3
Implementação Eficiente das Heurísticas para min-RWA

Experimentos computacionais em [88] mostraram que os resultados obtidos por `FFD-RWA` e `BFD-RWA` superam aqueles obtidos por `Greedy-EDP-RWA` [64]. No entanto, os tempos computacionais relatados são altos e o conjunto de instâncias de teste utilizado nos experimentos não permitiu uma comparação precisa entre as heurísticas propostas. Para resolver estas questões, são propostos neste capítulo algoritmos e estruturas de dados que têm como objetivo acelerar o tempo de execução das heurísticas `FF-RWA`, `BF-RWA`, `FFD-RWA` e `BFD-RWA`. Também são propostos novos conjuntos de instâncias de teste, que incluem instâncias aleatórias, instâncias com topologias em grade e instâncias baseadas em redes reais. Por fim, `BFD-RWA` é utilizada na construção de uma heurística de multi-partida para min-RWA.

3.1
Análise da Complexidade Computacional

Dado um grafo direcionado conexo $N = (X, A)$ representando a topologia da rede ótica e o conjunto $\mathcal{T}$ de requisições de caminhos óticos. Sejam $n = |X|$, $m = |A|$ e $l = |\mathcal{T}|$, define-se $c^{sp}(m, n)$ como a complexidade de pior caso de calcular o caminho mais curto de um vértice para todos os outros vértices em N e $c^{del}(m, n)$ como a complexidade de pior caso de remover um arco em N. Por questão de simplicidade, assume-se que $n \leq m$ e $n \leq l$, o que é verdade para todas as instâncias na literatura.

A complexidade de pior caso é igual para as heurísticas `FF-RWA` e `BF-RWA` (Figura 2.6), conforme calculada a seguir. Primeiramente, os conjuntos Ω e S são inicializados em tempo constante na linha 1. Em seguida, as linhas 2 a 12 são repetidas para cada caminho ótico em $\mathcal{T}$. Na linha 3, verifica-se a existência de uma cópia de N em Ω onde o caminho ótico $t_{\pi(i)}$ pode ser roteado com menos de d arcos em tempo $O(l \cdot c^{sp}(m, n))$, calculando-se o caminho mais curto entre os nós terminais de $t_{\pi(i)}$ em todas as cópias de N em Ω. Caso tal caminho não exista, uma nova cópia de N é criada em $O(m)$ na linha 5 e inserida no conjunto Ω em tempo constante na linha 6. As heurísticas `FF-RWA` e `BF-RWA` diferenciam-se apenas na linha 8. `FF-RWA` seleciona qualquer cópia

de N em Ω no qual $t_{\pi(i)}$ pode ser roteado com menos de d arcos, enquanto `BF-RWA` seleciona aquela no qual $t_{\pi(i)}$ pode ser roteado com o menor número de arcos. Em ambos os casos, a complexidade de pior caso é $O(l \cdot c^{sp}(m,n))$. O caminho mais curto entre os nós terminais de $t_{\pi(i)}$ na cópia de N selecionada $p_{\pi(i)}$ é calculado em tempo $O(c^{sp}(m,n))$ na linha 9. Finalmente, o conjunto S é atualizado na linha 10 em tempo constante, e todos os arcos na rota $p_{\pi(i)}$ são removidos da cópia de N selecionada em tempo $O(d \cdot c^{del})$ na linha 11. Sendo assim, a complexidade das heurísticas é:

$$\begin{aligned} T(n,m,l) &= O(1) + l \cdot (O(l \cdot c^{sp}(m,n)) + O(m) + O(1) + O(l \cdot c^{sp}(m,n)) \\ &\quad + O(c^{sp}(m,n)) + O(1) + O(d \cdot c^{del}(m,n))) \\ &= O(l^2 \cdot c^{sp}(m,n) + l \cdot d \cdot c^{del}(m,n)). \end{aligned}$$

Uma implementação de `FF-RWA` e `BF-RWA` usando o algoritmo de Dijkstra [23] para calcular os caminhos mais curtos e uma lista encadeada [23] para representar a lista de adjacência do grafo resultaria em $c^{sp}(m,n) = O(n \cdot \log n + m)$ e $c^{del}(m,n) = O(n)$. Conseqüentemente, a complexidade das heurísticas seria:

$$T(n,m,l) = O(l^2 \cdot (n \cdot \log n + m) + l \cdot d \cdot n).$$

No entanto, como os arcos de N têm pesos unitários, o caminho mais curto pode ser calculado usando um algoritmo de busca em amplitude [23] em tempo $O(m)$. Além disso, um arco pode ser removido de uma cópia de N em tempo constante usando a estrutura de dados descrita a seguir. Para cada nó $i \in X$, armazena-se uma lista duplamente encadeada L_i cujos elementos correspondem aos arcos (i,j) originados no nó i. Além disso, armazena-se uma matriz de ponteiros F_{ij}, para $i,j \in X$, onde $F_{i,j}$ aponta para o endereço de memória de cada arco (i,j) em L_i, caso $(i,j) \in L_i$; caso contrário, $F_{ij} = null$. Uma vez que um arco (i,j) é removido, usa-se F_{ij} para obter o endereço de memória do nó correspondente a (i,j) em L_i. Como L_i é duplamente encadeada, o nó correspondente a (i,j) pode ser removido de L_i em tempo constante. Uma implementação de `FF-RWA` e `BF-RWA` usando esta estrutura de dados e o algoritmo de caminhos mais curtos baseados em busca em amplitude resulta em $c^{sp}(m,n) = O(m)$ e $c^{del}(m,n) = O(1)$. Conseqüentemente, a complexidade das heurísticas é dada por:

$$T(n,m,l) = O(l^2 \cdot m + l \cdot d \cdot 1) = O(l^2 \cdot m).$$

As complexidades de pior caso das heurísticas `FFD-RWA` e `BFD-RWA` são

iguais as de `FF-RWA` e `BF-RWA`, exceto pela ordenação prévia dos caminhos óticos em ordem decrescente do tamanho dos seus caminhos mais curtos. Os caminhos mais curtos de todos os caminhos óticos podem ser calculados em $O(l \cdot m)$ utilizando o algoritmo de busca em amplitude, e a ordenação dos caminhos óticos pode ser realizada em $O(l \cdot \log l)$ utilizando o algoritmo Heapsort [23]. Conseqüentemente, a complexidade das heurísticas `FFD-RWA` e `BFD-RWA` utilizando os algoritmos e estruturas de dados descritos no parágrafo anterior é dada por:

$$T(n, m, l) = O(l \cdot m) + O(l \cdot \log l) + O(l^2 \cdot m + l \cdot d \cdot 1) = O(l^2 \cdot m).$$

A implementação padrão (STD) das heurísticas `FF-RWA`, `BF-RWA`, `FFD-RWA` e `BFD-RWA` utiliza os algoritmos e estruturas de dados apresentadas nos dois parágrafos anteriores.

3.2 Implementações Avançadas

A operação mais custosa das heurísticas aparece na linha 3 da figura 2.6, onde o caminho mais curto é calculado para l caminhos óticos em no máximo l cópias de N. Neste ponto, somente o tamanho do caminho mais curto é necessário. Portanto, propõe-se uma implementação alternativa baseada em uma matriz de distâncias $n \times n$, onde cada entrada é o valor do caminho mais curto entre dois pares de vértices em X. Esta matriz é inicializada em tempo $O(n \cdot m)$ na linha 1 e instanciada para cada nova cópia de N na linha 5 em tempo $O(n^2)$. Uma vez que um arco é removido de uma cópia de N, os caminhos mais curtos naquela cópia da N podem mudar e a matriz de distâncias deve ser atualizada. Esta nova estrutura de dados permite que consultas ao valor de um caminho mais curto numa cópia de N sejam realizadas em tempo constante. Entretanto, a eficiência das heurísticas depende de o quão rápido essa estrutura de dados é atualizada após a remoção de um arco.

Nesta tese, são propostos quatro algoritmos para atualizar a matriz de distâncias de uma cópia de N após a remoção de um arco. Os dois primeiros, RRg e RRt, são baseados no trabalho de Ramalingam e Reps [80] sobre algoritmos dinâmicos para calcular caminhos mais curtos em grafos. Os dois últimos, NRRg e NRRg, são adaptações dos dois primeiros.

Os algoritmos de Ramalingam e Reps [80] são brevemente descritos a seguir. Um estudo aprofundado sobre estes e outros algoritmos dinâmicos para caminhos mais curtos pode ser encontrado em [13, 16]. Dado o grafo $N = (X, A)$ e um nó $v \in X$, o grafo de caminhos mais curtos de v é um grafo

$G^v = (X, A^v)$, onde $A^v \subseteq A$, tal que qualquer caminho de um vértice $u \in X$ até o vértice $v \in X$ em G^v corresponde ao caminho mais curto de u para v em N. Vale salientar que todos os possíveis caminhos mais curtos de u até v estão contidos em G^v. Já a árvore de caminhos mais curtos de v é um grafo acíclico $T^v = (X, A^v)$, onde $A^v \subseteq A$, tal que o caminho de qualquer vértice $u \in X$ até v em T^v corresponde ao caminho mais curto de u para v em N. Diferentemente do grafo G^v, a árvore T^v armazena apenas um dos possíveis caminhos mais curtos de u para v.

O algoritmo de Ramalingam e Reps [80] para atualização dinâmica de grafos de caminhos mais curtos após a remoção de um arco é descrito a seguir. Após a remoção do arco $(i, j) \in A$, o algoritmo verifica se (i, j) pertence a A^v e se for este o caso remove-o de G^v. Se $(i, j) \notin A^v$ ou se G^v permanece conexo após a remoção de (i, j), o tamanho do caminho mais curto de qualquer vértice em X para v não se altera e o algoritmo termina. Caso contrário, o algoritmo identifica o conjunto U de vértices cujos caminhos mais curtos até v são alterados após a remoção de (i, j). Em seguida, os vértices $z \in U$ são retirados de G^v e inseridos numa fila de prioridades [23], com prioridade igual ao peso do arco de menor custo de z para algum dos vértices que continuam em G^v. Deste ponto em diante, o algoritmo procede exatamente como o algoritmo de Dijkstra.

O algoritmo de Ramalingam e Reps [80] para atualização dinâmica de árvores de caminhos mais curtos após a remoção de um arco é similar ao algoritmo descrito acima. Entretanto, após a remoção de um arco, a identificação dos vértices em U e a atualização dos caminhos mais curtos são executadas de forma mais eficiente em árvores do que em grafos de caminhos mais curtos. Por outro lado, todas as vezes que o arco (i, j) é removido de A^v, a árvore de caminhos mais curtos tem que ser necessariamente atualizada.

A complexidade no pior caso dos dois algoritmos é a mesma do algoritmo de Dijkstra, $O(n \cdot \log n + m)$, usando-se uma *heap de Fibonacci* [23] para implementar a fila de prioridades. Entretanto, como os arcos de N têm pesos unitários, eles podem ser implementados em tempo $O(m)$ usando *buckets* [23] para implementar a fila de prioridades.

O primeiro algoritmo proposto para atualização da matriz de distâncias RRg, armazena n grafos de caminhos mais curtos, cada um deles contendo os caminhos mais curtos de um vértice em $v \in X$ para todos os outros vértices em X. O grafo de caminhos mais curtos de um vértice $v \in X$ pode ser calculado em $O(m)$ da seguinte forma. Primeiramente, constrói-se uma árvore de caminhos mais curtos enraizada em v em tempo $O(m)$, utilizando-se um algoritmo de busca em amplitude. Em seguida, são acrescentados a esta árvore todos os

arcos de A cujo nó de origem está num nível imediatamente acima do nó de destino na árvore de caminhos mais curtos. Os n grafos de caminhos mais curtos são inicializadas em $O(n \cdot m)$ na linha 1 da Figura 2.6 e instanciados para cada nova cópia de X em tempo $O(n \cdot m)$ na linha 5. Após cada arco em $p_{\pi(i)}$ ser removido da cópia de N correspondente a $w_{\pi(i)}$ na linha 11, RRg verifica se algum grafo de caminhos mais curtos nesta cópia de N precisa ser atualizado. Se for este o caso, o algoritmo de Ramalingam e Reps [80] para grafos de caminhos mais curtos é utilizado e a matriz de distâncias é atualizada. No pior caso, os n grafos de caminhos mais curtos são atualizados em $O(n \cdot m)$ após a remoção de um arco. Este algoritmo resulta em $c^{sp}(m,n) = \overline{O}(1)$ e $c^{del}(m,n) = O(n \cdot m)$. Portanto, a complexidade das quatro heurísticas para min-RWA usando RRg é $T(n,m,l) = O(l^2 \cdot 1 + l \cdot d \cdot n \cdot m) = O(l^2 + l \cdot d \cdot n \cdot m)$.

O algoritmo RRt armazena n árvores de caminhos mais curtos, cada uma delas contendo um caminho mais curto de um vértice em $v \in X$ para todos os outros vértices em X. As n árvores de caminhos mais curtos são inicializados em $O(n \cdot m)$ na linha 1 da Figura 2.6 e instanciadas para cada nova cópia de N em tempo $O(n^2)$ na linha 5. Assim como RRg, após cada arco em $p_{\pi(i)}$ ser removido da copia de N correspondente a $w_{\pi(i)}$ na linha 11, RRt verifica se alguma árvore de caminhos mais curtos nesta cópia de N deve ser atualizada. Se for este o caso, o algoritmo de Ramalingam e Reps [80] para árvores de caminhos mais curtos é utilizado e a matriz de distâncias é atualizada. No pior caso, as n árvores de caminhos mais curtos são atualizadas em $O(n \cdot m)$ após a remoção de um arco. Este algoritmo resulta em $c^{sp}(m,n) = \overline{O}(1)$ e $c^{del}(m,n) = O(n \cdot m)$. Portanto, a complexidade das quatro heurísticas para min-RWA usando RRt é $T(n,m,l) = O(l^2 + l \cdot d \cdot n \cdot m)$.

O algoritmo RRg (resp. RRt) pode não ser eficiente na prática, devido ao fato de que até n grafos (resp. árvores) de caminhos mais curtos podem ser atualizados após a remoção de um arco. Para remediar este fato, propõem-se duas novas estratégias de implementação. O algoritmo NRRg (resp. NRRt) usa as mesmas estruturas de dados de RRg (resp. RRt), mas não as atualiza assim que um arco é removido. Como o caminho mais curto entre um par de vértices só pode crescer após a remoção de um arco, a matriz de distâncias armazena para cada par de vértices um limite inferior para o tamanho do caminho mais curto entre eles. Se o limite inferior é maior que d, a distância correta não é necessária na linha 3 da Figura 2.6. Caso contrário, a distância correta pode ser calculada recuperando-se o caminho mais curto do caminho ótico no grafo (resp. árvore) de caminhos mais curtos em tempo $O(d)$. Se nenhum arco ao longo do caminho mais curto tiver sido removido, o valor armazenado na matriz de distâncias está correto. Caso contrário, o algoritmo NRRg

(resp. NRRt) reconstrói o grafo (resp. árvore) de caminhos mais curtos em tempo $O(m)$ e atualiza os respectivos valores na matriz de distâncias. Portanto, a complexidade de pior caso de se calcular um caminho mais curto é $O(m)$. Como os grafos (resp. árvores) de caminhos mais curtos são atualizados na linha 3, a remoção de um arco é realizada em tempo constante na linha 11. Este algoritmo resulta em $c^{sp}(m,n) = O(m)$ e $c^{del}(m,n) = O(1)$. Portanto, a complexidade das quatro heurísticas para min-RWA usando NRRg (resp. NRRt) é $T(n,m,l) = O(l^2 \cdot m + l \cdot d \cdot 1) = O(l^2 \cdot m)$.

3.3 Heurística de Multi-partida para min-RWA

Como pode-se observar a seguir, a heurística `BFD-RWA` obteve os melhores resultados nos experimentos computacionais. A qualidade das soluções geradas por `BFD-RWA` é sensível à qualidade da ordenação π de caminhos óticos que é passada como parâmetro para a heurística. Como os arcos de N têm pesos unitários, existem muitos caminhos óticos com o mesmo valor de caminho mais curto. A forma como estes empates são resolvidos ao gerar-se a ordenação π influencia na qualidade das soluções retornadas pela heurística. Sendo assim, é natural embutir `BFD-RWA` em um procedimento multi-partida que resolve esses empates aleatoriamente. A heurística `BFD-RWA` é executada várias vezes partindo de diferentes permutações de caminhos óticos e, ao final, retorna a melhor solução encontrada em todas as iterações. Este procedimento, chamado de `MS-BFD`, encerra sua execução quando um número máximo de iterações é atingido ou quando uma solução com um custo tão bom quando um dado alvo é encontrada.

3.4 Experimentos Computacionais

Denota-se por `FF-RWA`^{STD}, `FF-RWA`^{RRg}, `FF-RWA`^{RRt}, `FF-RWA`^{NRRg} e `FF-RWA`^{NRRt} as implementações da heurística `FF-RWA` usando STD, RRg, RRt, NRRg e NRRt, respectivamente. A mesma notação também é empregada para as diferentes implementações das heurísticas `FFD-RWA`, `BF-RWA` e `BFD-RWA`. As heurísticas foram codificas em C++ e compiladas com GNU GCC versão 4.0.3. As opções de otimização de código não foram utilizadas, já que a otimização pode não ser proporcional para todos os algoritmos estudados. Os dois primeiros experimentos descritos abaixo foram executados em um Pentium IV 2.8 GHz, enquanto o terceiro foi executado em um Pentium IV 3.4 GHz. Os tempos de processamento são relatados em segundos. A qualidade das heurísticas é apresentada como o *desvio relativo* Δ=(UB-LB)/LB entre o

custo UB da solução fornecida pela heurística e o limite inferior LB para o custo da solução ótima, dado pelo valor ótimo da relaxação linear da formulação (2-1) a (2-4), apresentada no capítulo 2.

Quatro conjuntos de instâncias de teste foram utilizados nos experimentos computacionais. O conjunto K foi gerado aleatoriamente como proposto em [88]. O conjunto W é uma coleção das instâncias realistas mais estudadas na literatura. Já os conjuntos Y e Z são propostos nesta tese. Os grafos que representam a topologia física das redes óticas são conexos e cada enlace ótico é representado por um par de arcos bidirecionais. As matrizes de tráfego são assimétricas, ou seja, pode existir uma requisição de caminho ótico de i para j e não existir uma de j para i. A descrição de cada conjunto de instâncias é apresentada a seguir.

O conjunto de instâncias K foi gerado aleatoriamente, exatamente como sugerido em [88]. Os grafos tem 100 nós e a probabilidade γ de existir um enlace entre um par de nós assume valores iguais a 0,03, 0,04 e 0,05. A probabilidade ρ de existir uma requisição de caminho ótico entre dois nós assume valores iguais a 0,2, 0,4, 0,6, 0,8 e 1,0. Quinze grupos de cinco instâncias foram criados, combinando cada possível par de valores de γ e ρ.

Durante os experimentos computacionais, observou-se que a maioria das instâncias no conjunto K era de fácil resolução e que este fato se devia a suas características estruturais. Primeiramente, existem nós conectados a um único enlace e, conseqüentemente, todos os caminhos óticos incidentes nestes nós atravessam este arco e devem ter comprimentos de onda diferentes. Além disso, existem arcos que fazem parte de todas as rotas de vários caminhos óticos e, conseqüentemente, estes caminhos óticos têm que ter comprimentos de onda diferentes. Estas características implicam em um limite inferior alto para o número de comprimentos de onda necessários para estabelecer os caminhos óticos. Para a maioria das instâncias em K, uma solução com este número de comprimentos de onda pode ser facilmente encontrada.

O conjunto Y de instâncias é proposto nesta tese. Os grafos das instâncias neste conjunto são gerados aleatoriamente com o mesmo número de nós e os mesmos valores de γ e ρ usados para gerar as instâncias do conjunto K. Entretanto, quando γ é igual a 0,04 e 0,05 considera-se somente os grafos cujos nós têm pelo menos dois vizinhos. Além disso, restringe-se o diâmetro dos grafos a 5, 6 e 7 nas instâncias com γ igual a 0,03, 0,04 e 0,05, respectivamente. As matrizes de tráfego são as mesmas usadas para o conjunto K de instâncias. Como anteriormente, quinze grupos de cinco instâncias foram gerados aleatoriamente, combinando cada possível par de valores para γ e ρ.

As instâncias do conjunto Z são construídas na forma de uma grade

envolvida em um torus $n \times m$. Cada nó está conectado com seus quatro vizinhos mais próximos. A figura 3.1 mostra um exemplo de uma grade 3×4. Cinco grafos em grade foram gerados com aproximadamente 100 nós (10×10, 8×13, 6×17, 5×20, 4×25). Para cada um deles, cinco matrizes de tráfego foram geradas aleatoriamente com a probabilidade ρ de existir uma requisição de caminho ótico entre dois pares de nós assumindo valores iguais a 0,2, 0,4, 0,6, 0,8 e 1,0.

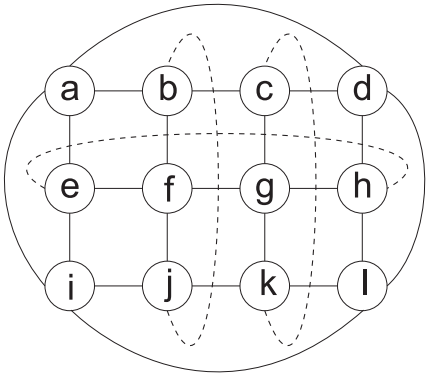

Figura 3.1: Exemplo de uma topologia em grade 3×4.

O conjunto W é uma coleção das instâncias realistas mais estudadas na literatura, além de duas novas instâncias ATT e ATT2, introduzidas nesta tese. As topologias e matrizes de tráfego destas instâncias se assemelham a duas redes reais de telecomunicações. A topologia da rede Finland foi obtida em [48] e sua matriz de tráfego é a mesma utilizada em [70]. As redes EON, NSF e NSF2, assim como suas respectivas matrizes de tráfego, foram extraídas de [51]. As três primeiras colunas da Tabela 3.5 apresentam o nome, o número de nós e o número de enlaces em cada instância do conjunto W. O número total de caminhos óticos e o número máximo de caminhos óticos que partem de um mesmo nó são mostrados na quarta e na quinta coluna, respectivamente.

O primeiro experimento avalia e compara o desempenho de $\texttt{FF-RWA}^{STD}$, $\texttt{FFD-RWA}^{STD}$, $\texttt{BF-RWA}^{STD}$ e $\texttt{BFD-RWA}^{STD}$ para as 187 instâncias nos conjuntos K, Y, Z e W. As heurísticas foram executadas cinco vezes para cada instância, com diferentes sementes para o gerador de números aleatórios [86]. A Tabela 3.1 mostra o desvio relativo médio e o tempo de processamento médio de $\texttt{FF-RWA}^{STD}$, $\texttt{FFD-RWA}^{STD}$, $\texttt{BF-RWA}^{STD}$e $\texttt{BFD-RWA}^{STD}$ para cada conjunto de instâncias separadamente. Os valores médios sobre as 187 instâncias são mostrados nas Figuras 3.2(a) e 3.2(b), respectivamente.

A heurística $\texttt{BFD-RWA}^{STD}$ encontrou resultados em média melhores que as outras heurísticas para a maioria das instâncias testadas. Observa-se na Figura 3.2(a) que o desvio relativo médio de $\texttt{BFD-RWA}^{STD}$ (6,0%) é menor

Tabela 3.1: Desvios relativos médios e tempos de execução médios (em segundos) de FF-RWA^{STD}, BF-RWA^{STD}, FFD-RWA^{STD} e BFD-RWA^{STD} para as instâncias dos conjuntos K, Y, Z e W.

	FF-RWA		FFD-RWA		BF-RWA		BFD-RWA	
Conjunto	Δ (%)	t (s)	Δ (%)	t (s)	Δ (%)	t (s)	Δ (%)	t (s)
K	4,7	0,60	1,9	0,71	3,0	1,33	1,2	2,10
Y	23,1	0,52	17,0	0,58	13,9	0,76	8,4	1,07
Z	13,3	0,89	9,7	1,12	10,8	1,10	7,0	1,41
W	7,3	0,10	6,3	0,10	7,5	0,11	7,1	0,12

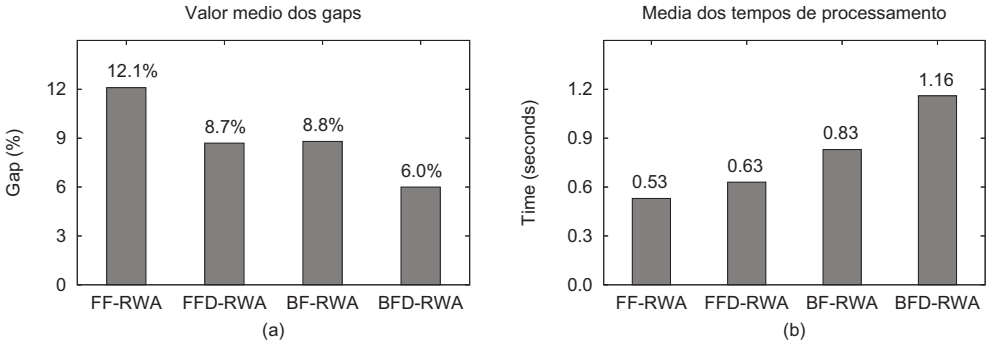

Figura 3.2: (a) Desvios relativos médios e (b) tempos de execução médios (em segundos) de FF-RWA^{STD}, BF-RWA^{STD}, FFD-RWA^{STD} e BFD-RWA^{STD} sobre as 187 instâncias.

do que a metade daquele associado a FF-RWA^{STD} (12,1%) e pouco maior que dois terços daqueles associados a FFD-RWA^{STD} (8.7%) e BF-RWA^{STD} (8.8%). A heurística BFD-RWA^{STD} encontrou a solução ótima para 62 das 75 instâncias do conjunto K e o desvio relativo médio neste conjunto foi 1,2%, o que confirma a hipótese de que a maioria das instâncias neste conjunto são de fácil resolução. Os desvios relativos médios para as instâncias dos conjuntos Y e Z, propostos nesta tese, foram maiores ou iguais aos desvios relativos médios para as instâncias dos conjuntos K e W para as quatro heurísticas, indicando que as novas instâncias propostas são realmente mais difíceis do que as instâncias já existentes na literatura.

Como esperado, os tempos de processamento das heurísticas BF-RWA^{STD} e BFD-RWA^{STD} foram maiores do que os de FF-RWA^{STD} e FFD-RWA^{STD}, pois na linha 8 da figura 2.6, FF-RWA^{STD} e FFD-RWA^{STD} selecionam qualquer cópia de N em Ω na qual o caminho ótico $t_{\pi(i)}$ pode ser roteado com menos de d arcos, enquanto BF-RWA^{STD} e BFD-RWA^{STD} examinam todas as cópias de N, em busca daquela onde $t_{\pi(i)}$ pode ser roteado com o menor número possível de arcos. Apesar do desvio relativo médio de BFD-RWA^{STD} ter sido menor do

que o das outras heurísticas, seus tempos computacionais em média foram os maiores. Entretanto, os tempos de execução de $\texttt{BFD-RWA}^{STD}$ para o conjunto K de instâncias foram sempre inferiores a 10 segundos, bem menores que os 8 minutos relatados em [88] para a mesma heurística, o mesmo conjunto de instâncias e a mesma configuração de máquina.

O segundo experimento avalia e compara o desempenho das implementações de `FF-RWA`, `FFD-RWA`, `BF-RWA` e `BFD-RWA` usando NRRt, NRRg, RRt e RRg. Os tempos de execução são comparados com aqueles usando a respectiva implementação padrão (STD). Para cada heurística e cada conjunto de instâncias, a Tabela 3.2 mostra os tempos médios de execução das quatro heurísticas implementadas com NRRt, NRRg, RRt e RRg divididos pelos tempos das respectivas heurísticas implementadas com STD. As heurísticas foram executadas cinco vezes para cada instância com diferentes sementes para o gerador de números aleatórios [86]. As implementações usando RRt e NRRt foram mais rápidas do que aquelas usando RRg e NRRg, respectivamente. Isto se deve ao fato de que os grafos de caminhos mais curtos não foram suficientemente densos para compensar o maior custo de atualização em relação às árvores de caminhos mais curtos. Isto já era esperado, devido ao fato das topologias de redes estudadas serem esparsas. Também pode-se observar na Tabela 3.2 que as implementações das quatro heurísticas usando RRt e RRg foram mais lentas do que usando a implementação padrão proposta nesta tese. Isto se deve ao grande número de atualizações de grafos de caminhos mais curtos que têm que ser realizadas em RRt e RRg após a remoção de um arco.

NRRt mostrou-se o melhor algoritmo para atualização da matriz de distâncias. A porcentagem de melhora de NRRt em relação a STD para as quatro heurísticas nos quatro conjuntos de instâncias é plotada na figura 3.3. As melhorias observadas em $\texttt{BF-RWA}^{NRRt}$ e $\texttt{BFD-RWA}^{NRRt}$ são maiores que as observadas em $\texttt{FF-RWA}^{NRRt}$ e $\texttt{FFD-RWA}^{NRRt}$. Isto se deve ao fato de que o número de consultas à matriz de distâncias em `BF-RWA` e `BFD-RWA` é maior que em `FF-RWA` e `FFD-RWA`, enquanto o número de atualizações da matriz de distâncias é aproximadamente o mesmo. A heurística que tirou mais proveito de NRRt foi `BFD-RWA`, cujo tempo médio de execução reduziu-se quase à metade daqueles utilizando a implementação padrão. O maior tempo de execução de $\texttt{BFD-RWA}^{NRRt}$ sobre as 187 instâncias testadas foi inferior a 3 segundos, bem menor que os 10 segundos consumidos por $\texttt{BFD-RWA}^{STD}$. Portanto, $\texttt{BFD-RWA}^{NRRt}$ foi o algoritmo utilizado para a implementação da heurística `MS-BFD`.

O terceiro experimento compara o desempenho da heurística construtiva `BFD-RWA` e da heurística de multi-partida `MS-BFD` para as 112 instâncias nos

Tabela 3.2: Média dos tempos de execução das diferentes implementações das heurísticas `FF-RWA`, `FFD-RWA`, `BF-RWA` e `BFD-RWA` nos conjuntos de instâncias K, Y, Z e W. Os tempos são apresentados como o desvio percentual em relação aos tempos das respectivas heurísticas implementadas com STD.

Heurística	Conjunto	NRRt (%)	NRRg (%)	RRt (%)	RRg (%)
`FF-RWA`	K	88,3	115,6	671,8	817,4
	Y	83,9	107,6	699,9	824,9
	Z	73,4	95,8	745,7	884,4
	W	98,4	119,0	423,8	497,6
`FFD-RWA`	K	84,8	112,4	562,2	690,1
	Y	81,1	103,9	626,0	739,1
	Z	62,0	79,7	595,2	699,1
	W	93,4	111,8	406,6	471,3
`BF-RWA`	K	50,3	67,5	254,2	333,2
	Y	67,8	88,1	400,7	496,2
	Z	63,4	83,4	568,1	685,7
	W	91,8	114,4	363,7	431,5
`BFD-RWA`	K	37,1	50,0	163,8	209,2
	Y	52,8	70,4	277,3	349,0
	Z	53,2	67,9	440,4	528,8
	W	88,7	107,5	344,0	407,5

conjuntos Y, Z e W. O critério de parada de `MS-BFD` foi estabelecido em 1000 iterações.

Os resultados para o conjunto Y são apresentados na Tabela 3.3. Para cada grupo de cinco instâncias no conjunto Y, as primeiras quatro colunas da Tabela 3.3 mostram o nome, o número de nós, a probabilidade γ de existir um enlace entre um par de nós na rede e a probabilidade ρ de existir uma requisição de caminho ótico entre dois pares de nós na rede. As duas colunas seguintes apresentam, respectivamente, a média do desvio relativo das soluções e a média dos tempos de execução da heurística `BFD-RWA` em cinco execuções com diferentes sementes para o gerador de números aleatórios [86]. Os mesmos resultados são apresentados para `MS-BFD` nas duas últimas colunas. Pode-se observar que `MS-BFD` é capaz de melhorar a qualidade das soluções com respeito a `BFD-RWA`. O desvio relativo médio das soluções obtidas por `MS-BFD` para todas as instâncias foi 6,5%, enquanto o mesmo valor para `BFD-RWA` foi 8,4%. Além disso, `MS-BFD` forneceu certificados de otimalidade (ou seja, soluções cujo custo é igual ao limite inferior) para 36 das 75 instâncias no conjunto Y, enquanto `BFD-RWA` encontrou soluções ótimas para apenas 24 delas.

A Tabela 3.4 apresenta os resultados para o conjunto Z de instâncias. As primeiras quatro colunas mostram o nome, o número de nós, o grau dos

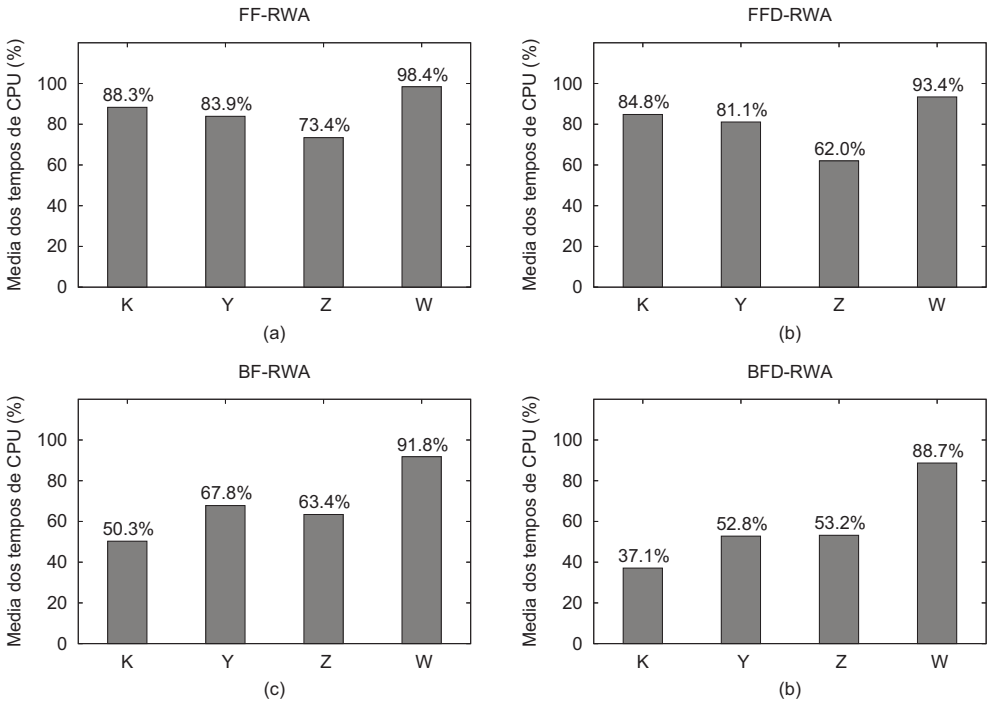

Figura 3.3: Média dos tempos de execução de (a) $\texttt{FF-RWA}^{NRRt}$, (b) $\texttt{FFD-RWA}^{NRRt}$, (c) $\texttt{BF-RWA}^{NRRt}$ e (d) $\texttt{BFD-RWA}^{NRRt}$ nos conjuntos de instâncias K, Y, Z e W. Os tempos são apresentados como um desvio percentual dos tempos de $\texttt{BF-RWA}^{STD}$.

nós e o valor da probabilidade ρ para cada grupo de instâncias. As duas colunas seguintes apresentam, respectivamente, a média do desvio percentual das soluções e a média dos tempos de execução da heurística `BFD-RWA` em cinco execuções usando diferentes sementes para o gerador de números aleatórios [86]. Os mesmos resultados são apresentados para `MS-BFD` nas duas últimas colunas. Os desvios relativos das soluções encontradas pelas duas heurísticas para as instâncias do conjunto Z foram em média menores do que aqueles obtidos para as instâncias no conjunto Y. Entretanto, nenhuma solução ótima foi encontrada pelas duas heurísticas. Isto se deve à alta simetria na topologia das redes do conjunto Z. Para este conjunto, a melhoria média na qualidade das soluções observada por `MS-BFD` com respeito a `BFD-RWA` foi de apenas 1%. No entanto, vale a pena salientar que, para a maioria das instâncias de min-RWA, é relativamente fácil encontrar soluções com D comprimentos de onda, mas muito difícil encontrar soluções usando apenas um comprimento de onda a menos. Sendo assim, qualquer melhoria no custo da solução é significante.

Os resultados numéricos para as instâncias do conjunto W são relatados na Tabela 3.5. As três primeiras colunas mostram o nome, o número de nós e o número de enlaces para cada instância, enquanto as duas colunas seguintes

Tabela 3.3: Resultados computacionais de `BFD-RWA` e `MS-BFD` para o conjunto de instâncias Y.

				`BFD-RWA`		`MS-BFD`	
Grupo	nós	γ	ρ	Δ (%)	t (s)	Δ (%)	t (s)
Y.3.20	100	0,03	0,2	14,1	0,1	11,8	73,3
Y.4.20	100	0,04	0,2	14,7	0,1	11,1	59,7
Y.5.20	100	0,05	0,2	7,6	0,1	3,2	48,6
Y.3.40	100	0,03	0,4	11,3	0,2	9,3	174,8
Y.4.40	100	0,04	0,4	10,6	0,2	8,8	142,3
Y.5.40	100	0,05	0,4	5,0	0,1	2,7	108,0
Y.3.60	100	0,03	0,6	8,6	0,3	7,3	305,4
Y.4.60	100	0,04	0,6	8,9	0,3	7,8	245,2
Y.5.60	100	0,05	0,6	5,6	0,2	2,9	180,9
Y.3.80	100	0,03	0,8	7,3	0,5	6,1	455,5
Y.4.80	100	0,04	0,8	8,0	0,4	7,0	368,1
Y.5.80	100	0,05	0,8	5,3	0,3	3,4	263,0
Y.3.100	100	0,03	1,0	6,7	0,7	6,0	630,2
Y.4.100	100	0,04	1,0	8,0	0,5	6,9	500,7
Y.5.100	100	0,05	1,0	4,0	0,4	3,0	359,9
			Média:	8,4	0,3	6,5	261,0

mostram o número total de caminhos óticos de cada instância e o número máximo de caminhos óticos partindo de um mesmo nó da rede. A sexta e a sétima coluna apresentam, respectivamente, a média do desvio relativo das soluções e a média dos tempos de execução sobre cinco execuções de `BFD-RWA`. Os mesmos resultados são relatados para `MS-BFD` nas duas últimas colunas. Para este conjunto de instâncias, o desvio relativo médio das instâncias retornadas por `MS-BFD` foi quase a metade daquele das instâncias retornadas por `BFD-RWA`. Além disso, `MS-BFD` encontrou a solução ótima para seis dentre as 12 instâncias do conjunto W (EON, ATT2, NSF.48, NSF2.1, NSF2.3 e NSF2.48). Já a heurística `BFD-RWA` encontrou a solução ótima apenas para as instâncias EON e NSF2.48.

Neste capítulo, foram propostos algoritmos e estruturas de dados que permitiram a implementação eficiente das heurísticas propostas em [88]. Os resultados computacionais mostram que ao contrário do observado em [88], `BFD-RWA` obtém soluções melhores que `BF-RWA` e `FFD-RWA` na maioria das instâncias testadas. Os tempos computacionais da melhor implementação de `BFD-RWA` foram sempre menores que três segundos, enquanto os tempos relatados para a mesma heurística em [88] chegaram a oito minutos nas mesmas instâncias e no mesmo computador Pentium IV 2.8 GHz. A melhor implementação proposta para a heurística `BFD-RWA` foi utilizada na construção de um heurística de multi-partida para min-RWA que obteve desvios relativos

Tabela 3.4: Resultados computacionais de `BFD-RWA` e `MS-BFD` para o conjunto de instâncias Z.

				`BFD-RWA`		`MS-BFD`	
Grupo	nós	grau	ρ	Δ (%)	t (s)	Δ (%)	t (s)
Z.4×25.20	100	4	0,2	4,5	0,1	3,0	99,9
Z.5×20.20	100	4	0,2	3,3	0,1	1,9	84,1
Z.6×17.20	102	4	0,2	6,8	0,1	4,5	73,2
Z.8×13.20	104	4	0,2	9,1	0,1	9,1	57,4
Z.10×10.20	100	4	0,2	19,3	0,1	18,5	46,2
Z.4×25.40	100	4	0,4	3,3	0,2	2,4	218,1
Z.5×20.40	100	4	0,4	4,4	0,2	3,0	184,8
Z.6×17.40	102	4	0,4	5,7	0,2	4,5	160,8
Z.8×13.40	104	4	0,4	7,9	0,1	6,3	124,4
Z.10×10.40	100	4	0,4	16,5	0,1	15,7	97,1
Z.4×25.60	100	4	0,6	3,1	0,4	2,1	371,1
Z.5×20.60	100	4	0,6	3,0	0,3	2,6	310,8
Z.6×17.60	102	4	0,6	4,8	0,3	3,9	272,2
Z.8×13.60	104	4	0,6	6,7	0,2	5,2	207,6
Z.10×10.60	100	4	0,6	15,6	0,2	14,3	159,2
Z.4×25.80	100	4	0,8	2,3	0,6	1,2	529,3
Z.5×20.80	100	4	0,8	2,9	0,5	2,0	446,5
Z.6×17.80	102	4	0,8	3,7	0,4	3,2	394,0
Z.8×13.80	104	4	0,8	5,3	0,3	4,2	300,3
Z.10×10.80	100	4	0,8	13,6	0,3	12,6	225,7
Z.4×25.100	100	4	1,0	2,7	0,7	2,2	703,2
Z.5×20.100	100	4	1,0	3,2	0,6	2,8	596,6
Z.6×17.100	102	4	1,0	3,7	0,6	3,2	526,9
Z.8×13.100	104	4	1,0	4,9	0,5	4,2	416,1
Z.10×10.100	100	4	1,0	14,7	0,3	13,5	295,1
			Média:	6,8	0,3	5,8	276,0

médios iguais a 6,5%, 5,5% e 3,8% para as instâncias dos conjuntos Y, Z e W, respectivamente.

Tabela 3.5: Resultados computacionais de `BFD-RWA` e `MS-BFD` para o conjunto de instâncias W.

			Caminhos óticos		`BFD-RWA`		`MS-BFD`	
Instância	nós	enlaces	total	máximo	Δ (%)	t (s)	Δ (%)	t (s)
Finland	31	51	930	1	3,0	0,0	2,2	9,6
EON	20	39	374	2	0,0	0,0	0,0	2,1
ATT	90	137	359	5	32,0	0,0	24,0	19,8
ATT2	71	175	4456	34	2,1	0,2	0,7	140,8
NSF.1	14	21	284	3	6,4	0,0	4,5	0,9
NSF.3	14	21	258	3	8,2	0,0	4,5	0,8
NSF.12	14	21	551	6	8,9	0,0	4,7	1,8
NSF.48	14	21	547	6	3,4	0,0	2,0	1,8
NSF2.1	14	22	284	3	5,7	0,0	0,0	0,8
NSF2.3	14	22	258	3	6,7	0,0	0,0	0,8
NSF2.12	14	22	551	6	6,3	0,0	2,9	1,7
NSF2.48	14	22	547	6	1,5	0,0	0,0	1,7
				Média:	7.0	0.0	3.8	15.2