, \emptyset, n_l) &= f(n_1, n_l) \text{ para } n_l \in S, \\ g(\emptyset, \{n_l\}, n_l) &= f(n_1, n_l) \text{ para } n_l \in \bar{S}(= \bar{N} - S) \end{cases} \tag{2.16}$$

(ii) $|L \cup M| > 1$,

$$g(L, M, n_l) = \begin{cases} \inf\{g(L \setminus \{n_l\}, M, n_k) + f(n_k, n_l) | n_k \in (L \setminus \{n_l\}) \cup M\} \\ \qquad\qquad \text{se } n_l \in L, \\ \inf\{g(L, M \setminus \{n_l\}, n_k) + f(n_k, n_l) | n_k \in L \cup (M \setminus \{n_l\})\} \\ \qquad\qquad \text{se } n_l \in M \end{cases} \tag{2.17}$$

(iii)

$$g(S, n_p) = \begin{cases} \inf_M \inf\{g(S, M, n_k) + f(n_k, n_p) | n_k \in S \cup M\} \\ \qquad\qquad \text{se } S \neq \emptyset, \\ \inf[f(n_1, n_p), \inf_M \inf\{g(S, M, n_k) + f(n_k, n_p) | n_k \in S \cup M\}] \\ \qquad\qquad \text{se } S = \emptyset \end{cases} \tag{2.18}$$

Na equação (2.16) é simples verificar que tendo apenas um elemento em $|L \cup M|$ com $n_1, n_p \notin L \cup M$, a distância entre n_1 e n_l corresponde ao valor de $f(n_1, n_l)$.

No caso da equação (2.17) é necessário reconhecer que para qualquer caminho elementar pertencente a (L, M, n_l) o nó que precede n_l identificado por n_k deve pertencer ao conjunto $L \cup M$, tal que $I' = \langle n_1, n_{i_1}, n_{i_2}, ..., n_{i_r}, n_k, n_l \rangle \in (L, M, n_l)$ com $f(I') = g(L, M, n_l)$. Deve-se ter em conta que $I'' = \langle n_1, n_{i_1}, n_{i_2}, ..., n_{i_r}, n_k \rangle \in (L \setminus \{n_l\}, M, n_k)$ ou $(L, M \setminus \{n_l\}, n_k)$, dependendo se $n_l \in L$ ou $n_l \in M$, e portanto $f(I'') = g(L \setminus \{n_l\}, M, n_k)$ ou $g(L, M \setminus \{n_l\}, n_k)$.

Para concretizar, primeiro obtemos $g(\{n_l\}, \emptyset, n_l)$ e $g(\emptyset, \{n_l\}, n_l)$ denotado pela equação (2.16). Depois calculamos $g(L, M, n_l)$ por ordem de crescimento dos conjuntos L e M. Note-se que $g(L, M, n_l)$ pode ser calculado para todos os $g(L', M', n_k)$, onde $L' \subseteq L, M' \subseteq M$ e $(L', M') \neq (L, M)$. Depois de obter $g(L, M, n_l)$ para todos os $L \subseteq S$, $M \subset \bar{S}$, $n_l \in L \cup M$, $g(S, n_p)$ é calculado pela equação (2.18).

As equações de recorrência (2.17) e (2.18) tornam-se mais claras se for tornada explícita a dependência dos elementos pertencentes ao conjunto L com a passagem para o passo (iii). Para alcançar o conjunto L desejado recorre-se a um processo iterativo. A iteração corresponde à adição de um dado nó $\{n_l\}$ de maneira que $L_i(n_l)$ e $M_i(n_l)$ represente o conjunto L e M obtido após a iteração i. Deve-se iterar até que o conjunto de nós seja tal que $L = S$ e o caminho mais curto tenha que passar obrigatoriamente por esses nós gerando um conjunto de possibilidades.

Deste conjunto de possibilidades descarta-se aquelas que não são possíveis pelo facto de não se poder completar um caminho com os nós envolvidos; aqueles que apresentam solução mas cujo conjunto L ainda não contém todos os nós pertencentes a S necessita de ser iterado até que tal se concretize. Da mesma forma avalia-se nessa iteração se é possível estabelecer o caminho com o conjunto de nós em L até ao respectivo n_l.

O algoritmo vai ser explicado com recurso a um exemplo utilizando a rede da figura 2.5.

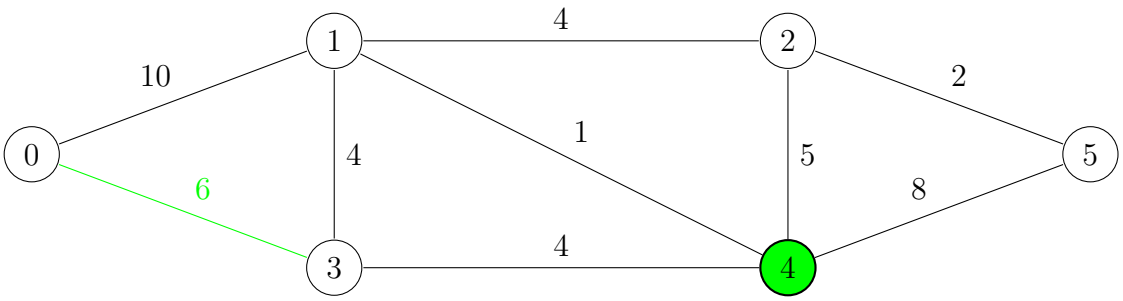


Figura 2.5: Rede alterada a partir da original usada no artigo [1].

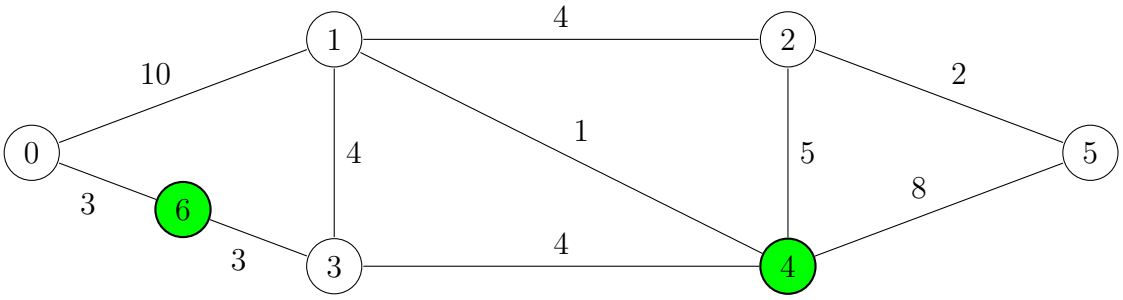


Figura 2.6: Rede alterada a partir da original usada no artigo [1] com inserção do nó fictício 6.

Suponha-se o problema de determinação do caminho mais curto entre o nó origem $n_1 = 0$ e o nó destino $n_p = 5$ tendo como nó obrigatório o nó 4 (isto é $S = \{4\}$) e o arco $(0, 3)$ obrigatório (donde $T = \{(0, 3)\}$). O algoritmo de Ibaraki constrói os caminhos possíveis por cada nível associado à árvore da figura 2.7. De notar, que tal como foi visto no final da secção anterior 2.1 os arcos obrigatórios são transformados num nó fictício, e portanto, no contexto do algoritmo um arco obrigatório corresponde à adição de um nó ao conjunto S. Neste caso o nó fictício é o nó 6 representado na figura 2.6. Para cada nível verifica-se a veracidade da condição: $L = S$ (no exemplo $S = \{4, 6\}$) e quando atingida significa que agora o objectivo seguinte e final é atingir o destino com um sub-caminho de menor custo possível, sem repetir os nós já pertencentes ao caminho identificados pelos conjuntos S e M.

O primeiro passo identificado pela equação (2.16) traduzido na tabela 2.12 corresponde ao primeiro nível da árvore da figura 2.7. Neste primeiro passo atinge-se logo o nó obrigatório 6 com o sub-caminho $\langle 0, 6\rangle$, e portanto $L = \{6\}$. O outro sub-caminho é $\langle 0, 1\rangle$ e como o nó 1 é não obrigatório $M = \{1\}$.

O segundo passo vem na sequência do primeiro, sendo feitas várias iterações, até que $L = S$ como já foi referido anteriormente. Encontram-se a negrito nas tabelas 2.14 – 2.16 os sub-caminhos em que $L = S$. Neste caso, na segunda iteração do passo traduzido pela equação (2.17), temos o sub-caminho $\langle 0, 6, 3, 4\rangle$ (ver tabela 2.14), com $L = S$ o que permite passar imediatamente para o passo três dado pela equação (2.18) que irá ser responsável pela

construção dos caminhos finais. Contudo antes de avançar para este passo será necessário ainda prosseguir para iterações seguintes do passo representado pela equação (2.17) e construir os restantes sub-caminhos até ao momento em que bastará adicionar um arco incidente em n_p para obter um caminho candidato a caminho mais curto. Serão assim obtidos a partir de $\langle 0, 6, 3, 4\rangle$ os sub-caminhos $\langle 0, 6, 3, 4, 2, 5\rangle$, $\langle 0, 6, 3, 4, 1, 2, 5\rangle$ e $\langle 0, 6, 3, 4, 5\rangle$ apresentados na tabela 2.17. Os restantes sub-caminhos resultantes do passo 2.17 estão representados nas tabelas 2.13 a 2.16.

Note-se que na tabela 2.16 quando $n_l = 2$ existem dois caminhos com $M = \{1, 2, 3\}$ e $L = \{4, 6\}$, pelo que pela equação (2.17) basta considerar o caminho de menor custo, razão pela qual na figura 2.7 um desses caminhos (o de maior custo até $n_l = 2$) termina no nó 2. Assim na tabela final surgem apenas dois caminhos com $n_l = 2$. De forma semelhante quando $n_l = 6$ ainda na tabela 2.16 é escolhido o sub-caminho de menor custo entre os dois presentes nessa tabela que, contudo não consta na tabela final porque não é possível atingir (sem ciclos) o nó n_p a partir desses sub-caminhos.

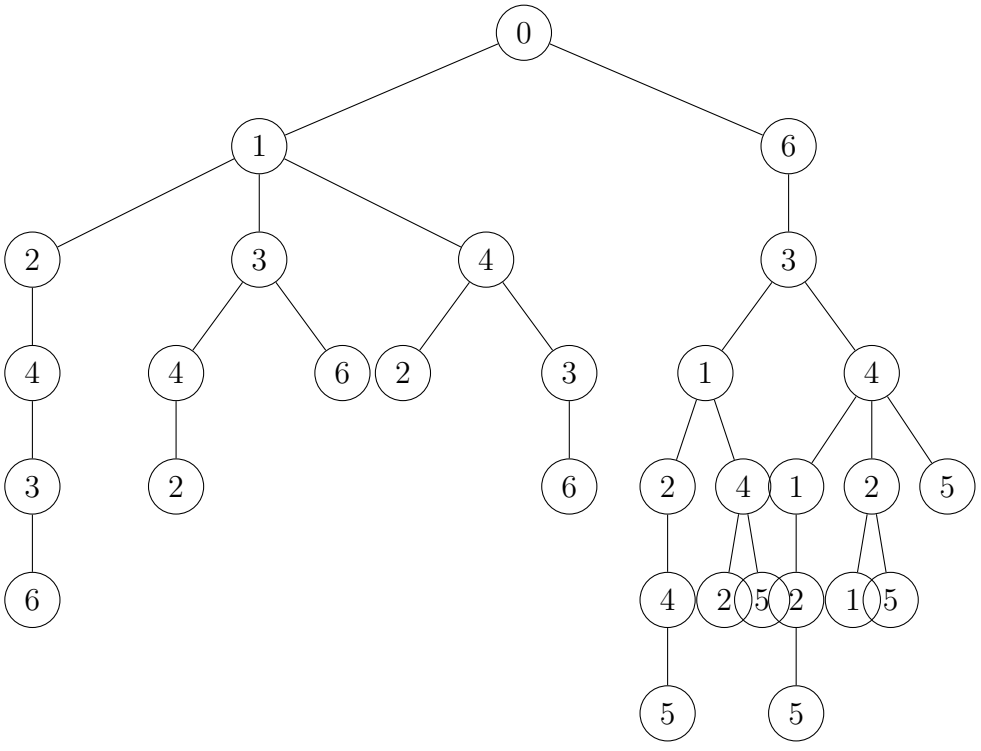


Figura 2.7: Árvore com os caminhos considerados na rede da figura 2.5 para o problema com origem $n_1 = 0$ e $n_p = 5$.

Na tabela 2.17 encontram-se os caminhos obtidos pelo algoritmo onde se podem identificar dois caminhos alternativos de custo mínimo.

n_l	I	$L_1(n_l)$	$M_1(n_l)$	$g(L_1(n_l), M_1(n_l), n_l)$
1	$\langle 0,1\rangle$	$\emptyset$	$\{1\}$	10
3	$\langle 0,6\rangle$	$\{6\}$	$\emptyset$	3

Tabela 2.12: Exemplo do algoritmo de Ibaraki – Passo (i).

n_l	I	$L_1(n_k)$	$M_1(n_k)$	$L_2(n_l)$	$M_2(n_l)$	$g(L_2(n_l), M_2(n_l), n_l)$
2	$\langle 0,1,2\rangle$	$L_1(1) = \emptyset$	$M_1(1) = \{1\}$	$\emptyset$	$\{1,2\}$	14
3	$\langle 0,1,3\rangle$	$L_1(1) = \emptyset$	$M_1(1) = \{1\}$	$\emptyset$	$\{1,3\}$	14
	$\langle 0,6,3\rangle$	$L_1(6) = \{6\}$	$M_1(6) = \emptyset$	$\{6\}$	$\{3\}$	6
4	$\langle 0,1,4\rangle$	$L_1(1) = \emptyset$	$M_1(1) = \{1\}$	$\{4\}$	$\{1\}$	11

Tabela 2.13: Exemplo do algoritmo de Ibaraki – Passo (ii) - 1ª iteração.

n_l	I	$L_2(n_k)$	$M_2(n_k)$	$L_3(n_l)$	$M_3(n_l)$	$g(L_3(n_l), M_3(n_l), n_l)$
1	$\langle 0,6,3,1\rangle$	$L_3(3) = \{6\}$	$M_3(3) = \{3\}$	$\{6\}$	$\{1,3\}$	10
2	$\langle 0,1,4,2\rangle$	$L_3(4) = \{4\}$	$M_3(4) = \{1\}$	$\{4\}$	$\{1,2\}$	16
3	$\langle 0,1,4,3\rangle$	$L_3(4) = \{4\}$	$M_3(4) = \{1\}$	$\{4\}$	$\{1,3\}$	15
4	$\langle 0,1,2,4\rangle$	$L_3(2) = \emptyset$	$M_3(2) = \{1,2\}$	$\{4\}$	$\{1,2\}$	19
	$\langle 0,1,3,4\rangle$	$L_3(3) = \emptyset$	$M_3(3) = \{1,3\}$	$\{4\}$	$\{1,3\}$	18
	$\langle \mathbf{0,6,3,4}\rangle$	$\mathbf{L_3(3) = \{6\}}$	$\mathbf{M_3(1) = \{3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{3}\}$	**11**
6	$\langle 0,1,3,6\rangle$	$L_3(3) = \emptyset$	$M_3(3) = \{1,3\}$	$\{6\}$	$\{1,3\}$	17

Tabela 2.14: Exemplo do algoritmo de Ibaraki – Passo (ii) - 2ª iteração.

n_l	I	$L_3(n_k)$	$M_3(n_k)$	$L_4(n_k)$	$M_4(n_k)$	$g(L_4(n_l), M_4(n_l), n_l)$
1	$\langle 0,6,3,4,1\rangle$	$L_3(4) = \{4\}$	$M_3(4) = \{2,3\}$	$\{4\}$	$\{1,2,3\}$	11
2	$\langle 0,1,3,4,2\rangle$	$L_3(4) = \{4\}$	$M_3(4) = \{1,3\}$	$\{6\}$	$\{1,2,3\}$	23
	$\langle 0,6,3,1,2\rangle$	$L_3(1) = \{6\}$	$M_3(1) = \{1,3\}$	$\{6\}$	$\{1,2,3\}$	14
	$\langle \mathbf{0,6,3,4,2}\rangle$	$\mathbf{L_3(4) = \{4,6\}}$	$\mathbf{M_3(4) = \{3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{2,3}\}$	**15**
3	$\langle 0,1,2,4,3\rangle$	$L_3(4) = \{4\}$	$M_3(4) = \{1,2\}$	$\{4\}$	$\{1,2,3\}$	23
4	$\langle \mathbf{0,6,3,1,4}\rangle$	$\mathbf{L_3(1) = \{6\}}$	$\mathbf{M_3(1) = \{1,3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,3}\}$	**11**
6	$\langle \mathbf{0,1,4,3,6}\rangle$	$\mathbf{L_3(3) = \{4\}}$	$\mathbf{M_3(3) = \{1,3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,3}\}$	**18**

Tabela 2.15: Exemplo do algoritmo de Ibaraki – Passo (ii) - 3ª iteração.

n_l	I	$L_4(n_k)$	$M_4(n_k)$	$L_5(n_k)$	$M_5(n_k)$	$g(L_5(n_l), M_5(n_l), n_l)$
2	$\langle \mathbf{0,6,3,4,2,1}\rangle$	$\mathbf{L_4(2) = \{4,6\}}$	$\mathbf{M_4(2) = \{2,3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,2,3}\}$	**19**
2	$\langle \mathbf{0,6,3,4,1,2}\rangle$	$\mathbf{L_4(1) = \{4,6\}}$	$\mathbf{M_4(1) = \{1,3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,2,3}\}$	**15**
	$\langle \mathbf{0,6,3,1,4,2}\rangle$	$\mathbf{L_4(4) = \{4,6\}}$	$\mathbf{M_4(4) = \{1,3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,2,3}\}$	**16**
4	$\langle \mathbf{0,6,3,1,2,4}\rangle$	$\mathbf{L_4(2) = \{6\}}$	$\mathbf{M_4(2) = \{1,3,4\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,3,4}\}$	**19**
6	$\langle \mathbf{0,1,2,4,3,6}\rangle$	$\mathbf{L_4(3) = \{4\}}$	$\mathbf{M_4(3) = \{1,2,4\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,2,3}\}$	**26**
	$\langle \mathbf{0,3,4,2,1,6}\rangle$	$\mathbf{L_4(1) = \{4\}}$	$\mathbf{M_4(1) = \{1,2,3\}}$	$\{\mathbf{4,6}\}$	$\{\mathbf{1,2,3}\}$	**21**

Tabela 2.16: Exemplo do algoritmo de Ibaraki – Passo (ii) - 4ª iteração.

n_k	I	S	$g(S, n_p)$
2	$\langle 0, 6, 3, 4, 2, 5\rangle$	$\{4, 6\}$	17
	$\langle 0, 6, 3, 4, 1, 2, 5\rangle$	$\{4, 6\}$	17
4	$\langle 0, 6, 3, 1, 4, 5\rangle$	$\{4, 6\}$	19
	$\langle 0, 6, 3, 1, 2, 4, 5\rangle$	$\{4, 6\}$	27
	$\langle 0, 6, 3, 4, 5\rangle$	$\{4, 6\}$	18

Tabela 2.17: Exemplo do algoritmo de Ibaraki – Passo (iii).

2.3.2 Variantes propostas ao algoritmo de Ibaraki

No algoritmo original, tal como ele é descrito em [3], o passo traduzido pela equação (2.18) só pode ser executado quando $L = S$ com n_l adjacente a n_p, para depois seleccionar o caminho de mínimo custo descrito por $g(S, n_p)$.

Conhecido o caminho mais curto do conjunto (L, M, n_l), quando $L = S$, o caminho total pode ser obtido a partir da concatenação do sub-caminho inicial, representado por $I_{n_1-n_l}$, com o sub-caminho mais curto de n_l até n_p, $I_{n_l-n_p}$, ou seja, $I_{n_1-n_l} \diamond I_{n_l-n_p}$. Esta observação conduziu à primeira alteração proposta ao algoritmo: uma vez obtido $L = S$, determina-se o caminho mais curto desde n_l até n_p na rede sem os nós $n_1 \cup L \cup M \setminus \{n_l\}$ para que seja garantido que o caminho resultante de n_1 a n_p é elementar.

Seja $P_{n_l} = \{I : I = I_{n_1-n_l} \diamond I_{n_l-n_p} : N_{I_{n_1-n_l}} \cap S = S \wedge N_{I_{n_1-n_l}} \cap N_{I_{n_l-n_p}} = \{n_l\}\}$, isto é o conjunto de todos os caminhos de n_1 para n_p cujos elementos no seu sub-caminho de n_1 até n_l, $n_l \in S$ passam em todos os nós específicos $n_k \in S \setminus \{n_l\}$.

Proposição 1. *Para um grafo $G(N \setminus \{n_p\}, A, f)$ com a função de pesos $f : A \to \mathbb{R}$, seja $\langle n_1, n_2, ..., n_l\rangle$ o caminho mais curto de n_1 para n_l, $n_l \in S$, e que passa por um conjunto de nós específicos, $S \setminus \{n_l\}$. Seja $p' = \langle n_l, n_{l+1}, ..., n_p\rangle$ o caminho mais curto do nó n_l para o nó n_p numa rede sem os nós $n_1 \cup L \cup M \setminus \{n_l\}$. Pode afirmar-se que o mais curto de n_1 para n_p, entre todos os caminhos pertencentes a P_{n_l}, resulta da concatenação $I_{n_1-n_l} \diamond I_{n_l-n_p}$.*

Na sequência do raciocínio anterior foi feita mais uma alteração a este algoritmo. Quando se obtém um caminho do conjunto (L, M, n_l), com $|L| = |S| - 1$, identifica-se o nó obrigatório em falta conhecido por *missing node*, $\{n_m\} = S \setminus L$ e calcula-se o caminho mais curto desde n_l até ao n_m na rede em que foram removidos os nós $L \cup M \cup \{n_p\} \setminus \{n_l\}$, ou seja calcula-se o caminho $I_{n_l-n_m}$. Se este caminho existir, $I_{n_1-n_l} \diamond I_{n_l-n_m}$, será o caminho de menor custo no conjunto representado por (S, M, n_m). Em seguida remove-se o nó n_l e os nós intermédios pertencentes ao caminho $I_{n_l-n_m}$, reintroduz-se o nó n_p e, finalmente, calcula-se o caminho mais curto desde n_m até n_p. Quando este caminho não existe segue-se para a iteração seguinte do algoritmo de Ibaraki até que $L = S$, e calcula-se o caminho mais curto de n_l para n_p de acordo com a primeira modificação. Para além destas duas alterações, deve-se ter conta, apesar de ainda não ter sido referido, que para um dado conjunto de caminhos

(L, M, n_l) apenas são armazenados os caminhos de menor custo actual. Desta forma reduz-se a quantidade de memória utilizada e é expectável que o tempo de CPU diminua.

Em ambas as modificações acima descritas é necessário um algoritmo que permita a determinação do caminho mais curto entre dois nós de uma rede. Para tal usa-se o algoritmo de Dijkstra que tem sido usado há algumas décadas numa gama vasta de contextos. O algoritmo de Dijkstra encontra o caminho mais curto de um dado nó origem para um outro nó destino assumindo que os arcos têm custos não negativos. Foi implementado o algoritmo de Dijkstra com a vertente de *binary heap*.

Estas alterações são mantidas aquando da implementação do código para a construção do caminho maximamente disjunto nos nós ou nos arcos.

O armazenamento dos caminhos candidatos é feito através de uma tabela de elementos, cada um com a seguinte informação: os conjuntos L, M, o nó n_l, o custo do caminho até n_l assim como a posição na tabela do caminho que lhe deu origem. Em cada iteração do algoritmo é necessária a informação de L e M, e caso não fosse armazenado teria de ser construída em cada passo iterativo, pelo que neste caso optou-se por utilizar mais memória com o objectivo de reduzir o tempo de processamento.

2.4 Conclusão

O algoritmo proposto por Ibaraki (descrito em detalhe na sub-secção 2.3.1) é a base de todo este trabalho. Tratando-se de um algoritmo exacto mostrou que seria necessário um elevado processamento mesmo para redes de pequenas dimensões. Partindo deste conhecimento procurou-se diminuir a quantidade de memória a ser utilizada pela redução do número de sub-caminhos candidatos com a aplicação do algoritmo de Dijkstra em fases distintas do algoritmo, formalizando na sub-secção 2.3.2 duas variantes desse algoritmo que serão incorporadas no cálculo de um caminho com protecção descrito no capítulo 3.

Capítulo 3

Caminho de Protecção

Uma maneira simples de recuperar aquando da ocorrência de falhas é usando um esquema de protecção. A rede pode ser corrompida por diversos factores como catástrofe natural, cortes, mal formações no equipamento e até mesmo devido à necessidade de fazer manutenção à rede. Segundo Kuipers em [12] são necessários três componentes para a sobrevivência da rede:

a) Conectividade da rede, isto é, a rede deve ser bem interligada;

b) Aumento da rede, isto é, novos arcos devem ser adicionados para aumentar a conectividade da rede;

c) Caminho de protecção, ou seja, o procedimento para encontrar um caminho alternativo em caso de falha.

O componente escolhido nesta dissertação para a sobrevivência da rede foi o caminho de protecção global. O caminho de protecção pode ser configurado para proteger contra uma falha num arco ou num nó. Desde que os caminhos de protecção sejam pré-calculados, não existe um atraso significativo no tráfego da rede. Dependendo da altura do cálculo do CP, isto é, antes ou depois do cálculo do CA, a sobrevivência da rede pode ser avaliada por técnicas de protecção ou reencaminhamento:

a) Protecção: é um esquema pró-activo onde os CPs são calculados e guardados previamente. No caso 1:1, o tráfego é recalculado para o CP a partir do momento em que é detectada uma falha no CA. No caso 1+1, o tráfego é duplicado e enviado ao mesmo tempo pelos dois caminhos, CA e CP;

b) Esquema de reencaminhamento: é um mecanismo reactivo que apenas procura resolver a ausência de falha quando detectada e, portanto, o CP não é conhecido *a priori*.

Resumidamente, a protecção exige um tempo menor de recuperação face ao reencaminhamento, mas é menos eficiente no que diz respeito à capacidade utilizada. O reencaminhamento é considerado então mais flexível e eficiente quanto à quantidade de recursos utilizados, mas com um maior tempo de recuperação e sem garantia de que é possível recuperar.

Para além do esquema de recuperação utilizado existem três técnicas para a sobrevivência da rede:

a) Protecção global do caminho que se baseia no cálculo prévio de um caminho disjunto nos nós ou nos arcos;

b) Protecção local (ao arco ou ao nó) que tem pré-atribuída uma rota local que deverá ser usada sempre que o arco ou nó falha;

c) Protecção de um segmento do caminho que consiste num compromisso entre a técnica de protecção do caminho e a técnica de protecção local;

Nesta dissertação o estudo incide sobre o tipo a) nos dois casos, isto é, CP disjunto nos nós ou nos arcos. Para além disto serão também estudados CPs maximamente disjuntos nos nós ou nos arcos. Sendo os caminhos disjuntos nos nós, tanto as falhas isoladas nos nós como nos arcos podem ser recuperadas; quando os caminhos são apenas disjuntos nos arcos, apenas a recuperação no caso de falhas isoladas nos arcos é garantida. A capacidade de uma rede se manter operacional quando um ou mais componentes da rede falham é indispensável tendo em vista a importância dos sistemas de comunicação nos dias de hoje. Neste trabalho a resiliência da rede é assegurada pela existência de um caminho de protecção para o caminho activo. A prioridade é ter sempre um caminho de protecção disponível para o caminho activo, ou seja, a determinação do caminho activo é feita com a condição de que existe um caminho de protecção, ou quando isto não é possível por um caminho maximamente disjunto (nos nós ou nos arcos, dependendo do grau de resiliência desejado).

O caminho activo pode não ser o caminho elementar mais curto entre dois nós n_1 e n_p que passa em elementos específicos, mas sim, o caminho elementar mais curto para o qual é possível ter um caminho disjunto, ou seja, procura-se em primeira instância ter um par de caminhos disjuntos nos nós e se tal não for possível é dada a oportunidade ao utilizador de querer encontrar um par de caminhos disjuntos nos arcos. Finalmente, o utilizador terá a possibilidade de procurar um par de caminhos maximamente disjuntos nos nós ou nos arcos.

O caminho mais curto resulta das variantes desenvolvidas para o algoritmo de Ibaraki na secção 2.3.2 na página 23, e portanto pode ser o resultado de uma das seguintes concatenações:

1. $I_{n_1-n_l} \diamond I_{n_l-n_p}$
2. $I_{n_1-n_l} \diamond I_{n_l-n_m} \diamond I_{n_m-n_p}$

Seja $P_{n_1-n_p}$, o conjunto de todos os caminhos de n_1 para n_p sem restrições. Defina-se o seguinte conjunto de pares de caminhos de n_1 para n_p:

$$\dot{P}_{n_1-n_p} = \{(I, I') : N_I \cap N_{I'} = \{n_1, n_p\}, I \in P^S_{n_1-n_p}, I' \in P_{n_1-n_p}\} \tag{3.1}$$

Relembre-se que N_I é o conjunto dos nós utilizados pelo caminho I e que $P^S_{n_1-n_p}$ foi definido na equação (2.1).

O problema PCAE-DN pode agora ser formalizado:

$$(I^*, I'^*) = \arg \min_{(I,I') \in \dot{P}_{n_1-n_p}} f(I) \tag{3.2}$$

O caminho I'^* de menor custo será determinado uma vez conhecido I^*, ou seja, só interessa minimizar o custo do CA (sendo que um caminho é considerado activo se e só se tiver um CP associado).

Defina-se o conjunto

$$\bar{P}_{n_1-n_p} = \{(I, I') : A_I \cap A_{I'} = \emptyset, I \in P^S_{n_1-n_p}, I' \in P_{n_1-n_p}\} \tag{3.3}$$

em que A_I, $A_{I'}$ são o conjunto das arestas dos caminhos I e I', respectivamente.

O problema PCAE-DA pode agora ser formalizado:

$$(I^*, I'^*) = \arg \min_{(I,I') \in \bar{P}_{n_1-n_p}} f(I) \tag{3.4}$$

e, tal como anteriormente, o caminho I'^* de menor custo será determinado uma vez conhecido I^*.

3.1 Protecção ao nó

Para a determinação de um caminho disjunto do CA nos nós utiliza-se o seguinte método: à medida que o caminho activo vai sendo calculado, verifica-se se é possível encontrar um caminho alternativo, caminho de protecção, com os nós que não foram utilizados previamente, à excepção da origem, n_1 e do destino n_p.

O caminho alternativo tem de ser calculado numa rede diferente da original, numa rede em que não existam os nós intermédios do caminho activo, o primeiro calculado. Se este caminho não existir, o sub-caminho (ou caminho) candidato a CA é descartado, ou seja, se para um sub-caminho $I_{n_1-n_l}$, candidato a ser parte integrante de CA, não é possível obter um caminho de n_1 para n_p disjunto do CA com os nós no conjunto $N_{I_{n_1-n_l}} \setminus \{n_1\}$, esse sub-caminho é abandonado. Note-se que isto equivale a considerar que nos conjuntos (L, M, n_l) e (S, n_p) apenas são admissíveis os sub-caminhos para os quais existe um caminho de protecção disjunto do CA nos nós. Ou seja, no caso de um caminho $I_{n_1-n_l} \in (L, M, n_l)$

verifica-se se existe algum caminho de n_1 para n_p no sub-grafo de G induzido pelo conjunto de nós $N \backslash L \cup M$. A selecção, em cada iteração, dos caminhos de acordo com a equação (2.17) continua correcta pois $L \cap M$ é igual para todos os caminhos candidatos que terminam em n_l, pelo que ou são todos admissíveis ou nenhum o é.

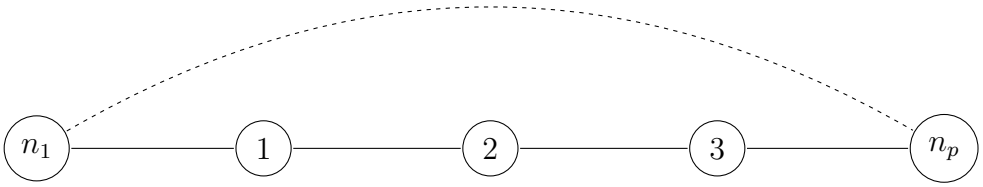


Figura 3.1: Determinação de caminho de protecção disjunto do CA nos nós: caminho de protecção a tracejado.

3.2 Protecção ao arco

Para a determinação de um caminho disjunto do CA nos arcos a abordagem tem por base a mesma premissa que no caso dos nós: à medida que vai sendo construído o caminho mais curto procura-se encontrar um caminho alternativo, o CP, mas neste caso são removidos os arcos que não foram utilizados previamente. O caminho alternativo é então calculado numa rede em que não existam os arcos presentes no caminho activo. Se este caminho não existir, o sub-caminho ou caminho candidato a CA é descartado tal como na secção 3.1. No caso da protecção ao arco, um caminho só é admissível se for possível proteger os arcos $A_{I_{n_1-n_l}}$ ao escolher entre todos os caminhos presentes em (L, M, n_l) aquele que tem menor custo.

3.3 Caminho de protecção maximamente disjunto

Este tipo de protecção requer ainda um maior consumo de memória que a determinação de um par de caminhos totalmente disjuntos porque o número de caminhos candidatos que podem ser descartados *a priori* será menor. O cálculo do CA é feito da mesma maneira que nos casos descritos anteriormente, ou seja como a concatenação de dois ou três sub-caminhos considerados admissíveis (isto é, aqueles para os quais é possível construir um caminho desde n_1 a n_p).

a) $I_{n_1-n_l} \diamond I_{n_l-n_p}$

b) $I_{n_1-n_l} \diamond I_{n_l-n_m} \diamond I_{n_m-n_p}$

3.3.1 Caminho de protecção maximamente disjunto nos nós

O CA para o qual é possível calcular um CP maximamente disjunto nos nós é calculado léxico-graficamente de acordo com os seguintes critérios:

1. Menor número de nós em comum no par de caminhos;

2. Menor número de arcos em comum no par de caminhos;

3. Menor custo do caminho activo.

Formalmente, isto corresponde a resolver um problema multi-critério lexicográfico para o qual se definem as seguintes funções objectivo:

$$z_1(I, I') = |N_I \cap N_{I'} \setminus \{n_1, n_p\}| \tag{3.5}$$
$$z_2(I, I') = |A_I \cap A_{I'}| \tag{3.6}$$
$$z_3(I, I') = f(I) \tag{3.7}$$

A função z_1 devolve o número de nós intermédios em comum entre os caminhos I e I'; a função z_2 o número de arcos em comum e, finalmente, a função z_3 corresponde ao custo do caminho I. Cada um dos problemas de optimização é resolvido sequencialmente [13]:

$$\min_{I \in P^S_{n_1-n_p}, I' \in P_{n_1-n_p}} z_i(I, I') \tag{3.8}$$

sujeito a $I \neq I'$, $z_j(I, I') \leq z_j(I_j^*, I_j'^*), j = 1, \cdots, i-1, i = 1, 2, 3$ em que z_i é a i-ésima função objectivo e i representa a ordem de importância de uma função na sequência de preferência. O par de caminhos $(I_j^*, I_j'^*)$ representa o argumento correspondente ao valor óptimo da j-ésima função obtida da j-ésima iteração.

O PCAE-MDA pode ser formalizado de forma semelhante ao PCAE-MDN, bastando suprimir a função z_1.

Para ilustrar o cálculo do caminho de protecção considerou-se o exemplo descrito no capítulo 2 na secção 2.3.1. Relembra-se o problema associado à rede da figura 2.5 com $n_1 = 0$, $n_p = 5$, $S = \{4\}$ e $D = \{(0, 3)\}$.

Tendo em conta que se pretende um par de caminhos maximamente disjuntos nos nós, de cada vez que se quer testar a existência de protecção cada nó intermédio n_i do sub-caminho candidato a caminho activo é dividido em dois: um nó emissor, $n_i^{''}$ e um nó receptor $n_i^{'}$; esses dois nós são ligados por um arco dirigido de $n_i^{'}$ para $n_i^{''}$ de custo M, sendo M um valor suficientemente grande; todos os arcos anteriormente incidentes em n_i são agora incidentes em n_i' e todos os arcos emergentes de n_i são agora emergentes de $n_i^{''}$. Finalmente todos os arcos têm o seu custo adicionado de M. Na figura 3.2 tem-se um esboço do procedimento para aquele que irá ser a solução do problema. Uma vez resolvido o problema para um sub-caminho ou caminho candidato, é necessário fazer a fusão dos nós $n_i^{'}$ e $n_i^{''}$ novamente no nó n_i que lhe deu origem. No exemplo, o caminho activo é $\langle 0, 3, 4, 5\rangle$ com custo 18 e o de protecção $\langle 0, 1, 2, 5\rangle$ com custo 16.

O cálculo do caminho de protecção é feito por três vezes ao longo do algoritmo para cada sub-caminho (ou caminho) candidato a caminho activo. De forma cíclica, verifica-se se o caminho activo até ao momento tem um CP associado. Note-se que se o CP for igual ao CA a proteger, o CA não é admissível; se existir um CA e respectivo caminho de protecção já encontrados que são lexico-graficamente uma melhor solução que obtida até ao momento, o caminho ou sub-caminho candidato é descartado. Sempre que um caminho é descartado passa-se para o elemento seguinte da tabela de sub-caminhos candidatos. O método está descrito em pseudo-código na página 31. O algoritmo 1 traduz o procedimento responsável pela construção dos sub-caminhos derivados de cada elemento de (L, M, n_l).

Na linha 35 um sub-caminho (e o seu CP) só é guardado se este e o CP correspondente não forem, tanto quanto é possível avaliar neste momento, lexico-graficamente piores do que o melhor par de caminhos actualmente armazenado. Adicionalmente esse caminho (e o seu CP) só será efectivamente armazenado se for lexico-graficamente superior a todos os elementos do mesmo conjunto, (L, M, n_l), já calculados.

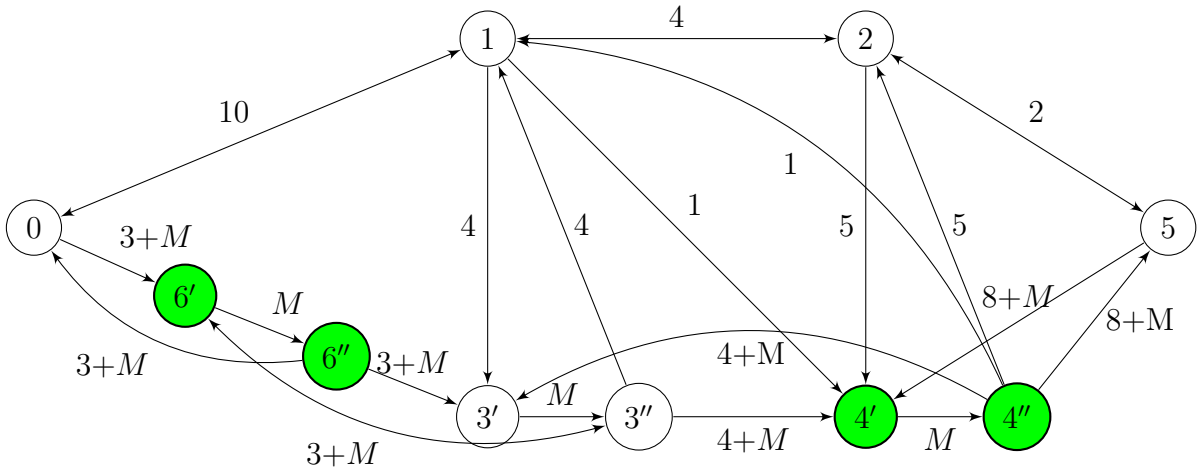


Figura 3.2: Esboço da rede fictícia para o cálculo do caminho maximamente disjunto nos nós para a rede da figura 2.5.

3.3.2 Caminho de protecção maximamente disjunto nos arcos

À semelhança do que é feito para o cálculo do par de caminhos maximamente disjuntos nos nós, a determinação do par de caminhos maximamente disjuntos nos arcos segue uma lista de prioridades:

1. Menor número de arcos em comum com o caminho activo candidato;
2. Menor custo do caminho activo.

A ideia subjacente à construção de um caminho maximamente disjunto do CA nos arcos é a mesma que no maximamente disjunto nos nós, mas neste caso é apenas necessário adicionar

Algoritmo 1: Rotina principal responsável pela construção do caminho activo e de protecção maximamente disjuntos nos nós.

Entrada: Tabela de caminhos de um elemento de (L, M, n_l) e valor conhecido para $z_i, (i = 1, 2, 3)$ que irá dar origem a novos caminhos e melhor par corrente (CA',CP')

Saída: Novos elementos inseridos na tabela de caminhos

```
1  if Candidato corrente não é para já pior que o par previamente guardado then
2      while não percorre todos os arcos emergentes de n_l do
3          if |L| igual a |S| − 1 then
4              Identifica-se o nó {n_m} = S \ L;
5              Cálculo do caminho I_{n_l−n_m};
6              if O caminho I_{n_l−n_m} não existe then
7                  Termina o algoritmo, não há solução para este caminho;
8              Cálculo do caminho I_{n_m−n_p};
9              if Verifica se o caminho I_{n_l−n_m} ⋄ I_{n_m−n_p} existe then
                   // O CA é I_{n_1−n_l} ⋄ I_{n_l−n_m} ⋄ I_{n_m−n_p}
10                 Cálculo do CP maximamente disjunto;
11                 if O CP existe e é diferente do activo then
12                     if O par é melhor que o previamente guardado then
13                         Actualiza CA' e CP' e o número de arcos e nós partilhados;
14             else
15                 Cálculo do caminho I_{n_m−n_p} na rede sem os nós N_{I_{n_1−n_l}};
16                 if O caminho I_{n_m−n_p} não existe then
17                     Termina o algoritmo;
18         else if |L| igual a |S| then
19             if O arco (n_l, n_p) existe then
                   // O CA é I_{n_1−n_l} ⋄ ⟨n_l, n_p⟩
20                 Cálculo do CP maximamente disjunto;
21                 if O CP existe e é diferente do CA then
22                     if O par é melhor que o previamente guardado then
23                         Actualiza CA' e CP' e o número de arcos e nós partilhados;
24                         Termina o algoritmo;
25             Cálculo do caminho I_{n_l−n_p} ;
26             if O caminho I_{n_l−n_p} existe then
                   // O CA é I_{n_1−n_l} ⋄ I_{n_l−n_p}
27                 Cálculo do CP maximamente disjunto;
28                 if O CP existe e é diferente do CA then
29                     if O par é melhor que o previamente guardado then
30                         Actualiza CA' e CP' e o número de arcos e nós partilhados;
31                         Termina o algoritmo;
32             else
33                 O CA não existe e termina o algoritmo.
           // O arco corrente é (n_l, n'_l)
34         if Sub-caminho I_{n_1−n'_l} não é para já pior que o previamente guardado then
               // I_{n_1−n'_l} ∈ (L ∪ {n_l}, M, n_l) ou I_{n_1−n'_l} ∈ (L, M ∪ {n_l}, n_l)
35             Guarda I_{n_1−n'_l} (e CP) selectivamente na tabela de caminhos;
36     end
```

Algoritmo 2: Rotina responsável pelo cálculo do caminho de protecção maximamente disjunto nos nós.

Entrada: Conjunto de nós e arcos do CA.
Saída: Caminho e custo de protecção, número de nós e arcos partilhados.

```
1  Mudar o custo dos arcos do caminho activo para o seu valor incrementado de um
   valor suficientemente grande M;
2  Criação dos nós fictícios para cada nó intermédio do CA em que o respectivo arco tem
   custo M, representados na figura 3.2;
3  Cálculo do caminho mais curto de n_1 para n_p com o algoritmo de Dijsktra;
4  Reposição da rede e dos custos originais;
5  if custo do caminho = ∞ then
       // Neste caso, não há CA
6      return caminho de protecção vazio;
7  else if custo do caminho ≤ M then
       // O CP é disjunto com os elementos dados do CA.
8      return o caminho de protecção com o mapa de elementos partilhados vazio;
9  else
       // Existem arcos em comum
10     return caminho de protecção com informação dos elementos partilhados entre CA
       e CP;
```

um custo M (valor suficientemente grande) aos arcos de um sub-caminho candidato a CA. Assim, o CP usará o menor número de arcos pertencentes ao CA. Por exemplo, para o mesmo caso que vem a ser estudado desde o capítulo anterior o CA é: $\langle 0, 3, 4, 2, 5\rangle$ – ver figura 3.3 – de custo 17 e o CP é $\langle 0, 1, 4, 5\rangle$ de custo 19 existindo portanto a partilha de um único nó, ou seja, os caminhos são disjuntos nos arcos. Tal como no caso anterior, o cálculo do CP é feito por três vezes ao longo do algoritmo. O procedimento é o mesmo que no caso dos nós com a diferença de como são feitas as modificações na rede, e portanto o pseudo-código é também válido para os arcos – ver algoritmo 1. Para analisar a construção do CP pode-se observar o pseudo-código do algoritmo 3.

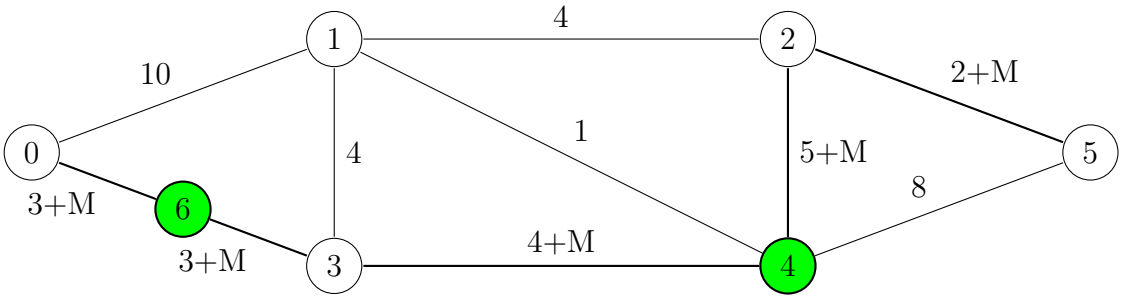


Figura 3.3: Esboço da rede fictícia para o cálculo do caminho maximamente disjunto nos arcos para a rede 2.5.

Algoritmo 3: Rotina responsável pelo cálculo do caminho de protecção maximamente disjunto nos arcos.

Entrada: Conjunto de nós e arcos do CA.
Saída: Caminho e custo de protecção, número de nós e arcos partilhados.

```
1 Mudar o custo dos arcos do caminho activo para o seu valor incrementado de um
  valor suficientemente grande M;;
2 Cálculo do caminho mais curto de n_1 para n_p com o algoritmo de Dijsktra;;
3 Reposição dos custos originais;;
4 if custo do caminho = ∞ then
      // Neste caso, não há CA
5     return caminho de protecção vazio;
6 else if custo do caminho ≤ M then
      // O CP é disjunto com os arcos dados do CA.
      // Não existem arcos em comum, mas podem existir nós.
7     return caminho de protecção com informação dos elementos partilhados entre CA
      e CP;
8 else
      // Existem arcos em comum
9     return caminho de protecção com informação dos elementos partilhados entre CA
      e CP;
```

Apenas é apresentado o algoritmo para o cálculo de um par de caminhos maximamente disjuntos porque este é o algoritmo mais elaborado. Os algoritmos para a obtenção do par de caminhos disjuntos nos nós ou nos arcos diferem deste apenas no mecanismo de cálculo do CP.

Capítulo 4

Análise de Desempenho

Os testes realizados para a avaliação do desempenho dos algoritmos desenvolvidos foram realizados usando a biblioteca de redes, [14], com as redes identificadas na tabela 4.1. O computador utilizado é detentor de um processador Intel Core I7 com CPU@3,07GHz × 8 de 64 bits.

SNDlib - Survivable Network Design Library					
Nome da Rede	Núm. nós	Núm. arcos	Mín. grau	Máx. grau	Grau médio
abilene	12	15	1	4	2.50
atlanta	15	22	2	4	2.93
france	25	45	2	10	3.60
geant	22	36	2	8	3.27
janos-us	26	84	4	10	6.46
newyork	16	49	2	11	6.12
nobel-eu	28	41	2	5	2.93
nobel-germany	17	26	2	6	3.06
nobel-us	14	21	2	4	3.00
norway	27	51	2	6	3.78
polska	12	18	2	5	3.00

Tabela 4.1: Designação e respectivas características das redes usadas na análise de desempenho.

Neste capítulo apenas são apresentadas as figuras com os resultados de 3 redes: Atlanta, Janos-us e Norway. Os restantes resultados estão em apêndice uma vez que os aqui mostrados são ilustrativos do comportamento padrão dos algoritmos. Para cada rede é estudado o comportamento de dois algoritmos distintos: primeira variante do algoritmo de Ibaraki (IbPS) e segunda variante do algoritmo de Ibaraki (IbPS-1). As duas rotinas foram usa-

das para resolver o PCAE-DN e o PCAE-DA. No caso do cálculo do caminho de protecção maximamente disjunto do CA (o PCAE-MDN e o PCAE-MDA) foi resolvido apenas com o recurso do IbPS-1, sendo os respectivos algoritmos designados por IbPS-1MDn e IbPS-1MDa. Nestes últimos dois algoritmos apenas foi usado o IbPS-1 porque resulta num menor consumo de memória. Foram considerados vinte pares origem e destino distintos (gerados aleatoriamente) para cada conjunto de elementos específicos considerado. O número de elementos específicos considerado foi escolhido levando em conta um número indicado pela PT-Inovação. Na tabela 4.2 encontra-se a chave que permite interpretar as figuras com os resultados.

Eixo dos xx	Significado
1N_1A	1 nó e 1 arco obrigatórios
1N_2A	1 nó e 2 arcos obrigatórios
1N_3A	1 nó e 3 arcos obrigatórios
2N_1A	2 nós e 1 arco obrigatórios
2N_2A	2 nós e 2 arcos obrigatórios
3N_1A	3 nós e 1 arco obrigatórios

Tabela 4.2: Tabela com os significados do eixo *xx*.

Foi recolhido o tempo médio de CPU com base nas vinte experiências e também o número médio de elementos na tabela que representa cada problema a resolver e o qual traduz a utilização da memória pelo algoritmo. As barras de erro em torno da média correspondem ao menor e ao maior valor no conjunto das vinte experiências. Não foi utilizado o desvio padrão porque não sendo muito elevado o número de amostras e apresentando estas uma grande variabilidade, considerou-se mais interessante apresentar a gama de variação absoluta.

A segunda variante proposta tem o comportamento esperado face à primeira variante: utiliza pouco mais tempo de CPU, mas requer menos memória, como se pode ver por exemplo nas figuras 4.1, 4.2, 4.4, 4.5, 4.7 e 4.8. Pode igualmente observar-se que, em geral, o cálculo de um par de caminhos disjuntos nos nós utiliza menos tempo de CPU e menos memória que o cálculo de caminhos disjuntos nos arcos. Isso resulta do facto do primeiro problema ser mais restritivo que o segundo conduzindo à eliminação de um maior número de caminhos candidatos, devido à impossibilidade de os proteger.

O cálculo de um par de caminhos maximamente disjuntos nos nós consome mais tempo de CPU e também mais memória do que a resolução de um par completamente disjunto nos nós ou nos arcos. Recorde-se que a determinação de um par de caminhos maximamente disjuntos requer a divisão dos nós, pelo que obriga, para cada caminho candidato em construção, a uma transformação na rede antes do cálculo do caminho de protecção tornando-o mais pesado.

Os tempos de resolução no caso da rede Atlanta são inferiores a 1 segundo em todos os

casos. No caso da rede Janos-us embora os tempos médios rondem 1 ou 2 segundos (depende de o problema ser PCAE-DN ou PCAE-DA) e no pior caso é próximo dos 8 segundos. Na rede Norway os tempos de CPU são significativamente mais elevados. Por exemplo, para o caso em que há um nó e três arcos obrigatórios, o tempo de CPU para a variante menos eficiente é em média cerca de 2,7 minutos e 3,7 minutos para a resolução dos problemas PCAE-DN e PCAE-DA, respectivamente. A segunda variante implementada conduz a tempos ligeiramente inferiores: 1,2 minutos e 1,5 minutos para os mesmos problemas. Note-se que a resolução dos problemas de cálculo de pares de caminhos maximamente disjuntos nos nós e nos arcos requer neste caso 6,7 minutos e 3,8 minutos, respectivamente.

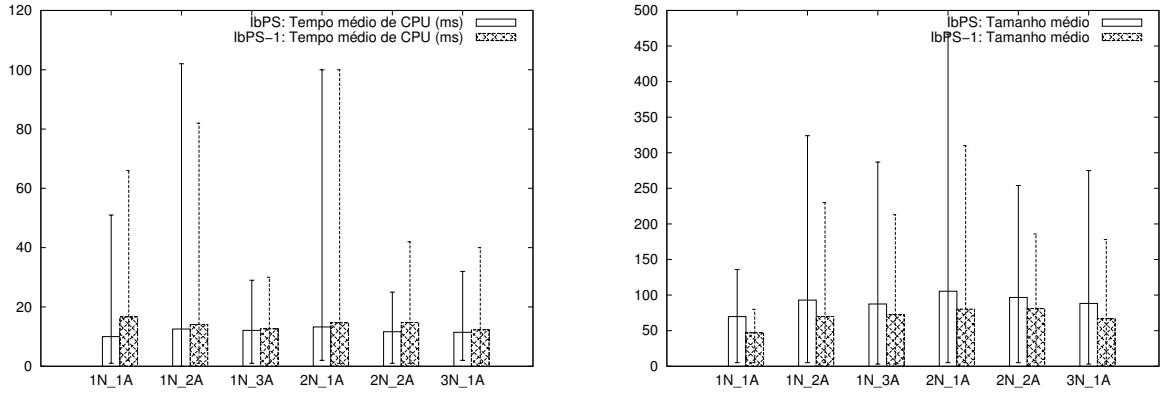
Os resultados apresentados aqui e no apêndice A dizem respeito a redes com menos de 30 nós e menos de 100 arcos. Verificou-se que os algoritmos implementados não conseguem resolver em tempo útil problemas em redes de maior dimensão, pelo que será necessário o desenvolvimento de heurísticas especializadas para a resolução destes problemas, mesmo em redes de não muito grande dimensão.

Foi adaptada a formalização proposta em [4] para a resolução do PCE, com as restrições adicionais necessárias à obtenção de um caminho disjunto do CA nos arcos e nos nós – ver apêndice C. Desta forma foi possível confirmar a correcção das soluções obtidas pelos algoritmos implementados para a resolução dos problemas PCAE-DN e PCAE-DA.

No apêndice C encontram-se os tempos de CPU requeridos pelo CPLEX 12.60 [15]. Como se pode observar os tempos de CPU no caso da rede Norway são sempre inferiores a 50 milissegundos. Por conseguinte, fizeram-se algumas experiências para averiguar a dimensão dos problemas que seria viável resolver em tempo útil com o CPLEX, tendo-se verificado que para uma rede com 1000 nós e 1250 arcos é necessário no máximo cerca de 5 minutos para o mesmo número de elementos obrigatórios anteriormente considerados. No caso de uma rede com 4096 nós e 24576 arcos demora cerca de 2 horas. Estas redes estão em [16], designadas por d04 e hc12u, respectivamente.

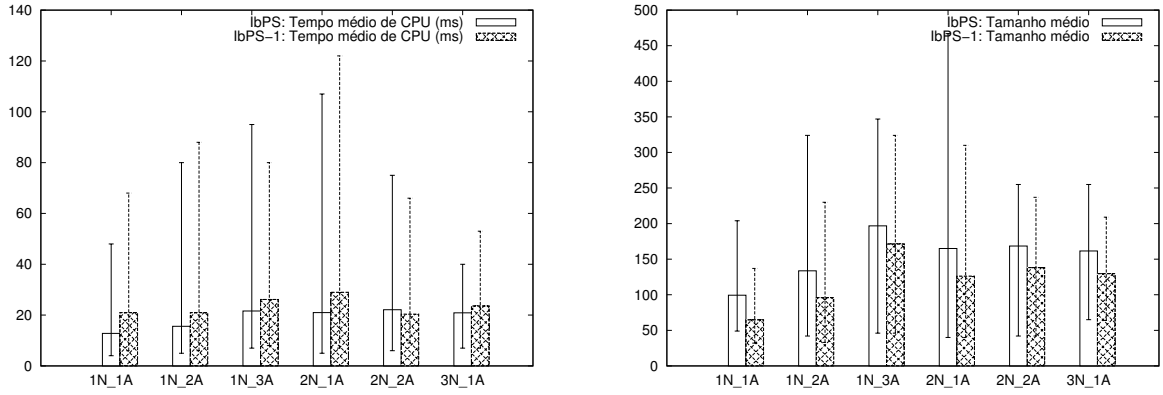
Note-se que quando não existe solução para os problemas PCAE-DN e PCAE-DA o CPLEX detecta rapidamente esse facto (em geral por análise das restrições) enquanto os algoritmos podem necessitar de processamento significativo para detectar a ausência da mesma, o que resulta num valor máximo muito superior ao médio em muitos dos casos.

A adaptação do algoritmo de Ibaraki para a resolução do problema PCAE-DN e PCAE-DA, que permite descartar muitos sub-caminhos candidatos, conduz contudo a tempos de execução elevados. A experiência adquirida indica que é necessária uma estratégia mais eficiente de redução do espaço de pesquisa, obviamente com o ónus de não garantir a obtenção da solução óptima.



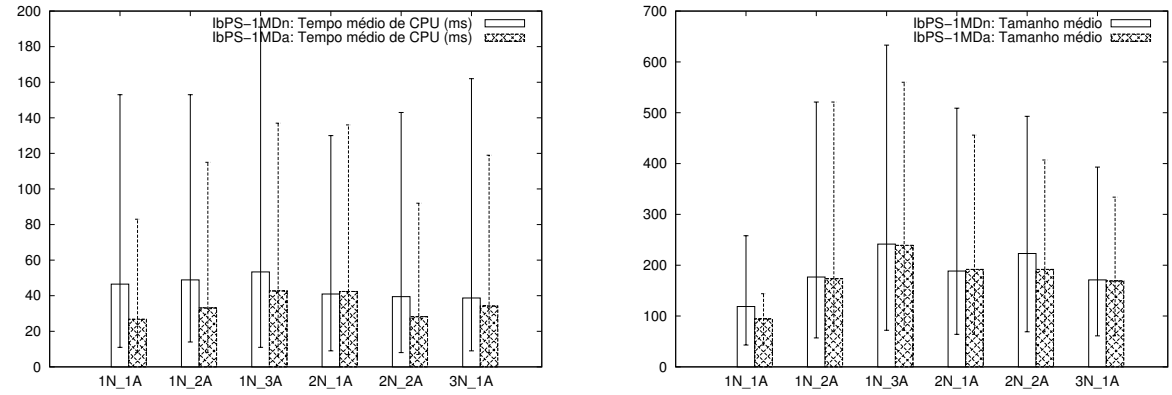
(a) (b)

Figura 4.1: Rede Atlanta: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



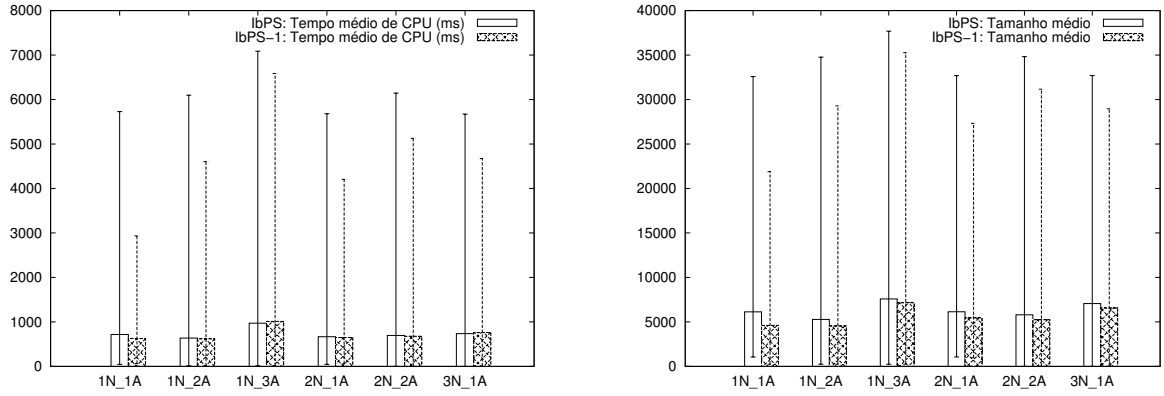
(a) (b)

Figura 4.2: Rede Atlanta: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



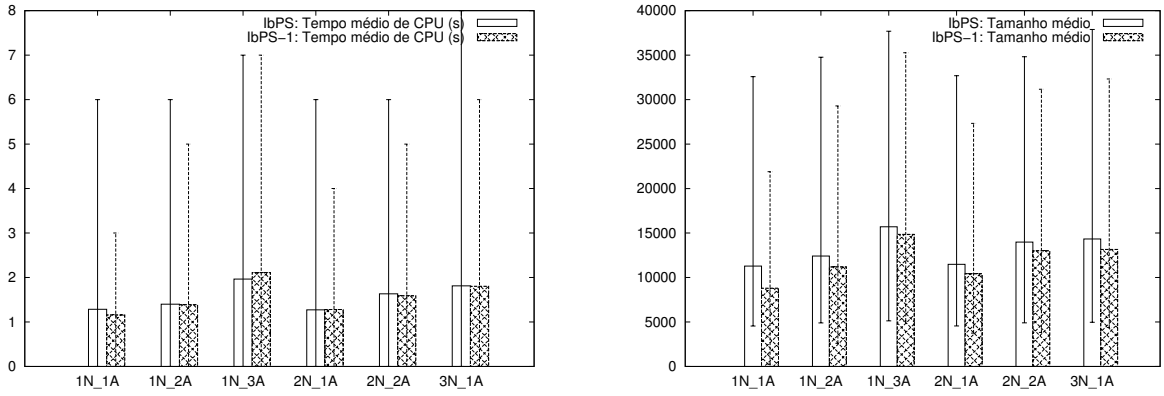
(a) (b)

Figura 4.3: Rede Atlanta: (a) Tempo médio e (b) Tamanho médio.



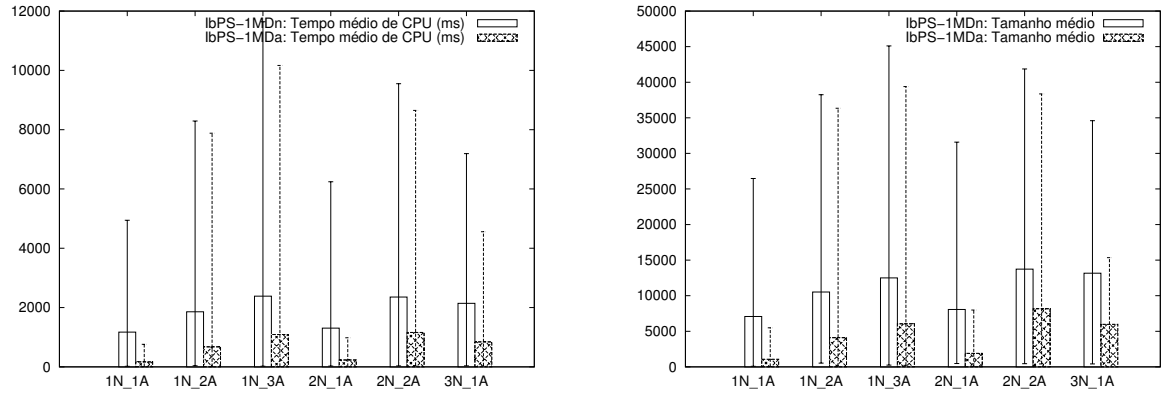
(a) (b)

Figura 4.4: Rede Janos-us: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



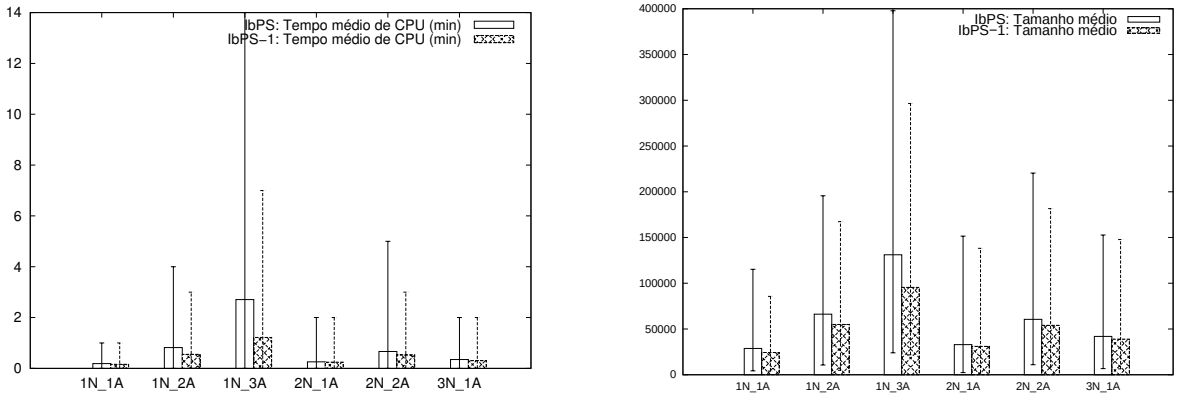
(a) (b)

Figura 4.5: Rede Janos-us: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



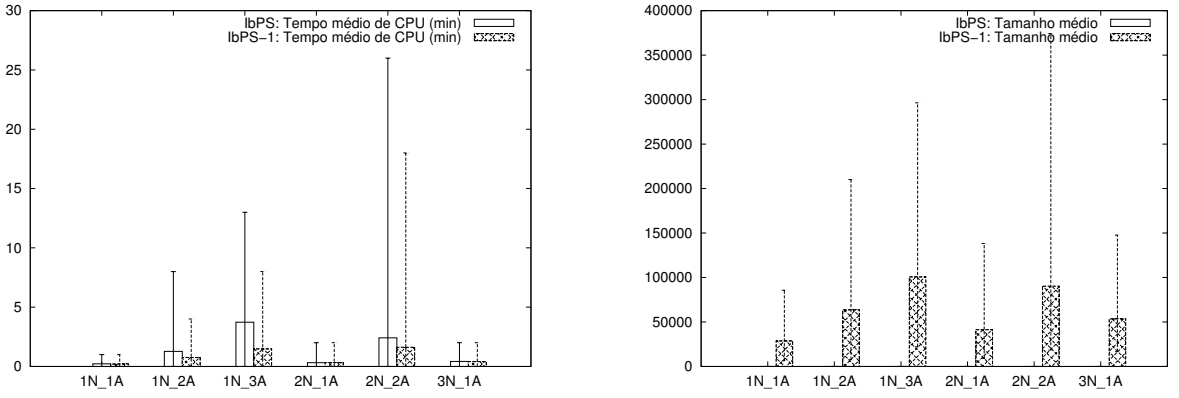
(a) (b)

Figura 4.6: Rede Janos-us: (a) Tempo médio e (b) Tamanho médio.



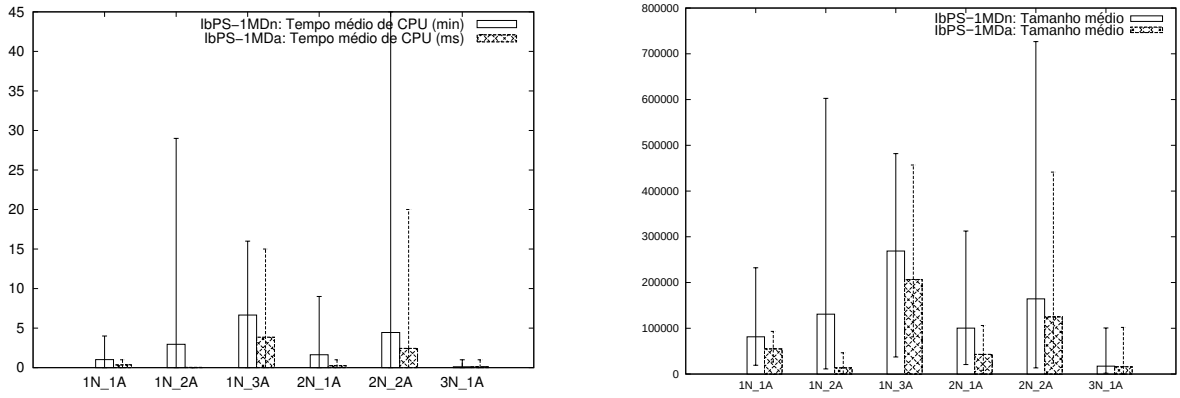
(a) (b)

Figura 4.7: Rede Norway: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



(a) (b)

Figura 4.8: Rede Norway: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



(a) (b)

Figura 4.9: Rede Norway: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.

Capítulo 5

Conclusões e Futuro Trabalho

5.1 Conclusões

Os principais objectivos da presente dissertação foram o desenvolvimento de algoritmos para a resolução do problema do caminho mais curto que passe em elementos (nós e/ou arcos) específicos da rede, com protecção. Para isso foi necessário em primeiro lugar implementar um algoritmo que resolvesse o problema do caminho mais curto que passasse em nós obrigatórios.

No capítulo 2 foram introduzidos os conceitos necessários à compreensão do problema, bem como o estudo feito ao primeiro artigo descoberto na literatura acerca deste tema, [1].

Dreyfus em [2] afirma que o algoritmo de Saksena e Kumar [1] contém um erro que pode impedir a obtenção da solução óptima. No entanto, o contra-exemplo dado por Dreyfus [2] de facto não ilustra o seu ponto de vista uma vez que o algoritmo de Saksena e Kumar [1] resolve correctamente o problema escolhido por Dreyfus. Foi apresentado neste trabalho um exemplo que de facto permitiu confirmar a conclusão de Dreyfus acerca da incorrecção do algoritmo proposto em 2.2.1.

No capítulo 2 na secção 2.3 é feita a descrição do algoritmo que foi a base de todo o trabalho desenvolvido nesta dissertação. O ponto de partida para o cálculo do caminho activo foi o algoritmo de Ibaraki [3]. A primeira tarefa consistiu numa alteração à rede que permitiu transformar arcos obrigatórios em nós obrigatórios. Ciente de que o problema a resolver era NP-difícil procurou-se em primeiro lugar obter uma versão mais eficiente do algoritmo de Ibaraki. A abordagem utilizada foi procurar reduzir o número de sub-caminhos gerados, tendo sido propostas duas variantes. Na primeira variante assim que se detecta que um sub-caminho já visitou todos os elementos obrigatórios, calcula-se o sub-caminho em falta até ao nó destino utilizando o algoritmo de Dijkstra; na segunda variante, quando se detecta que ao sub-caminho a ser construído apenas falta visitar um elemento obrigatório (qualquer), procura-se obter uma solução que terá como sub-caminho final a concatenação de dois auxiliares: o caminho mais curto entre o último nó obrigatório já visitado e o nó

obrigatório em falta e o caminho mais curto entre este e o nó destino.

O facto dos caminhos a obter requererem protecção, fez com que a geração de caminhos candidatos fosse condicionada pela possibilidade do mesmo poder ser protegido. Assim, sempre que um novo sub-caminho candidato era obtido o mesmo só era considerado admissível se não fosse possível provar a impossibilidade de o proteger. Esperava-se que esta restrição levasse a um elevado número de caminhos candidatos descartados, tornando mais eficiente a resolução dos problemas PCAE-DN e PCAE-DA face ao PCAE. E, embora os resultados computacionais mostrem que de facto a determinação de um par de caminhos disjuntos nos nós é mais rápida do que a determinação de um par de caminhos disjuntos nos arcos, o aumento de eficiência conseguido não permite contudo a resolução em tempo útil de problemas em redes de dimensão superior aos considerados neste trabalho.

Foram também desenvolvidos algoritmos para a determinação de pares de caminhos maximamente disjuntos (nos nós e nos arcos), tendo sido seleccionada para o cálculo do CA a segunda variante proposta para o algoritmo de Ibaraki uma vez que utiliza menos memória, sem contudo aumentar significativamente o tempo de CPU. Verificou-se como já se suspeitava que a resolução destes problemas é computacionalmente mais exigente do que os anteriormente referidos, devido a um menor número de caminhos candidatos descartados.

Foram apresentados resultados de experimentação computacional, utilizando onze redes da biblioteca SNDlib [14] com topologias de redes reais. Foi recolhido o tempo médio de CPU e o número médio de elementos na tabela que representa cada problema a resolver, o qual traduz a utilização da memória pelo algoritmo. Foi ainda adaptada a formalização proposta em [4] para a resolução do PCE, com as restrições adicionais necessárias à obtenção de um caminho disjunto nos arcos e nos nós com o CA, o que permitiu confirmar as soluções obtidas pelos algoritmos implementados para a resolução dos problemas PCAE-DN e PCAE-DA.

5.2 Trabalho futuro

Este trabalho mostrou que os algoritmos desenvolvidos para resolver os problemas PCAE-DN, PCAE-DA, PCAE-MDN e PCAE-MDA apenas são viáveis em redes até trinta nós (alguns problemas podem demorar até 7 minutos). Confirma-se assim que um algoritmo exacto tal como os propostos aqui não poderão ser aplicados a redes de maiores dimensões.

Considera-se que a abordagem usada por Saksena e Kumar [1] poderia dar origem a uma heurística eficaz se for considerado que em cada iteração poderiam ser utilizados apenas tantos sub-caminhos candidatos quantos os elementos obrigatórios. Em caso de empate (sub-caminhos com o mesmo custo associados ao mesmo nó obrigatório) seria seleccionado aquele com menor número de elementos. Isto limitaria drasticamente o problema da explosão combinatória inerente à resolução exacta deste problema. Um refinamento desta abordagem seria a inclusão de memória sobre as decisões tomadas, permitindo revisitá-las sempre que não fosse possível obter uma solução. Obviamente que esta abordagem iria sofrer também

da limitação inerente ao algoritmo de Saksena e Kumar [1].

Adicionalmente quer no algoritmo de Ibaraki [3] quer na heurística atrás esboçada considera-se que seria vantajosa e viável a utilização de computação paralela uma vez que no primeiro caso o algoritmo principal se desenha por níveis e no segundo por elementos obrigatórios, sendo em ambos os casos possível dividir o problema em sub-problemas que podem ser resolvidos concorrencialmente.

Apêndice A

Contra-Exemplo Falhado de Dreyfus

No contra-exemplo dado por Dreyfus é usado um grafo representado em A.1. O estudo foi feito com o intuito de descobrir o caminho mais curto de 1 para 5 passando no mínimo por dois nós intermédios quaisquer, ou seja, pode-se interpretar os nós $2, 3$ e 4 como nós específicos. Desta forma, o caminho deverá passar por um dos seguintes conjuntos possíveis para S: $\{2, 3\}$, $\{3, 4\}$ e $\{2, 4\}$. Como se poderá verificar neste anexo, o contra-exemplo falha pois o algoritmo de Saksena e Kumar [1] encontra o caminho óptimo neste caso particular.

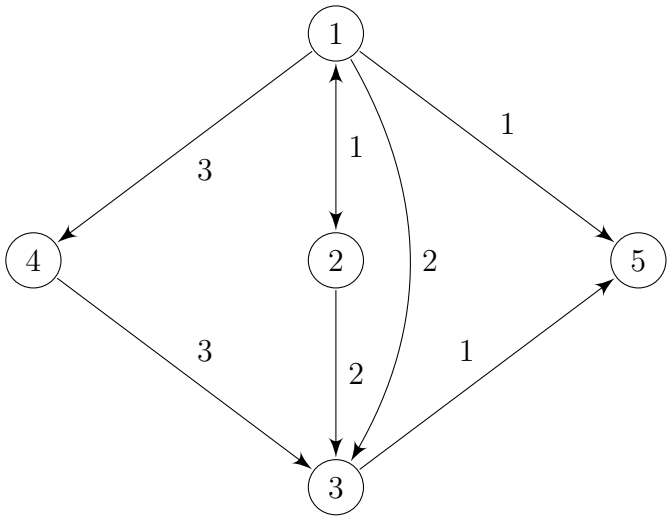


Figura A.1: Rede usada no artigo [2]. Os nós obrigatórios são 2 ,3 e 4.

Tal como na secção anterior, nas tabelas A.1 a A.3 o valor à esquerda é o custo do caminho e os valores entre parêntesis são os novos nós intermédios constituintes do caminho desde n_i até n_l.

$$f^1_{n_i=2} \quad (n_l = 3): \quad D(n_i = 2, n_l = 3) + f^0_3 = 2 + 1 = 3 \qquad \langle 2, 3, 5 \rangle \tag{A.1}$$

$$(n_l = 4): \quad D(n_i = 2, n_l = 4) + f^0_4 = 4 + 4 = 8 \qquad \langle 2, 1, 4, 3, 5 \rangle \tag{A.2}$$

$D^r(n_i, n_l)$			
n_i	$n_l = 2$	$n_l = 3$	$n_l = 4$
1	1	2	3
2	-	2	4 (1)
3	∞	-	∞
4	∞	∞	-

Tabela A.1: Tabela com os dados $D^r(n_i, n_l)$ referente à rede da figura A.1.

n_i	$f^0_{n_i}$
2	2 (1)
3	1
4	4 (3)

Tabela A.2: Tabela com os dados $f^0_{n_i}$ referente à rede da figura A.1.

Não é realizado o cálculo $f^1_{n_i=3}$ porque $D(3, n_l) = \infty$ para $n_l = 2, 4$.

$$f^1_{n_i=4} \quad (n_l = 2): \quad D(n_i = 4, n_l = 2) + f^0_2 = \infty + 2 = \infty \tag{A.3}$$

$$(n_l = 3): \quad D(n_i = 4, n_l = 3) + f^0_3 = 3 + 1 = 4 \qquad \langle 4, 3, 5 \rangle \tag{A.4}$$

Na tabela A.3 estão agrupados os valores de $f^1_{n_i}$ resultantes dos cálculos anteriores.

$f^1_{n_i}$				
n_i	$n_l = 2$	$n_l = 3$	$n_l = 4$	$\min(n_l = 2, n_l = 3, n_l = 4)$
2	-	3	8 (1)	3
3	∞	-	∞	∞
4	∞	4	-	4 (3)

Tabela A.3: Tabela com os dados $f^1_{n_i}$ referente à rede da figura A.1.

Seguem-se os cálculos detalhados de $f^2_{n_i}$ e a tabela A.4 resultante.

$$f^2_{n_i=2} \quad (n_l = 3): \quad D(n_i = 2, n_l = 3) + f^1_3 = 2 + \infty = \infty \tag{A.5}$$

$$(n_l = 4): \quad D(n_i = 2, n_l = 4) + f^1_4 = 4 + 4 = 8 \quad \langle 2, 1, 4, 3, 5 \rangle \tag{A.6}$$

Tal como no caso $f^1_{n_i=3}$, não é realizado o cálculo $f^2_{n_i=3}$ porque $D(3, n_l) = \infty$ para $n_l = 2, 4$.

$$f^2_{n_i=4} \quad (n_l = 2): \quad D(n_i = 4, n_l = 2) + f^1_2 = \infty + 3 = \infty \tag{A.7}$$

$$(n_l = 3): \quad D(n_i = 4, n_l = 3) + f^1_3 = 3 + \infty = \infty \tag{A.8}$$

$$\tag{A.9}$$

$f^2_{n_i}$				
n_i	$n_l = 2$	$n_l = 3$	$n_l = 4$	$\min(n_l = 2, n_l = 3, n_l = 4)$
2	-	∞	8 (1)	8 (1)
3	∞	-	∞	∞
4	∞	∞	-	∞

Tabela A.4: Tabela com os dados $f^2_{n_i}$ referente à rede da figura A.1.

Finalmente pode calcular-se f^2_1 e obter-se o caminho procurado do nó 1 para o nó 5.

n_i	$f^0_{n_i}$	$f^1_{n_i}$	$f^2_{n_i}$
2	2 (1)	3	8 (1)
3	1	∞	∞
4	4 (3)	4 (3)	∞

Tabela A.5: Tabela com os dados resultantes de A.2, A.3 e A.4.

$$f^2_1 = \min \begin{cases} D(1,2) + f^1_2 = 1 + 3 = 4 & \langle 1,2,3,5 \rangle \\ D(1,3) + f^1_3 = 2 + \infty = \infty & \\ D(1,4) + f^1_4 = 3 + 4 = 7 & \langle 1,4,3,5 \rangle \end{cases} \tag{A.10}$$

O caminho efectivamente encontrado é o caminho óptimo, $\langle 1,2,3,5 \rangle$, de custo mínimo 5. Segundo Dreyfus a aplicação do algoritmo descrito em [1] impede a obtenção do caminho $\langle 1,2,3,5 \rangle$. Este caminho não seria obtido porque, segundo Dreyfus, seria gerado o caminho $\langle 1,2,1,5 \rangle$ com custo 3 o qual é no entanto inadmissível. Como se verificou pela execução do algoritmo este caminho não chega a ser gerado pelo que o efeito colateral previsto por Dreyfus não se verifica neste caso.

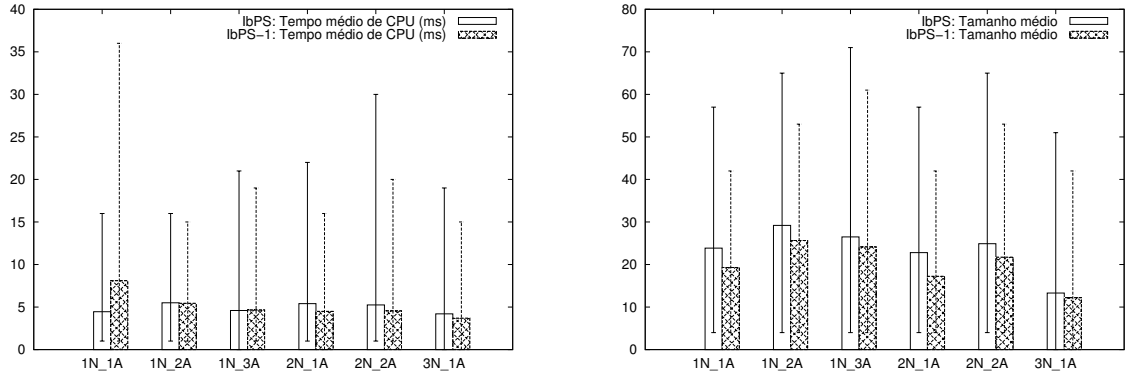
O caminho $\langle 1,2,1,5 \rangle$ só poderia ser gerado se o *nó origem* $n_1 = 1$ *fosse considerado um nó específico*, dando origem a mais uma linha e uma coluna na tabela $f^1_{n_i}$. Nesse caso tomando $n_l = 1$ $(n_l = 1) : D(n_i = 2, n_l = 1) + f^0_1 = 1 + 1 = 2 \quad \langle 2,1,5 \rangle$. Seria então este o caminho que minimizaria $f^2_{n_i=2}$, dando origem ao caminho final $\langle 1,2,1,5 \rangle$ de custo 3, que no entanto seria inadmissível por não conter dois nós obrigatórios.

Contudo, o sub-caminho $\langle 2, 1, 5 \rangle$ deveria desde logo ter sido considerado inadmissível pois Saksena e Kumar indicam que em cada passo deve ser verificado se o número de nós intermédios ξ é verificado: neste caso $\xi = 1$ e existem 0 nós obrigatórios intermédios, uma vez que o nó origem nunca deveria ter considerado como nó específico.

Apêndice B

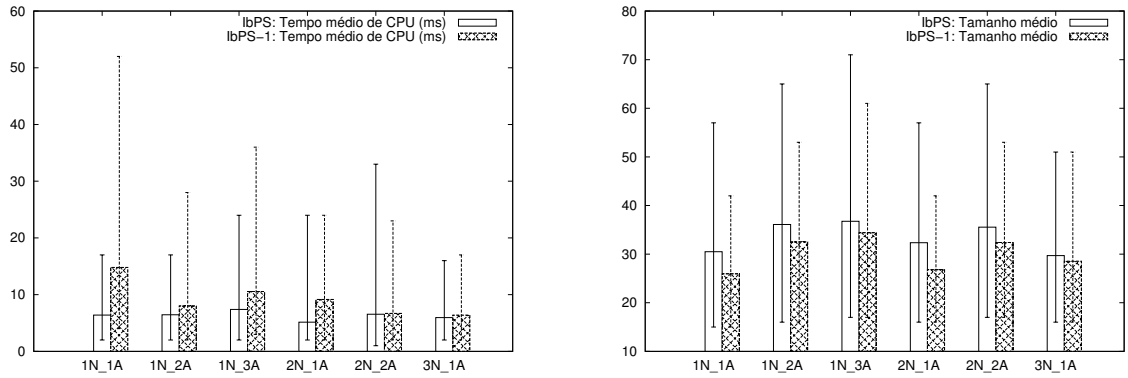
Resultados Obtidos com os Métodos

Apresentam-se neste apêndice os resultados obtidos com todas as redes referidas na tabela 4.1 e que não foram incluídas no capítulo 4.



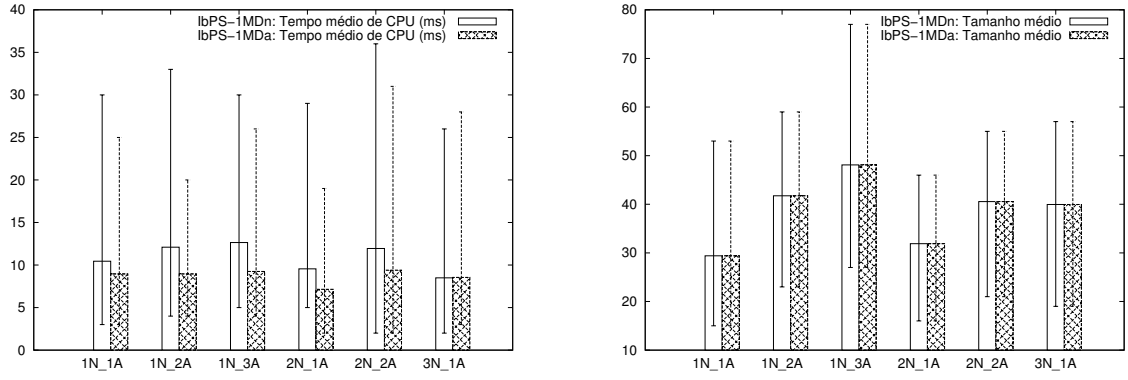
(a) (b)

Figura B.1: Rede Abilene: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



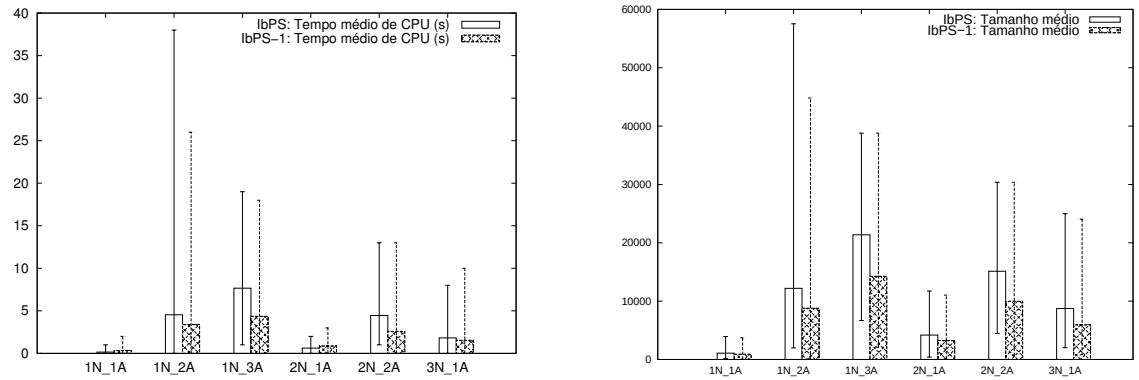
(a) (b)

Figura B.2: Rede Abilene: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



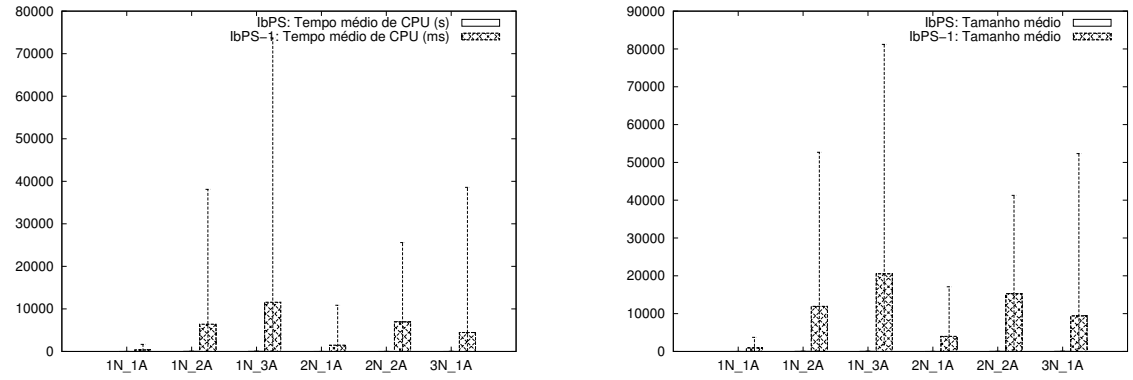
(a) (b)

Figura B.3: Rede Abilene: (a) Tempo médio e (b) Tamanho médio.



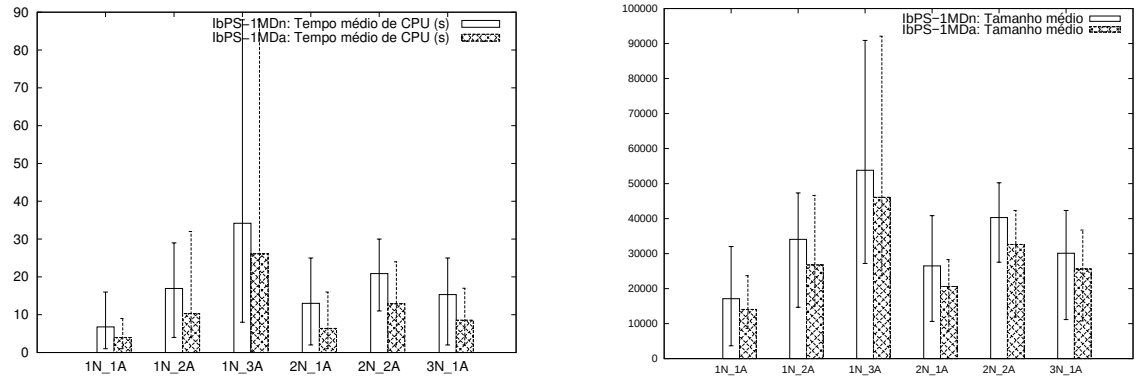
(a) (b)

Figura B.4: Rede Newyork: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



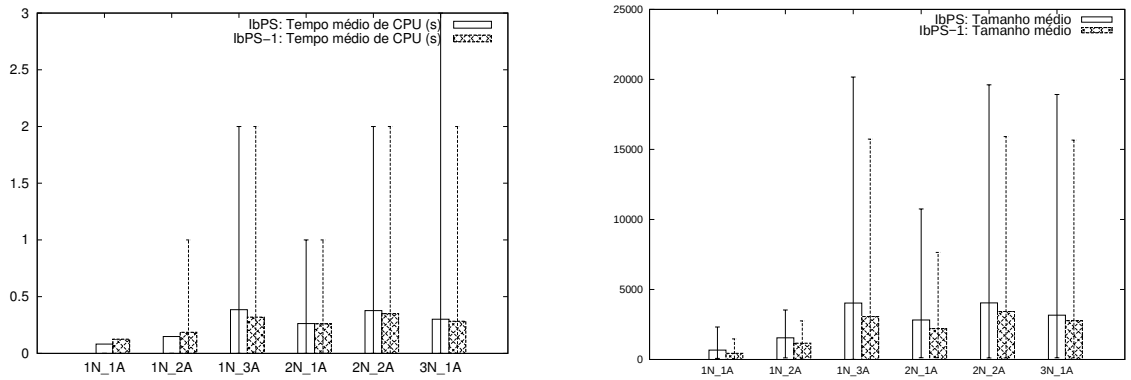
(a) (b)

Figura B.5: Rede Newyork: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



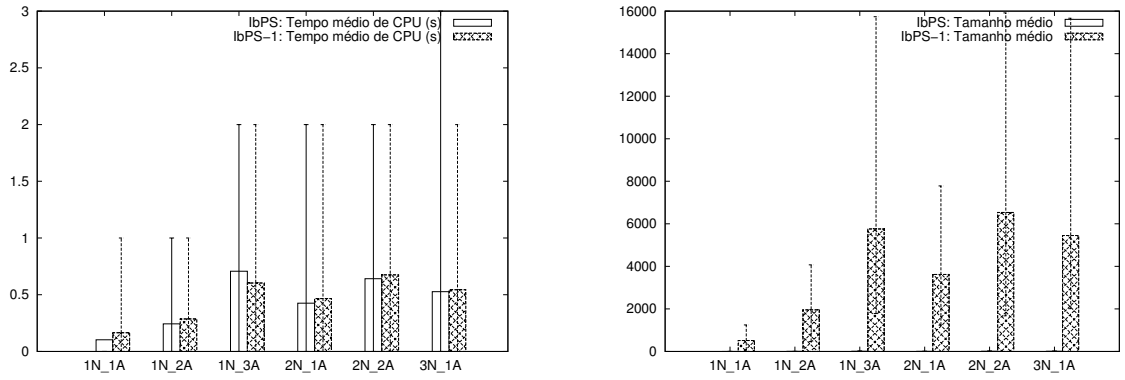
(a) (b)

Figura B.6: Rede Newyork: (a) Tempo médio (b) Tamanho médio.



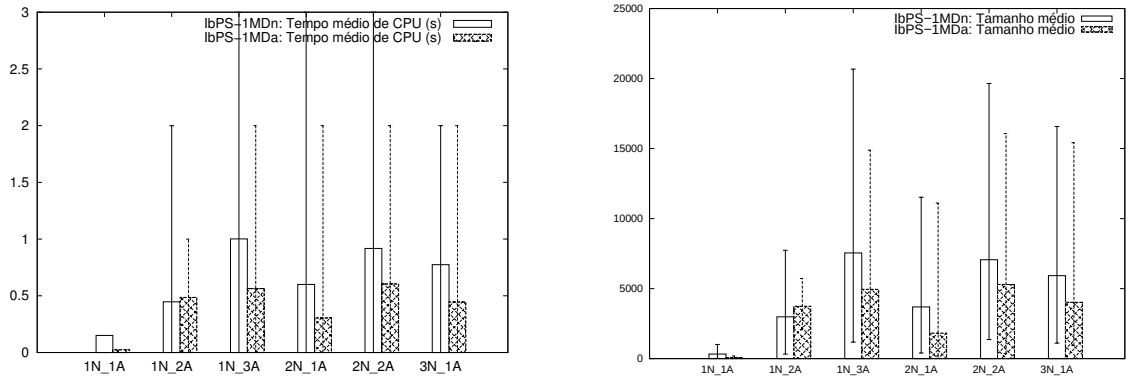
(a) (b)

Figura B.7: Rede Nobel-eu: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



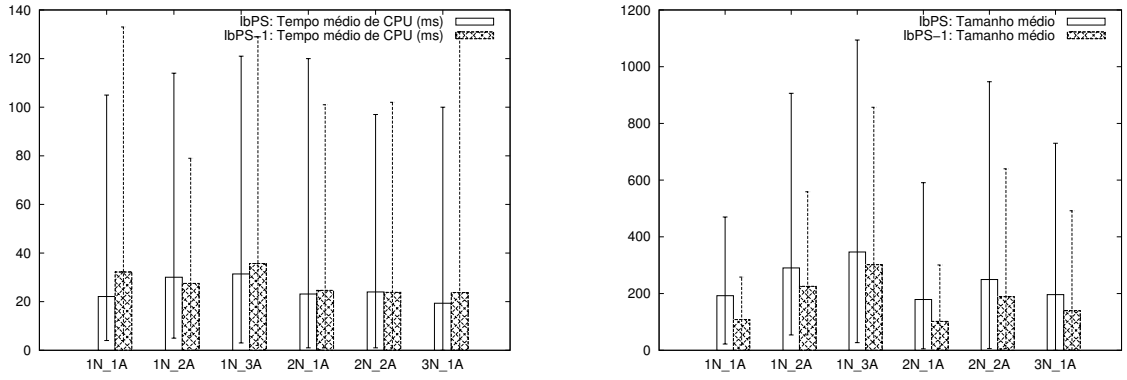
(a) (b)

Figura B.8: Rede Nobel-eu: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



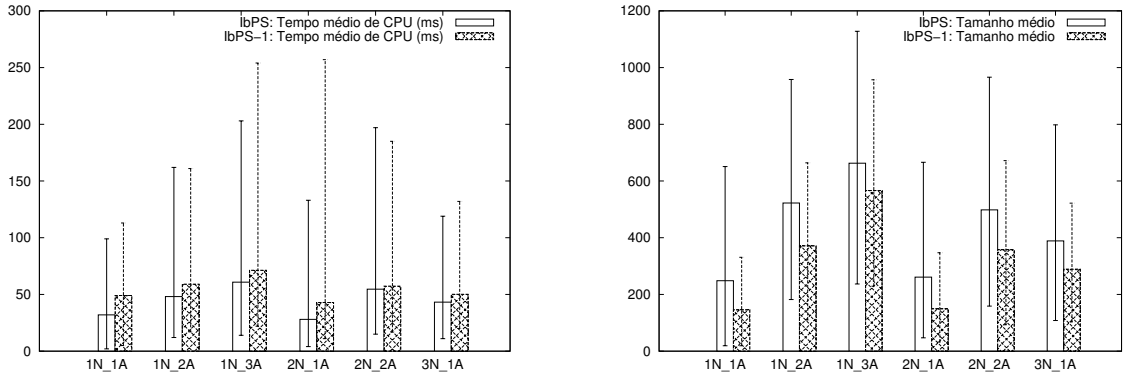
(a) (b)

Figura B.9: Rede Nobel-eu: (a) Tempo médio e (b) Tamanho médio.



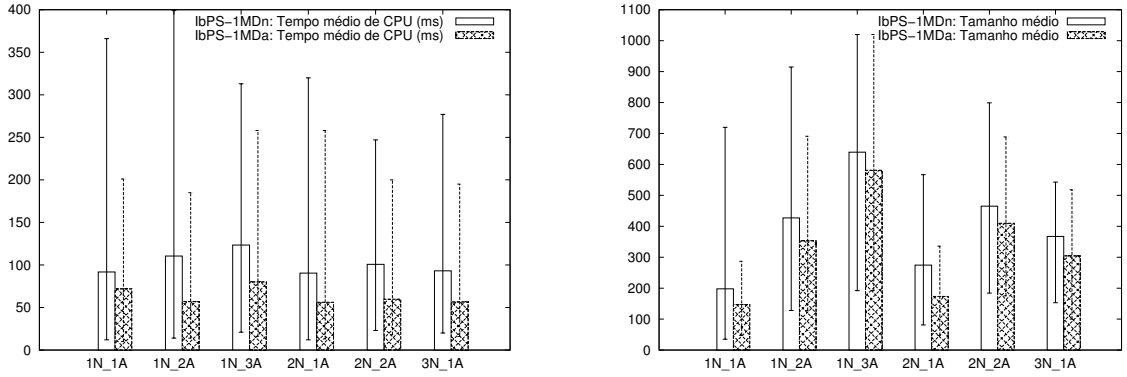
(a) (b)

Figura B.10: Rede Nobel-germany: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



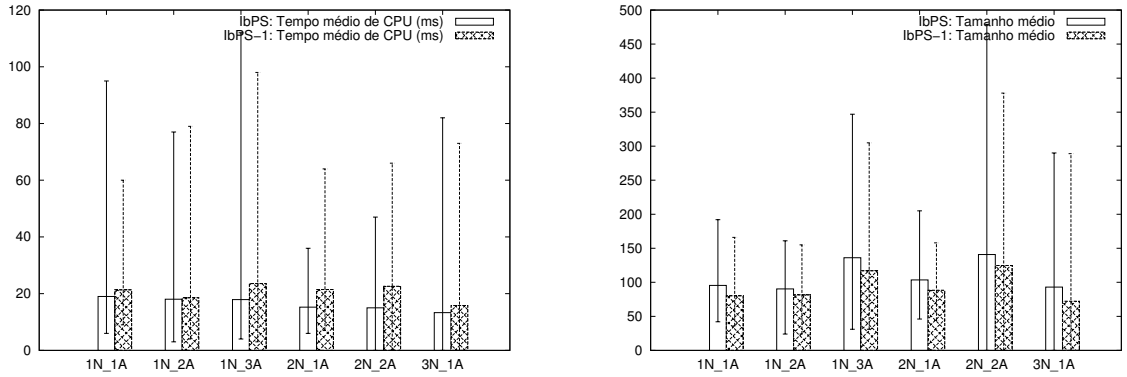
(a) (b)

Figura B.11: Rede Nobel-germany: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



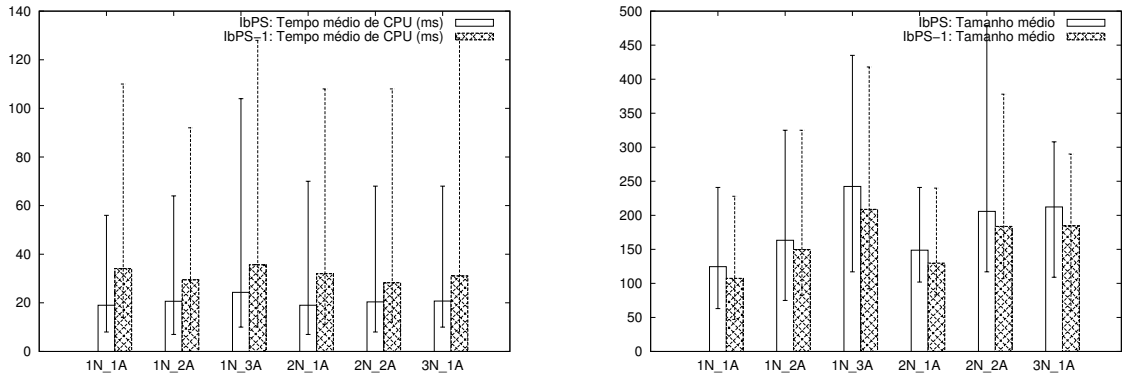
(a) (b)

Figura B.12: Rede Nobel-germany: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



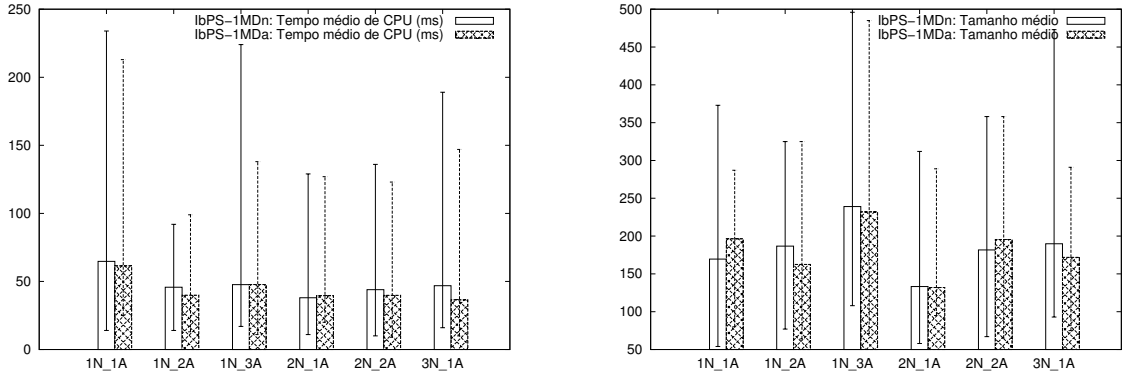
(a) (b)

Figura B.13: Rede Nobel-us: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



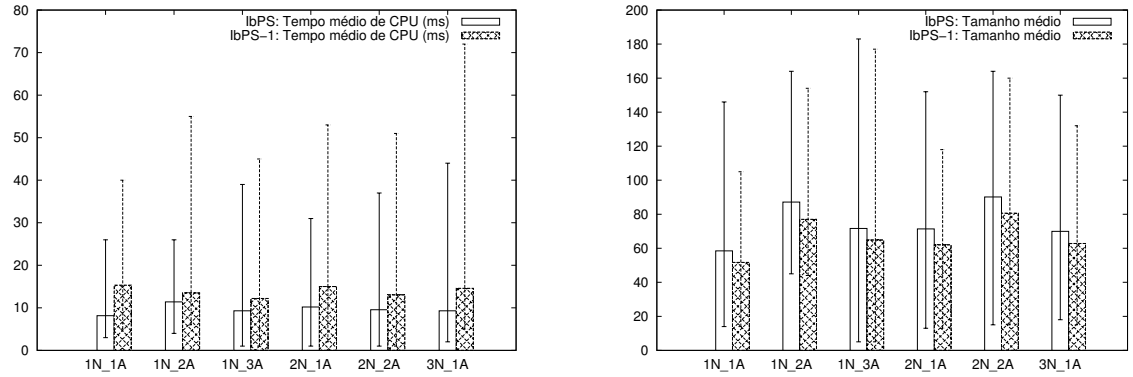
(a) (b)

Figura B.14: Rede Nobel-us: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



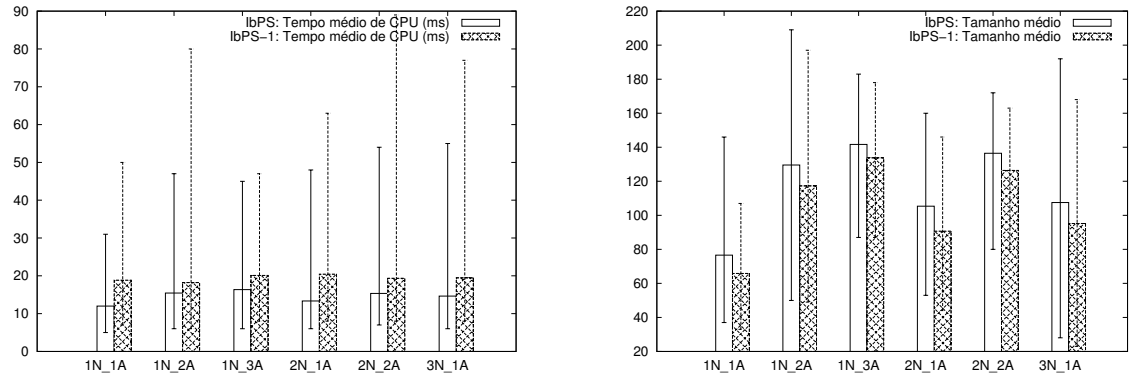
(a) (b)

Figura B.15: Rede Nobel-us: (a) Tempo médio e (b) Tamanho médio.



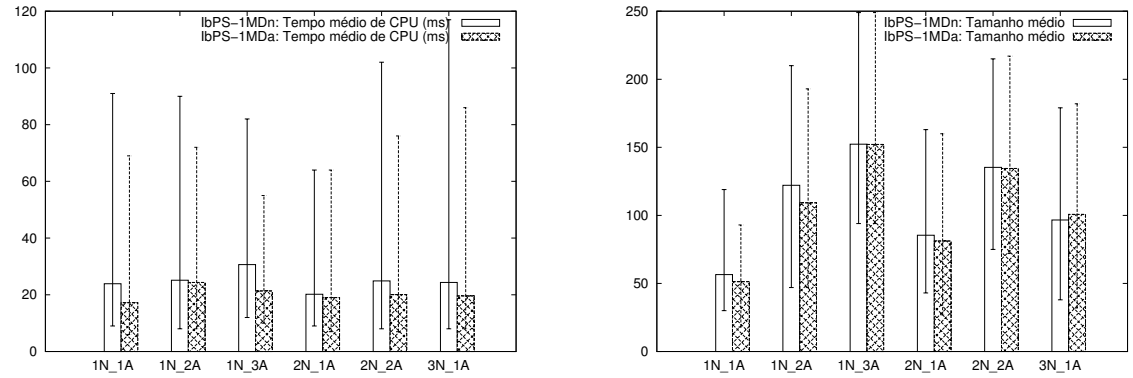
(a) (b)

Figura B.16: Rede Polksa: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



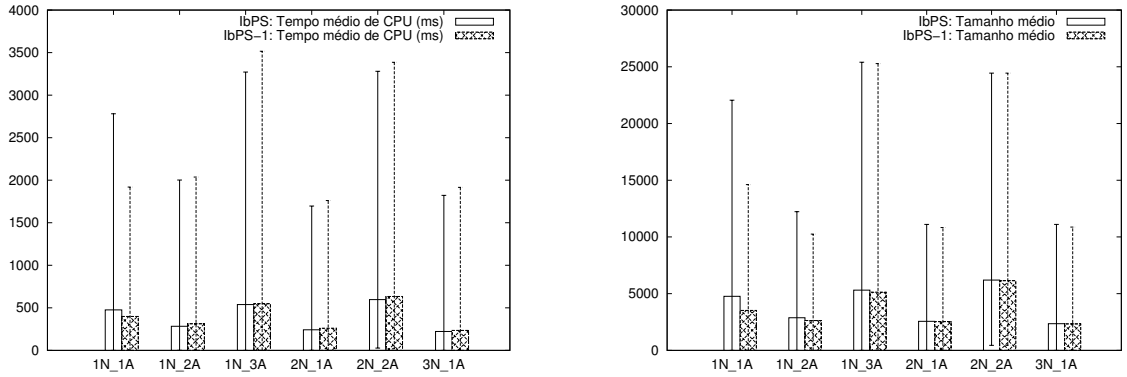
(a) (b)

Figura B.17: Rede Polska: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



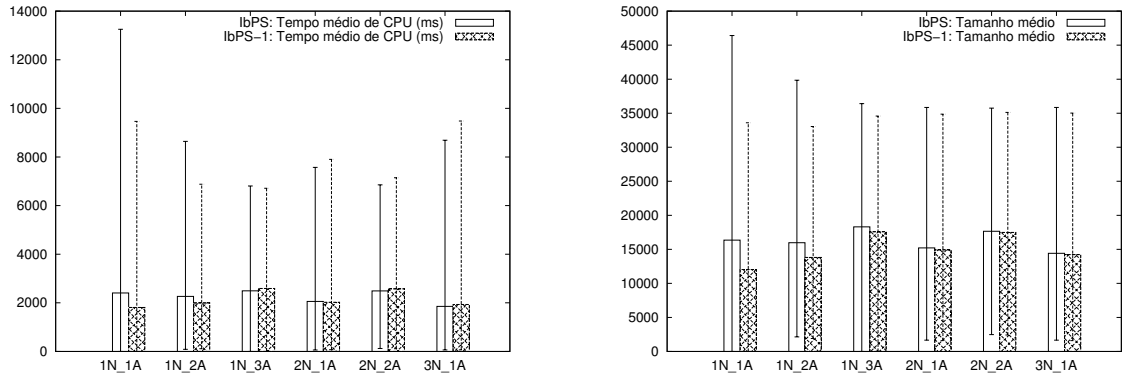
(a) (b)

Figura B.18: Rede Polska: (a) Tempo médio e (b) Tamanho médio.



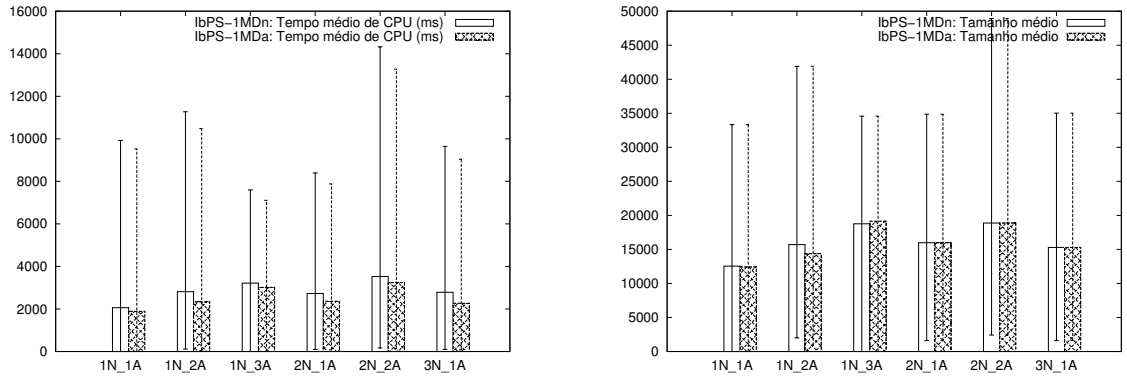
(a) (b)

Figura B.19: Rede France: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



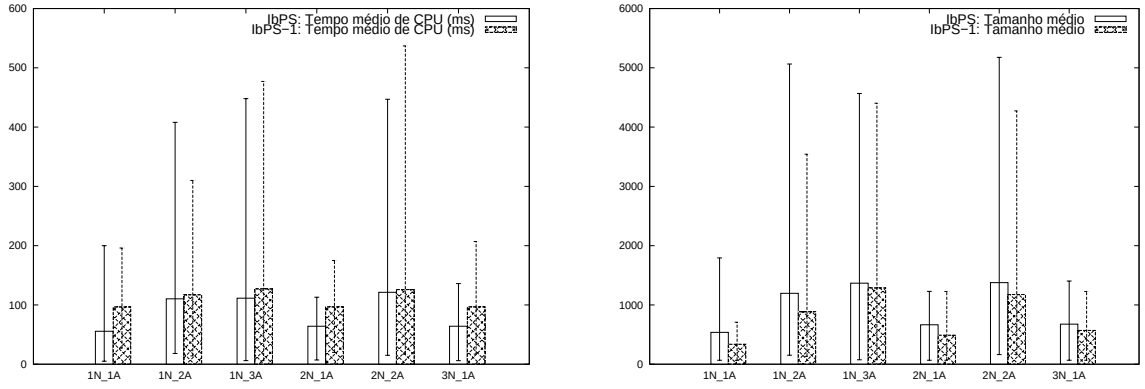
(a) (b)

Figura B.20: Rede France: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



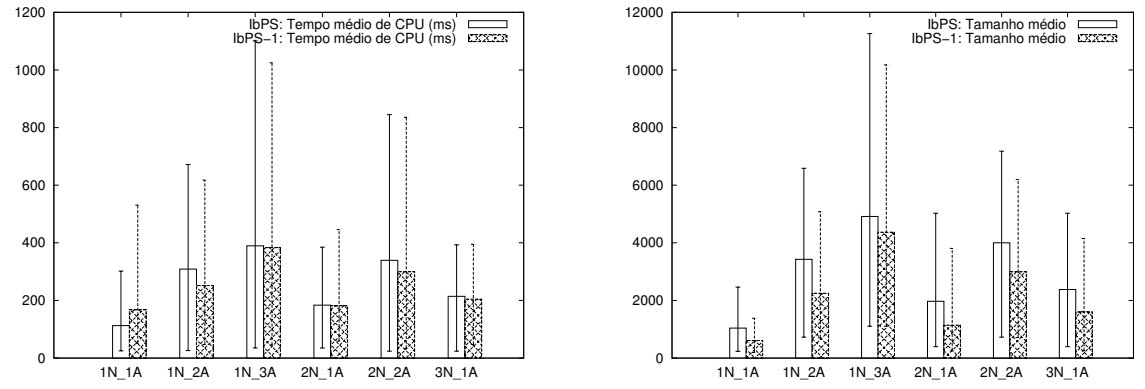
(a) (b)

Figura B.21: Rede France: (a) Tempo médio e (b) Tamanho médio.



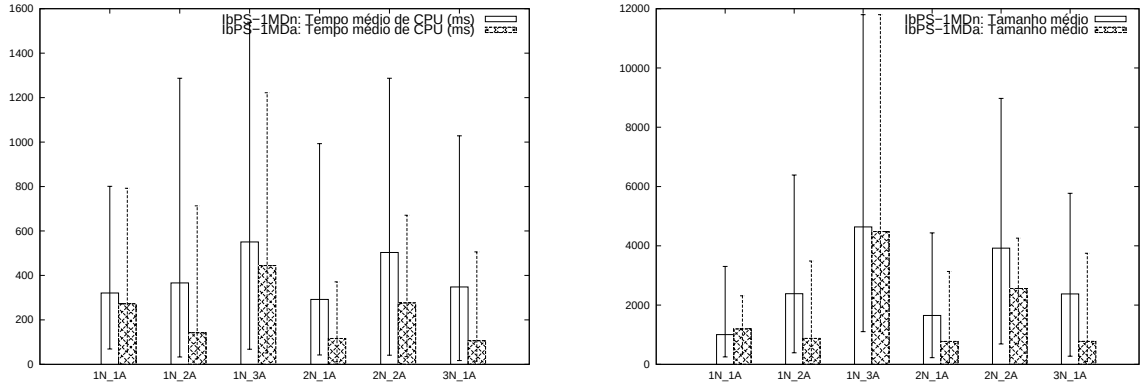
(a) (b)

Figura B.22: Rede Geant: (a) Protecção ao nó - Tempo médio, (b) Protecção ao nó - Tamanho médio.



(a) (b)

Figura B.23: Rede Geant: (a) Protecção ao arco - Tempo médio e (b) Protecção ao arco - Tamanho médio.



(a) (b)

Figura B.24: Rede Geant: (a) Tempo médio e (b) Tamanho médio.

Apêndice C

Formalização dos Problemas PCAE-DN e PCAE-DA

Em [4] são desenvolvidas duas formalizações distintas (Q2 e Q3) para a resolução do PCE. Uma vez que o autor apresenta resultados indicando que a Q3 é significativamente mais eficiente em termos de tempo de CPU que a Q2, utiliza-se então Q3, com as restrições adicionais necessárias para a resolução do PCAE-DN e PCAE-DA, respectivamente.

O modelo matemático para o PCAE-DN:

$$\min \quad \sum_{(n_i,n_j)\in A} f(n_i,n_j)x_{(n_i,n_j)} \tag{C.1}$$

$$\text{sujeito a:} \quad \sum_{(n_i,n_j)\in\delta(i)^+} x_{(n_i,n_j),u} - \sum_{(n_j,n_i)\in\delta(i)^-} x_{(n_j,n_i),u} = \begin{cases} 1 & : \quad n_i = n_1, \\ -1 & : \quad n_i = n_p, \\ 0 & : \quad n_i \in N\backslash\{n_1,n_p\} \end{cases} \tag{C.2}$$
$$\forall n_i \in N,\ u = 1,2$$

$$\sum_{n_i\in N|(n_i,n_j)\in A} x_{(n_i,n_j),1} = 1, \quad \forall n_i \in S, \tag{C.3}$$

$$\sum_{n_i\in N|(n_i,n_j)\in A} x_{(n_i,n_j),1} + x_{(n_j,n_i),1} = 1, \quad \forall (n_i,n_j) \in D, \tag{C.4}$$

$$\tag{C.5}$$

$$\pi_{n_j} - \pi_{n_i} \leq f(n_i, n_j) + M(1 - x_{(n_i,n_j),1}), \quad \forall (n_i, n_j) \in A \tag{C.6}$$

$$\pi_{n_j} - \pi_{n_i} \geq f(n_i, n_j) - M(1 - x_{(n_i,n_j),1}), \quad \forall (n_i, n_j) \in A \tag{C.7}$$

$$\pi_{n_1} = 0 \tag{C.8}$$

$$\pi_{n_i} \geq 0 \tag{C.9}$$

$$\sum_{n_i \in N | (n_i,n_j) \in A} x_{(n_i,n_j),1} + x_{(n_i,n_j),2} \leq 1, \quad n_i \in N \backslash \{n_1\}, \tag{C.10}$$

$x_{(n_i,n_j),u}, \quad \forall (n_i, n_j) \in A, u = 1, 2$ são as variáveis de decisão binárias.

onde M é uma constante positiva suficientemente elevada.

A restrição C.2 garante a continuidade de fluxo do nó n_1 para n_p para os dois caminhos que se deseja obter. Note-se que apenas o custo do caminho activo é minimizado pelo que as variáveis $x_{(n_i,n_j),2}$ definirão um caminho possivelmente com ciclos e cujo custo não foi minimizado.

As restrições C.3 e C.4 garantem que o caminho n_1 a n_p visita os nós pertencentes a S e os arcos pertencentes a T, respectivamente.

As restrições C.6 e C.7 impõem que se um arco (n_i, n_j) está na solução então $\pi_{n_j} - \pi_{n_i} = f(n_i, n_j)$ para este arco, caso contrário estas restrições são redundantes. Assumindo que os custos dos arcos são estritamente positivos então nenhum ciclo pode estar presente numa solução admissível. Cada variável π_{n_i} acumula a distância do nó n_1 até ao nó n_i presente na solução.

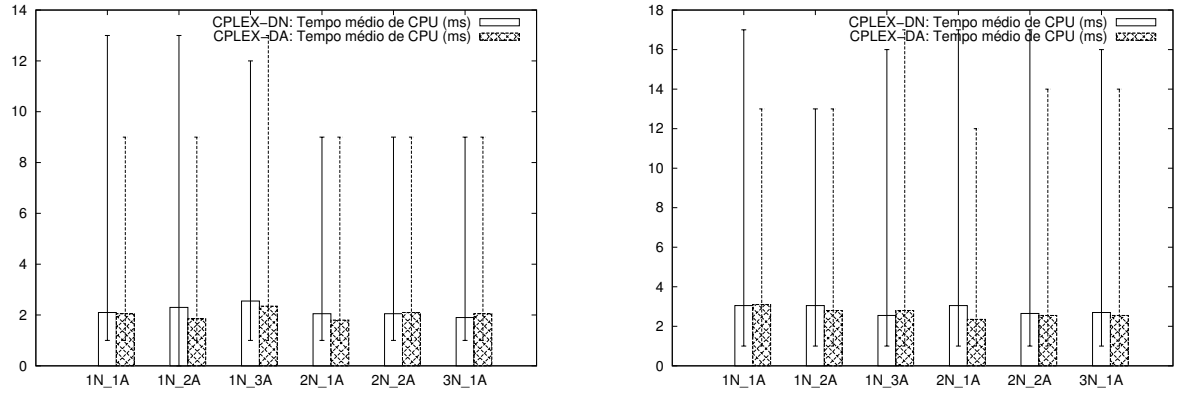
A restrição C.8 indica que a distância de n_1 a si próprio é zero e nos restantes casos, a restrição C.9 indica que as distâncias dos nós a n_1 são estritamente positivas.

Finalmente, a restrição C.10 garante que existem dois caminhos disjuntos nos nós. Se for desejado a obtenção de dois caminhos disjuntos nos arcos, ou seja a resolução do PCAE-DA, esta restrição deve ser substituída pela restrição que se segue:

$$x_{(n_i,n_j),1} + x_{(n_i,n_j),2} + x_{(n_j,n_i),1} + x_{(n_j,n_i),2} \leq 1, \quad (n_i, n_j) \in A \tag{C.11}$$

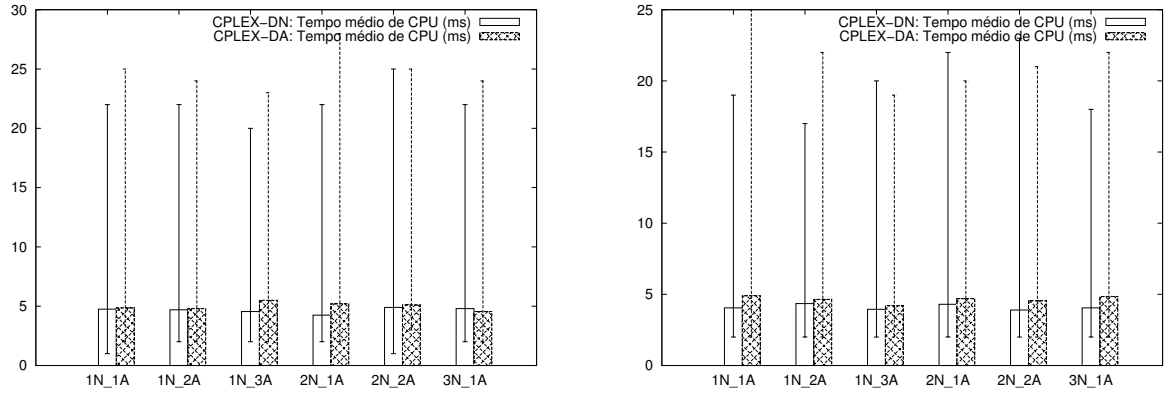
Apêndice D
Resultados Obtidos com o CPLEX

São apresentados aqui os tempos de execução utilizando o CPLEX 12.06 no mesmo computador em que foram obtidos os resultados dos algoritmos desenvolvidos.



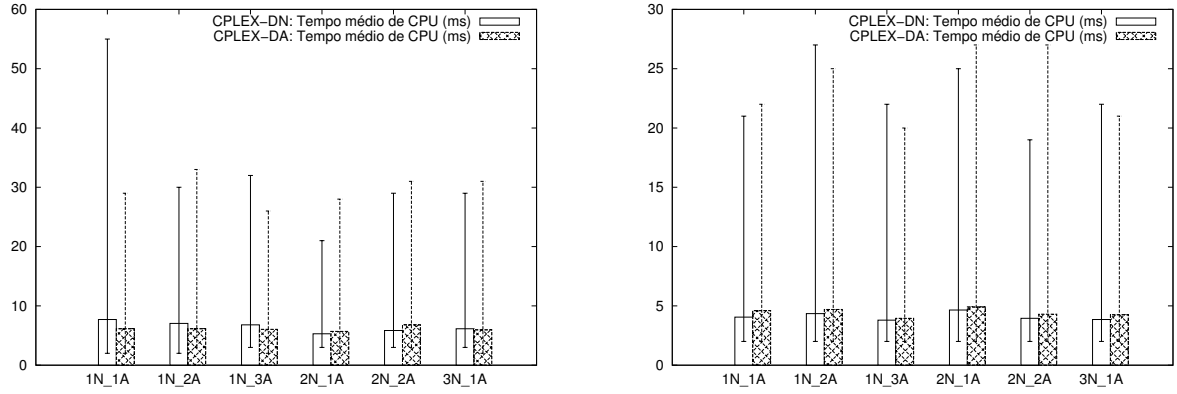
(a) (b)

Figura D.1: Tempo de CPU: (a) Rede Abilene, (b) Rede Atlanta.



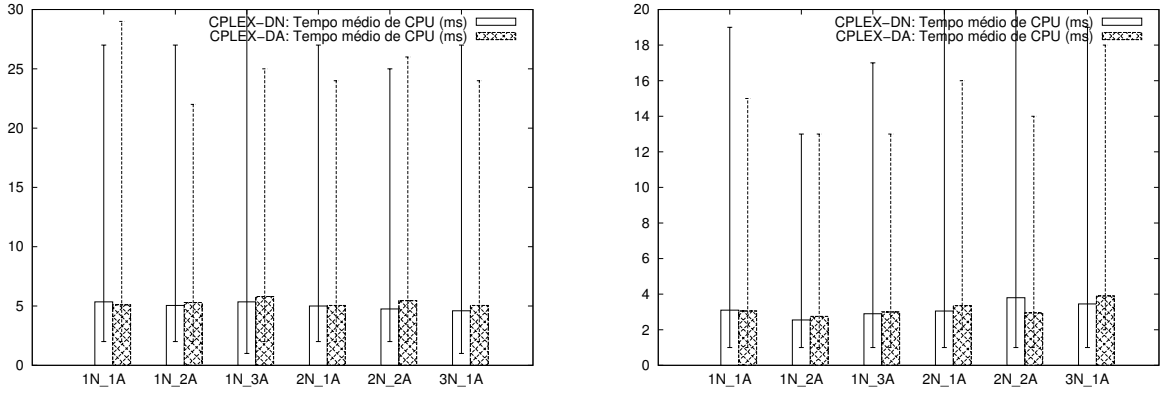
(a) (b)

Figura D.2: Tempo de CPU: (a) Rede France, (b) Rede Geant.



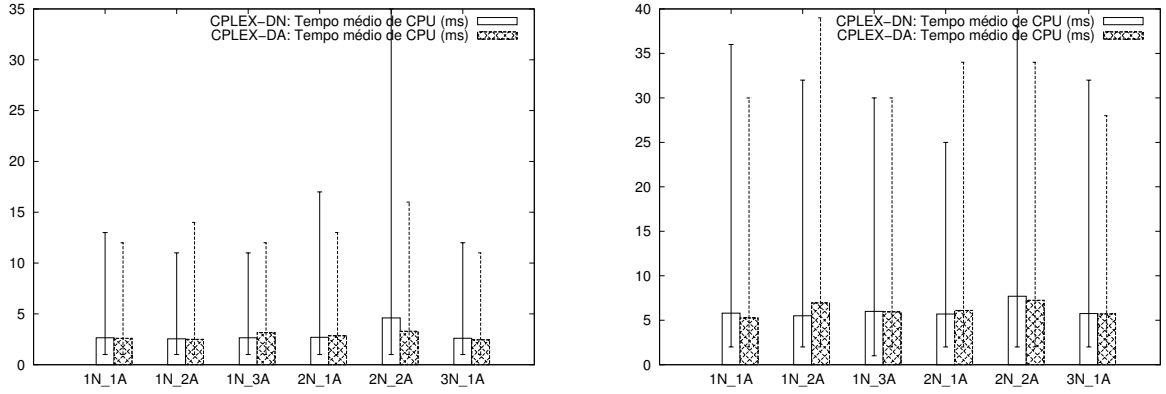
(a) (b)

Figura D.3: Tempo de CPU: (a) Rede Janos-us, (b) Rede Newyork.



(a) (b)

Figura D.4: Tempo de CPU: (a) Rede Nobel-eu, (b) Rede Nobel-germany.



(a) (b)

Figura D.5: Tempo de CPU: (a) Rede Nobel-us, (b) Rede Norway.

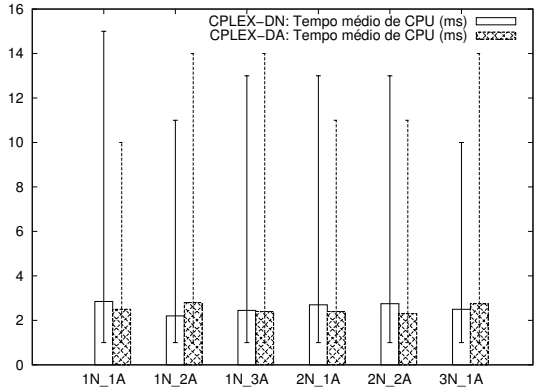


Figura D.6: Tempo de CPU: Rede Polska

Bibliografia

[1] J.P.Saksena and S. Kumar, "The routing problem with 'k' specified nodes," *Operational Research*, vol. 14(5), pp. 909–9013, 1966.

[2] S. E. Dreyfus, "An appraisal of some shortest-path algortihms," *Operational Research*, vol. 17(3), pp. 408–412, 1969.

[3] T. Ibaraki, "Algortihms for obtaining shortest paths visiting specified nodes," *SIAM Review*, vol. 15(2), pp. 309–317, 1973.

[4] R. C. de Andrade, "Shortest-paths visiting a given set of nodes," *Simposio Brasileiro de Pesquisa Operacional*, 2013.

[5] N. Deo and C.-Y. Pang, "Shortest-path algorithms: Taxonomy and annotation." *Networks*, vol. 14, no. 2, pp. 275–323, 1984. [Online]. Available: http://dblp.uni-trier.de/db/journals/networks/networks14.html#DeoP84

[6] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*, 2nd ed. McGraw-Hill Higher Education, 2001.

[7] J. M. S. S. Pereira, *Matemática Discreta - Grafos, Redes, Aplicações.* Editora Luz da Vida, 2009.

[8] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows: theory, algorithms, and applications.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1993.

[9] R. Bellman, "The theory of dynamic programming," *Bull. Amer. Math. Soc.*, vol. 60(6), pp. 503–515, 1954.

[10] M. Bellmore and G. L. Nemhauser, "The traveling salesman problem: A survey," *Operations Research*, vol. 16(3), pp. 538–558, 1968.

[11] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *J. Soc. Indust. Appl. Math.*, vol. 10, pp. 196–210, 1962.

[12] F. A. Kuipers, "An overview of algorithms for network survivability," *CN*, vol. 2012, 2012.

[13] R. Marler and J. Arora, "Survey of multi-objective optimization methods for engineering," *Structural and Multidisciplinary Optimization*, vol. 26, no. 6, pp. 369–395, 2004. [Online]. Available: http://dx.doi.org/10.1007/s00158-003-0368-6

[14] S. Orlowski, R. Wessäly, M. Pióro, and A. Tomaszewski, "Sndlib 1.0 - survivable network design library." *Networks*, vol. 55, no. 3, pp. 276–286, 2010. [Online]. Available: http://dblp.uni-trier.de/db/journals/networks/networks55.html#OrlowskiWPT10

[15] *IBM ILOG CPLEX Optimization Studio V12.6.* IBM, 2013.

[16] T. Koch, A. Martin, and S. Voš, "Steinlib: An updated library on steiner tree problems in graphs," in *Steiner Trees in Industry*, D.-Z. Du and X. Cheng, Eds., 2001, pp. 285 – 325.