Universidade do Estado do Rio Grande do Norte
Universidade Federal Rural do Semi-árido
Programa de Pós-Graduação em Ciência de Computação

Otimização na reparação da rede viária após desastres de grandes proporções

Thiago Jobson Barbalho

Mossoró-RN
Agosto, 2019

Thiago Jobson Barbalho

Otimização na reparação da rede viária após desastres de grandes proporções

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação das Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Linha de pesquisa: Otimização e Inteligência Artificial.

Orientador

Prof. Dr. Dario José Aloise

Co-orientadora

Prof[a]. Dr[a]. Andréa Cynthia Santos

PPgCC – Programa de Pós-Graduação em Ciência da Computação
DI – Departamento de Informática
UERN – Universidade do Estado do Rio Grande do Norte
UFERSA – Universidade Federal Rural do Semi-Árido

Mossoró-RN

Agosto, 2019

Catalogação da Publicação na Fonte.
Universidade do Estado do Rio Grande do Norte.

B228o Barbalho, Thiago Jobson
Otimização na reparação da rede viária após desastres de grandes proporções. / Thiago Jobson Barbalho. - Mossoró, RN, 2019.
84p.

Orientador(a): Prof. Dr. Dario José Aloise.
Dissertação (Mestrado em Programa de Pós-Graduação em Ciência da Computação). Universidade do Estado do Rio Grande do Norte.

1. Programa de Pós-Graduação em Ciência da Computação. 2. Otimização combinatória. 3. Gestão de crises. I. Aloise, Dario José. II. Universidade do Estado do Rio Grande do Norte. III. Título.

O serviço de Geração Automática de Ficha Catalográfica para Trabalhos de Conclusão de Curso (TCC´s) foi desenvolvido pela Diretoria de Informatização (DINF), sob orientação dos bibliotecários do SIB-UERN, para ser adaptado às necessidades da comunidade acadêmica UERN.

Dissertação sob o título *Otimização na reparação da rede viária após desastres de grandes proporções* apresentada por Thiago Jobson Barbalho e aceita pelo Programa de Pós-Graduação em Ciência da Computação das Universidade do Estado do Rio Grande do Norte e Universidade Federal Rural do Semi-Árido, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

Prof. Dr. Dario José Aloise
Orientador e Presidente
UERN – Universidade do Estado do Rio Grande do Norte

Profª. Drª. Andréa Cynthia Santos
Co-orientadora
UTT – *Université de Technologie de Troyes*

Prof. Dr. Francisco Chagas de Lima Júnior
Examinador
UERN – Universidade do Estado do Rio Grande do Norte

Prof. Dr. Carlos Heitor Pereira Liberalino
Examinador
UERN – Universidade do Estado do Rio Grande do Norte

Prof. Dr. Hugo Alexandre Dantas do Nascimento
Examinador externo
UFG – Universidade Federal de Goiás

Mossoró-RN, 15 de Agosto de 2019.

Aos meus pais.

Agradecimentos

Agradeço a Deus por toda força que Ele me deu para continuar mesmo nos momentos difíceis.

Não tenho palavras para agradecer aos meus orientadores, professor Dario e professora Andréa Cynthia, por toda dedicação, motivação e paciência que me deram durante a realização deste trabalho.

Obrigado ao meu querido amigo Isaías por toda força que me deu durante esses anos de mestrado. Sem seu apoio tudo teria sido muito mais difícil. Agradeço também a Jonathan e ao professor Fábio Fontes por terem me ajudado a entender as formulações matemáticas quando iniciei a pesquisa.

Agradeço aos meus colegas de mestrado por toda companhia, conversas e apoio mútuo. Em especial a Wilton, Daniel, Felipe, Nícholas e Geo. Estou ansioso para ver o futuro brilhante que terão.

Agradeço a Eliene, Paloma, Regina, Lorena e as outras meninas da cantina da UFERSA por terem me alimentado e me mantido abastecido de café durante as manhãs e tardes que passei na biblioteca implementando os meus algoritmos e escrevendo este trabalho. Aquele cubículo na biblioteca sabe que não foi nada fácil.

Obrigado à CAPES pelo apoio financeiro e à Universidade do Estado do Rio Grande do Norte (UERN) e Universidade Federal Rural do Semi-Árido (UFERSA), por todo suporte dado para a realização desta pesquisa.

Otimização na reparação da rede viária após desastres de grandes proporções

Autor: Thiago Jobson Barbalho
Orientador: Prof. Dr. Dario José Aloise
Co-orientadora: Prof[a]. Dr[a]. Andréa Cynthia Santos

Resumo

Após um desastre de grandes proporções, como na ocorrência de um grande terremoto, a rede viária de uma cidade pode sofrer sérios danos e as ruas serem bloqueadas por escombros de casas e prédios que colapsam, dificultando a locomoção da população, que se junta e busca abrigo em agrupamentos espontâneos pela cidade. A acessibilidade da rede se torna uma preocupação importante para as equipes que são responsáveis por ajudar os feridos e distribuir suprimentos aos desabrigados. Este trabalho busca estudar e apresentar maneiras de otimizar a reparação das ruas de uma cidade atingida por uma grande catástrofe, permitindo que a acessibilidade da rede viária melhore o mais rápido possível. O problema de decisão associado à reparação de vias é NP-Difícil, o que dificulta a utilização de algoritmos exatos para grafos do tamanho de uma cidade, com milhares de vértices e arestas. Este trabalho também apresenta as primeiras meta-heurísticas e de busca local aplicadas ao problema. Os resultados mostram soluções expressivamente melhores do que heurísticas simples presentes na literatura.

Palavras-chave: Gerenciamento pós-desastres, Logística humanitária, Redes viárias, Meta-heurísticas.

Optimizing road network after major disasters

Author: Thiago Jobson Barbalho
Supervisor: Prof. D. Sc. Dario José Aloise
Co-supervisor: Prof. D. Sc. Andréa Cynthia Santos

Abstract

After a major disaster, as in the aftermath of a major earthquake, the road network can be blocked by debris from collapsed buildings, affecting the accessibility to the population. In addition, people move and gather together in some points in the city. Road network accessibility becomes an important issue for logistics operations responsible for the relief and supplies distributions to population. We study and introduce new methods to rehabilitate the road network after a major disaster, improving the network accessibility. The decision problem associated with this problem is NP-Hard, preventing the usage of exact algorithms in graphs with thousands of nodes and edges (size of real cities). To the best of our knowledge, the first metaheuristics for the problem and local search methods are proposed in this study. Results show that solutions are expressively better than the greedy heuristics in the literature.

Keywords: Post disaster relief, Humitarian logistics, Road networks, Metaheuristics.

Lista de algoritmos

Lista de tabelas

Lista de figuras

Sumário

1 Introdução

Um desastre é um evento súbito e calamitoso que traz danos, perdas e devastação à vida e à propriedade. Quando uma cidade é atingida por um grande desastre natural, isso afeta diretamente o dia-a-dia das pessoas atingidas, influencia negativamente os sistemas de emergência e, dependendo da intensidade e gravidade, deteriora os serviços normais da população, como o acesso à água potável, alimentação, abrigo e acessibilidade (SHANKAR, 2011).

Fato dessa natureza ocorreu em 2010, na cidade de Porto Príncipe, capital do Haiti, atingida por um terremoto de magnitude 7.0 na escala Richter, seguido por diversos outros tremores de magnitude superior a 6.0. Neste terremoto, casas e prédios colapsaram, matando, ferindo e desabrigando a população (SAKURABA et al., 2016; BRILHAM, 2010). Esforços realizados pelos cidadãos e por organizações internacionais para prover assistência médica, comida e água aos sobreviventes foram dificultados pela falha do sistema de rede elétrica, perda de linhas de comunicação e ruas bloqueadas por detritos de desabamentos (PALLARDY, 2019).

No decorrer de uma catástrofe natural de grande magnitude, as pessoas atingidas tendem a se agrupar em acampamentos espontâneos. Nas primeiras 72 horas críticas após a catástrofe, a sobrevivência desta população depende diretamente da ajuda de equipes de socorro para ter acesso a suprimentos e alívio aos feridos. Uma das complicações comuns nessa situação é que a malha viária da cidade pode sofrer sérios danos (SANTOS, 2018), dificultando a acessibilidade e até mesmo podendo levar ao isolamento de áreas da cidade, impedindo que as equipes de socorro acessem os acampamentos de pessoas (FENG; WANG, 2003).

Define-se o problema que trata da acessibilidade e do alívio da população após um grande desastre como *RERP (Road Emergency Rehabilitation Problem)* (SAKURABA et al., 2016), onde o objetivo é reparar, com a ajuda de máquinas, a rede viária danificada e aumentar a acessibilidade dentro da cidade, possibilitando que as pessoas atingidas

recebam ajuda das equipes que fornecem suprimentos e socorro aos feridos.

1.1 Objetivos de pesquisa

Os métodos utilizados atualmente para determinar a reparação/limpeza das vias são modelos lineares e heurísticas simples. O primeiro tipo tem dificuldade em lidar com grafos maiores do que 20 vértices; o segundo, utiliza paradigmas e estratégias puramente gulosas.

O objetivo deste trabalho é desenvolver e avaliar as meta-heurísticas GRASP (do inglês: *Greedy Randomized Adaptive Search Procedure*) e ILS (do inglês: *Iterated Local Search*) para determinar a escala de reparação de uma rede viária danificada por um desastre natural. O motivo da escolha destas meta-heurísticas em particular é o de reutilizar/adaptar as heurísticas puramente gulosas já existentes na literatura como fases de construção. Tais heurísticas já foram aplicadas com sucesso em instâncias de diferentes tamanhos.

1.1.1 Objetivos específicos

O objetivo de pesquisa mencionado acima pode ser detalhado nas seguintes atividades específicas:

- Escolher as heurísticas para utilizar como fase de construção para o GRASP, que serão analisadas em termos de afinidade com o paradigma guloso-aleatório do método, compatibilidade com a técnica de lista de candidatos, qualidade de solução e complexidade computacional;
- Examinar o comportamento da heurística escolhida quando esta é parametrizada pelo α (alfa) do GRASP e determinar, através de experimentos de calibração, o valor de α que gera melhores soluções;
- Idealizar e propor um formato de solução para ser utilizado por este trabalho e trabalhos futuros;
- Desenvolver um método de busca local que melhore a qualidade das soluções criadas pela fase construtiva e que funcione para o formato de solução idealizado;
- Desenvolver o método ILS utilizando o movimento na vizinhança da busca local como forma de perturbação de soluções;

- Avaliar os resultados dos métodos GRASP e ILS desenvolvidos com os resultados dos métodos já existentes na literatura.

1.2 Metodologia de pesquisa

A metodologia científica utilizada nesta dissertação pode ser resumido nos itens a seguir:

- Replicação das heurísticas existentes na literatura;
- Implementação dos métodos GRASP e ILS utilizando as melhores heurísticas como fase de construção da solução inicial;
- Desenvolvimento de uma busca local e um modelo de vizinhança para as soluções construídas pelas heurísticas;
- Aplicação do GRASP e ILS às instâncias do problema;
- Comparação entre os resultados dos métodos da literatura com os método GRASP e ILS desenvolvidos;
- Conclusão retirada das implementações, experimentos e dos resultados obtidos;

1.3 Estrutura do documento

Para melhor entendimento do trabalho, este documento está dividido da seguinte forma: o Capítulo 2 contém a definição formal e a contextualização do problema tratado nesta dissertação, consistindo de seções de modelagem matemática, histórico e trabalhos relacionados; o Capítulo 3 apresenta uma revisão bibliográfica sobre as heurísticas existentes na literatura para o problema; e o desenvolvimento das meta-heurísticas, com a análise das heurísticas construtivas, escolha do formato de solução, detalhes de implementação e busca local; o Capítulo 4 apresenta os resultados dos experimentos de calibração para identificar o melhor valor de α para o GRASP e os resultados comparativos dos experimentos computacionais das instâncias; no Capítulo 5 são feitas conclusões sobre o trabalho e possibilidades de trabalhos futuros. Por fim, o Apêndice A apresenta uma prova de que o problema tratado nesta dissertação é NP-Difícil, e o Apêndice B lista os resultados detalhados dos experimentos computacionais para todas as instâncias.

2 Definição do problema e trabalhos relacionados

Neste capítulo, são apresentados os fundamentos do problema abordado neste trabalho. As próximas seções trazem a definição formal para o *Road Emergency Rehabilitation Problem*, por meio de uma formulação matemática e revisão da literatura.

2.1 *Road Emergency Accessibility Problem*

Define-se o problema que trata da acessibilidade após um grande desastre como *RERP (Road Emergency Rehabilitation Problem)* (SAKURABA et al., 2016). O RERP é dividido em dois subproblemas: RNAP (*Road Network Acessibility Problem*) e o WSP (*Work-troops Scheduling Problem*) , descritos nas Seções 2.1.1 e 2.1.2 seguintes.

Um grafo $G = (V, E)$ é definido e utilizado por ambos os subproblemas, onde V é o conjunto de n vértices e E é o conjunto de m arestas não-direcionadas. O subconjunto $O \subset V$ representa as origens, os pontos de entrada da rede de onde partem as WT (do inglês: *work-troops*), referindo-se às máquinas de trabalho que fazem as reparações das ruas: caminhões, tratores, escavadeiras, etc.; o subconjunto $D \subset V$, representa os acampamentos de pessoas ($O \cap D = \emptyset$), a quantidade de pessoas em cada acampamento é indicada pela constante $p_i \in \mathbb{N}^*$. Cada origem tem uma quantidade $q_i \in \mathbb{N}^*$ de WT disponíveis ($\sum_{i \in O} q_i = Q$) onde Q é o total de WT distribuídas entre todas as origens. O objetivo é reparar as vias danificadas do grafo com as WT disponíveis.

No grafo $G = (V, E)$, as vias são representadas por arestas $[i, j]$, com $i < j$. Cada aresta possui três parâmetros: o comprimento físico da via, $d_{ij} \in \mathbb{R}_+^*$; $u_{ij} \in \mathbb{N}^*$, a largura física da via, que implica em um limite máximo de WT que podem trabalhar em uma mesma extremidade da rua ao mesmo tempo; e $r_{ij} \in \mathbb{N}^*$, uma relação entre o dano sofrido pela via e a quantidade de trabalho necessária para repará-la completamente. Além disso, define-se $B \subset E$ como o subconjunto de vias que estão danificadas em decorrência do

desastre, isto é, ruas bloqueadas não trafegáveis em que $r_{ij} > 0$.

A Figura 1 ilustra um exemplo de um grafo $G = (V, E)$, onde V ={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}, E = {[1, 2], [1, 3], [1, 4], [2, 3], [2, 5], [2, 6], [3, 4], [3, 6], [3, 7], [4, 7], [4, 10], [5, 6], [5, 8], [6, 7], [6, 8], [6, 9], [7, 9], [7, 10], [8, 9], [9, 10]} e $B = E\backslash$[3, 6], [4, 10], [6, 7], [7, 9], [8, 9]}, onde as origens são representadas por nós em formato de círculo (1 e 8); acampamentos, por losangos (4, 5 e 7) e nós de trasbordo, por quadrados. Vias bloqueadas ($r_{ij} > 0$) são representados por arestas pontilhadas e não permitem passagem, as demais arestas ($r_{ij} = 0$) são transitáveis a custo d_{ij}. A largura $u_{ij} = 1$ das vias está oculta. A rede urbana representada por G, neste exemplo, mostra que a acessibilidade até os acampamentos 4 e 5 está comprometida e não é possível alcançá-los a partir das origens 1 ou 8, pois não existe um caminho composto totalmente de arestas disponíveis ($r_{ij} = 0$) entre origens e acampamentos.

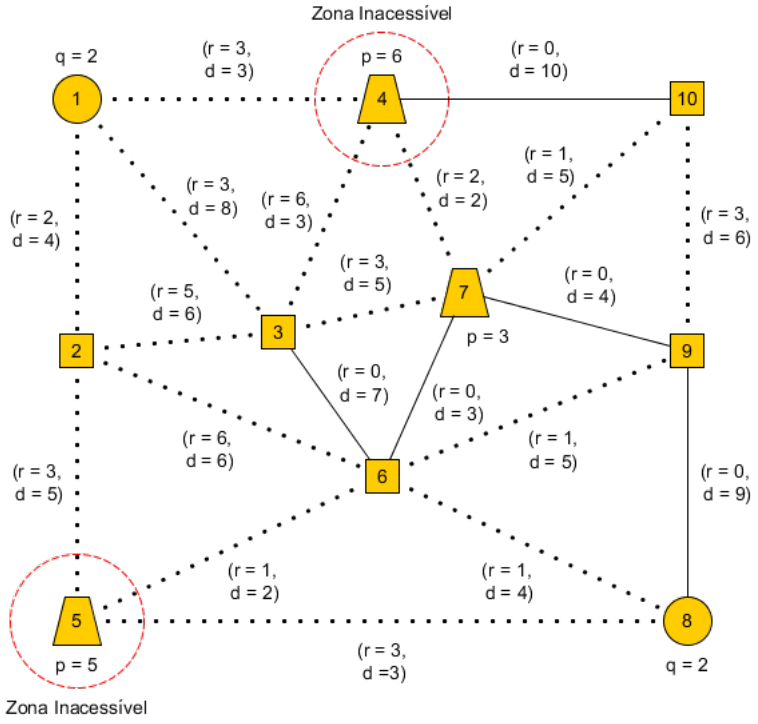

Figura 1: Exemplo de um grafo $G = (V, E)$.
Fonte: Autoria própria.

Os modelos presentes nas subseções seguintes fazem uso de G', um digrafo obtido pela conversão das arestas não-direcionais $[i, j] \in E$ por arcos (i, j) e (j, i) unidirecionais, gerando o conjunto de arcos A. Para proporcionar um ponto único de entrada ao digrafo G', adiciona-se uma raiz artificial (vértice 0) conectada a todas as origens por arcos $(0, i)$, com $d_{0i} = r_{0i} = 0, \forall i \in O$. Por fim, G' é composto dos vértices $V' = V \cup \{0\}$ e o conjunto

de arcos $A' = A \cup \{(0,i)|i \in O\}$, assim, $G' = (V', A')$. A Figura 2 ilustra o exemplo do grafo $G = (V, E)$ da Figura 1 convertido em um digrafo $G' = (V', A')$. Inicialmente, cada par de arcos (i, j) e (j, i) compartilha dos mesmos atributos da aresta $[i, j]$ que os originou. Nota-se que os arcos que conectam a raiz artificial (vértice 0) às origens são unidirecionais e possuem atributos r_{ij}, d_{ij} e u_{ij} iguais a 0, impedindo que haja passagem de um lado ao outro do grafo *através* dela.

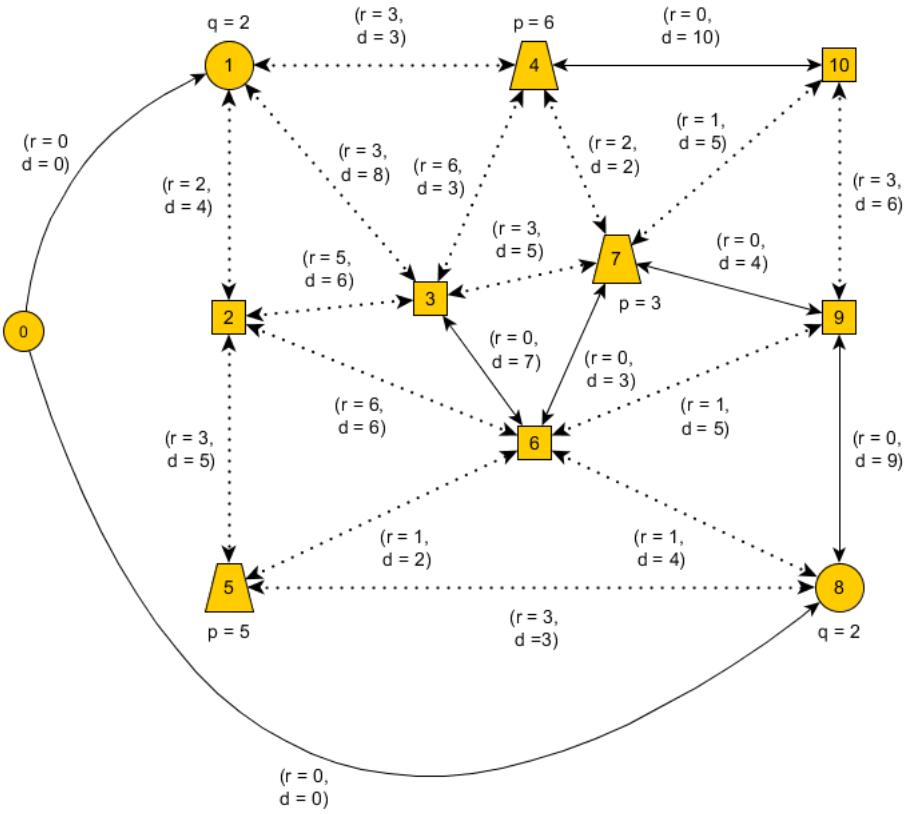

Figura 2: Exemplo do grafo $G = (V, E)$ da Figura 1 transformado em $G' = (V', A')$.
Fonte: Autoria própria.

2.1.1 *Road Network Accessibility Problem*

O primeiro dos subproblemas que compõe o RERP é conhecido como RNAP (*Road Network Accessibility Problem*) que corresponde a um modelo polinomial, segundo Sakuraba et al. (2016), e busca *calcular uma estimativa* do tempo mínimo de reparação necessário para alcançar a população que está nos acampamentos. Nos casos em que esta população se encontra em zonas isoladas do grafo e não existe um caminho trafegável de alguma origem até ela, o RNAP busca avaliar quais ruas devem ser reparadas prioritariamente para que seja possível alcançar essas zonas no menor tempo possível. Para o

RNAP, a limitação do número de WT não é considerada e os destinos não precisam ser alcançados a partir de uma mesma origem.

A ideia da formulação do RNAP é enviar fluxos a partir da raiz artificial (vértice 0) para cada um dos acampamentos. A variável $t_{ij} = [\frac{r_{ij}}{u_{ij}}]K + d_{ij}$ descreve os custos de cada arco $(i,j) \in A^{'}$, onde K é o peso para dar prioridade ao reparo de arestas bloqueadas e deve ser escolhido de forma que $[\frac{r_{ij}}{u_{ij}}]K > d_{ij}$ para todos os arcos. As variáveis x_{ij} e f_{ij} estão diretamente relacionadas; a primeira denota se o arco (i,j) é usado em algum caminho no período t, enquanto a segunda expressa a quantidade de fluxo no arco (i,j). A variável y_i estima o tempo em que o vértice i pode ser alcançado a partir da raiz artificial. Caso um vértice não possa ser alcançado, o seu valor de y_i é igual a constante M, um inteiro consideravelmente grande que representa um custo de travessia excessivamente alto. A Tabela 1 apresenta uma referência para as variáveis e constantes da formulação matemática do RNAP (Equações 2.1 - 2.10), proposta em Sakuraba et al. (2016).

Tabela 1: Tabela de referência para as variáveis e constantes do modelo do RNAP.

	Descrição
Variáveis	
y_i	Tempo necessário para alcançar o nó i a partir da raiz artificial (vértice 0)
f_{ij}	Quantidade de fluxo que passa por (i,j)
x_{ij}	Variável binária que denota se existe, ou não, fluxo que passa por (i,j)
Constantes	
p_i	População existente em i
t_{ij}	Tempo necessário para transpor (i,j)
0 (zero)	Raiz artificial
M	Tempo excessivamente alto
Conjuntos	
D	Conjunto de acampamentos

$$min\ z = \sum_{i \in D} p_i y_i\ s.a. \tag{2.1}$$

$$\sum_{(0,i) \in A' : i \in O} f_{0i} = |D| \tag{2.2}$$

$$\sum_{i:(i,j) \in A'} f_{ij} - \sum_{i:(j,i) \in A'} f_{ji} = 0 \qquad \forall j \in V^{'} \backslash D \tag{2.3}$$

$$\sum_{i:(i,j) \in A'} f_{ij} - \sum_{i:(j,i) \in A'} f_{ji} = 1 \qquad \forall j \in D \tag{2.4}$$

$$y_i - y_j + (M + t_{ij}) x_{ij} \leq M \qquad \forall (i,j) \in A^{'} \tag{2.5}$$

$$x_{ij} \geq \frac{f_{ij}}{|D|} \qquad \forall (i,j) \in A^{'} \qquad (2.6)$$

$$x_{ij} \leq f_{ij} \qquad \forall (i,j) \in A^{'} \qquad (2.7)$$

$$y_i \geq 0 \qquad \forall i \in V^{'} \qquad (2.8)$$

$$f_{ij} \geq 0 \qquad \forall (i,j) \in A^{'} \qquad (2.9)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A^{'} \qquad (2.10)$$

O objetivo (2.1) é alcançar, a partir da raiz artificial 0, o maior número de pessoas no menor tempo possível. As equações (2.2) – (2.4) representam as restrições de conservação de fluxo que saem da raiz artificial com destino aos acampamentos de pessoas. A inequação (2.5) estima o tempo em que cada destino é alcançado a partir de uma origem acumulando distâncias entre vértices de maneira semelhante ao algoritmo de caminho mínimo de Dijkstra (DIJKSTRA, 1959). As restrições (2.6) e (2.7) estabelecem a relação entre f_{ij} e x_{ij}. As variáveis são definidas em (2.8) – (2.10). Esta formulação contém $O(|A^{'}|)$ variáveis e restrições.

2.1.2 *Work-troops Scheduling Problem*

O WSP (*Work-troops Scheduling Problem*) (SAKURABA et al., 2016; SAKURABA; SANTOS; PRINS, 2016) visa definir a planificação das WT que fazem a reparação das vias bloqueadas, diminuindo as distâncias e melhorando a acessibilidade das origens até os acampamentos o mais rápido possível. A formulação matemática 2.11 - 2.28 considera o aspecto dinâmico do problema, a evolução do estado da rede viária conforme as vias são reparadas, recalculado os caminhos mínimos (SP) para os acampamentos em cada período de tempo.

O modelo do WSP tem dois componentes principais: um fluxo da raiz artificial para cada acampamento e um fluxo de WT das origens até a extremidade dos arcos. Define-se $B^{'} \subset A$ como o conjunto de todos os arcos direcionais obtidos pela substituição de arestas bloqueadas $[i,j] \in B$ por arcos (i,j) e (j,i); considera-se que o fluxo de WT não pode passar por um arco $(i,j) \in B^{'}$, e seu valor d_{ij} é definido como um inteiro excessivamente grande M.

O modelo do WSP funciona da seguinte forma: a reparação do grafo é discretizada em períodos de tempo $t = 1, ..., T$, e ao final de cada período de tempo t, a variável x_{ij}^{t} denota *se* existe fluxo passado pelo arco (i,j), enquanto $f_{i,j}^{t}$ representa a *quantidade* de fluxo passando por (i,j). Além disso, a variável z_{ij}^{t} especifica se a aresta $[i,j]$ está

disponível ou não ($\sum_{[i,j]\in B} z_{ij}^0 = 0$), i.e., se permite a passagem de WT. A variável s_i^t guarda a distância entre a raiz artificial 0 e o vértice i. Durante cada período, o fluxo de WT no arco (i,j) é determinado pela variável y_{ij}^t, enquanto w_{ij}^t representa o fluxo de WT entrando no arco (i,j) para repará-lo. A Tabela 2 apresenta uma referência rápida para as variáveis e constantes da formulação matemática do WSP (Equações 2.11 - 2.28), proposta por Sakuraba et al. (2016).

Tabela 2: Tabela de referência para as variáveis e constantes do modelo do WSP.

	Descrição
Variáveis	
s_i^t	Distância da raiz artificial até i no período t
f_{ij}^t	Quantidade de fluxo passando por (i,j) no período t
x_{ij}^t	Variável binária que denota se existe, ou não, fluxo passando por (i,j) no período t
y_{ij}^t	Fluxo de WT passando por (i,j) no período t
w_{ij}	Fluxo de WT reparando (i,j) no período t
z_{ij}^t	Denota se $[i,j]$ está disponível (reparada) no período t
Constantes	
p_i	População existente em i
r_{ij}	Quantidade de trabalho necessário para reparar $[i,j]$
d_{ij}	Comprimento físico de $[i,j]$
u_{ij}	Largura física de $[i,j]$
q_i	Quantidade de máquinas inicialmente na origem i
0 (zero)	Raiz artificial
M	Distância excessivamente alta
Conjuntos	
D	Conjunto de acampamentos
B	Conjunto de arestas bloqueadas

$$min\ Z = \sum_{i\in D}\sum_{t=1}^{T} p_i s_i^t\ s.a. \tag{2.11}$$

$$\sum_{i:(i,j)\in A'} f_{ij}^t - \sum_{i:(j,i)\in A'} f_{ji}^t = \begin{cases} D, se\ i = 0 \\ 0, se\ \forall j \in V\backslash D \\ 1, se\ \forall j \in D \end{cases} \qquad t = 1, ..., T \tag{2.12}$$

$$x_{ij}^t \geq \frac{f_{ij}^t}{|D|} \qquad \forall (i,j) \in A', t = 1, ..., T \tag{2.13}$$

$$x_{ij}^t \leq f_{ij}^t \qquad \forall (i,j) \in A', t = 1, ..., T \tag{2.14}$$

$$\sum_{j:(i,j)\in A} y_{ij}^t + \sum_{k:(i,k)\in B'} w_{ik}^t \leq q_i \qquad \forall i \in O, t = 1, ..., T \tag{2.15}$$

$$\sum_{i:(i,j)\in A} y_{ij}^t - \sum_{i:(j,i)\in A} y_{ji}^t = \sum_{k:(j,k)\in B'} w_{jk}^t \qquad \forall j \in V\backslash O, t = 1, ..., T \qquad (2.16)$$

$$\sum_{[i,j]\in B} z_{ij}^t = |B| \qquad (2.17)$$

$$r_{ij} z_{ij}^t \leq \sum_{t'=1}^{t} (w_{ij}^{t'} + w_{ji}^{t'}) \qquad \forall [i,j] \in E, t = 1, ..., T \qquad (2.18)$$

$$y_{ij}^t + y_{ji}^t \leq \sum_{i\in O} q_i z_{ij}^{t-1} \qquad \forall [i,j] \in E, t = 1, ..., T \qquad (2.19)$$

$$w_{ij}^t \leq u_{ij} \qquad \forall (i,j) \in A, t = 1, ..., T \qquad (2.20)$$

$$s_j^t \geq M(x_{ij}^t - z_{ij}^t) \qquad \forall (i,j) \in A', t = 1, ..., T \qquad (2.21)$$

$$s_i^t - s_j^t + (M + d_{ij}) x_{ij} \leq M \qquad \forall (i,j) \in A', t = 1, ..., T \qquad (2.22)$$

$$f_{ij}^t \geq 0 \qquad \forall (i,j) \in A', t = 1, ..., T \qquad (2.23)$$

$$s_i^t \geq 0 \qquad \forall i \in V', t = 1, ..., T \qquad (2.24)$$

$$w_{ij}^t \geq 0 \qquad \forall (i,j) \in A, t = 1, ..., T \qquad (2.25)$$

$$y_{ij}^t \geq 0 \qquad \forall (i,j) \in A, t = 1, ..., T \qquad (2.26)$$

$$x_{ij}^t \in \{1, 0\} \qquad \forall (i,j) \in A', t = 1, ..., T \qquad (2.27)$$

$$z_{ij}^t \in \{1, 0\} \qquad \forall [i,j] \in E, t = 1, ..., T \qquad (2.28)$$

A função objetivo (2.11) é a soma ponderada das distâncias da raiz artificial multiplicada pela população de cada um dos acampamentos durante todos os períodos. As equações (2.12) garantem que exista um fluxo unitário da raiz artificial para cada acampamento, enquanto as inequações (2.13) e (2.14) relacionam as variáveis f_{ij}^t e x_{ij}^t. As restrições (2.15) e (2.16) limitam o fluxo de WT ao máximo de q_i WTs inicialmente nas origens, alocando-as para arcos bloqueados. A equação (2.17) garante que todas as arestas inicialmente bloqueadas sejam reparadas até o último período T, enquanto que as inequações (2.18) calculam a disponibilidade de uma aresta de acordo com a quantidade de trabalho recebido das WT. As restrições (2.19) forçam as WT a transitarem somente por arestas disponíveis. As inequações (2.20) limitam o número de WT que podem trabalhar em cada extremidade das vias de acordo com a sua largura física. As restrições (2.21) fixam a distância até um vértice a M se todos os caminhos estiverem bloqueados. A distância da raiz artificial até cada vértice é calcula pelas inequações (2.22), de maneira semelhante ao algoritmo de caminho mínimo de Dijkstra, acumulando-se as distâncias entre vértices. As variáveis são definidas em (2.23) – (2.28). Esta formulação contém $O(|A'|T)$ variáveis e restrições.

2.2 Trabalhos relacionados

Diversos trabalhos sobre logística humanitária e gerenciamento pós-catástrofes têm surgido nos últimos anos. Tópicos variam de relatos do impacto de desastres, reparo dos danos causados; evacuação da população atingida; e alocação de diferentes tipos de recursos escassos para melhor atender aos sobreviventes. Uma boa introdução a este campo de pesquisa pode ser encontrada em Doerner, Gutjahr e Wassenhove (2011), Hu, Liu e Jiang (2012) e Zheng, Chen e Ling (2015).

Lu, Bengtsson e Holme (2012) utilizaram dados de telefonia da maior operadora de celulares do Haiti para analisar o movimento de 1.9 milhão de pessoas no período de 42 dias antes a 341 dias após o terremoto que devastou o Haiti, em 2010. Dezenove dias após o desastre, a população em Porto Príncipe diminuiu em 23%. Os dados foram utilizados para prever o movimento da população durante os meses que sucederam a catástrofe. Dada a importância de conhecer a localização das pessoas atingidas, os resultados mostraram que este tipo de predição possui uma boa acurácia, mesmo em meio ao cenário catastrófico de Porto Príncipe.

Os trabalhos de Hirayama et al. (2010) e Yücel, Salman e Arsik (2018) apresentam modelos preditivos para estimar os danos causados por terremotos. Em Hirayama et al. (2010), um procedimento que utiliza mapas de risco a partir de dados observados de terremotos reais foi aplicado a uma zona metropolitana de Tóquio, onde mostrou que a quantidade de detritos pode ser estimada para ser utilizada em planejamento pré-catástrofes. Yücel, Salman e Arsik (2018) apresentam um estudo para aumentar a resiliência da rede viária contra desastres, otimizando os investimentos para fortalecer a estrutura. Os autores apresentam um modelo estocástico de dependência para prever o estado da rede. O objetivo é reforçar um conjunto de vias para que a expectativa de acessibilidade pós-catástrofe seja maximizada. A acessibilidade é medida pela soma ponderada das distâncias entre depósitos e pontos de demanda. Os autores desenvolveram uma meta-heurística Busca Tabu eficiente que gera soluções e testa o impacto na acessibilidade de acordo com o modelo de dependência e aplicaram a um grafo da cidade de Istambul com 60 nós e 83 arestas.

Feng e Wang (2003) apresentam um modelo matemático de natureza estática e multiobjetivo que maximiza a acessibilidade e o número de vidas salvas, enquanto minimiza os riscos envolvidos na reparação de nós de uma rede atingida por um desastre natural. O modelo também utiliza o conceito de *work-troops*, porém a reparação se dá nos nós, e não nas arestas, com a restrição de que cada nó seja reparado por somente uma *work-troop*.

O modelo foi aplicado e resolvido de forma ótima em um grafo com 52 nós e 62 arestas da cidade de Chi-Chi, Taiwan, após um terremoto de magnitude 7.3, em 1999.

Duque e Sörensen (2011) apresentam o problema de alocar uma quantidade escassa de recursos para reparar uma rede de estradas rurais após ter sido danificada por uma catástrofe natural. Os autores propõem uma solução baseada na meta-heurística GRASP com VNS (*Variable Neighborhood Search*), com o objetivo de maximizar a acessibilidade do maior número de pessoas até as cidades próximas da zona atingida. Embora parecido com o WSP, o problema tratado em Duque e Sörensen (2011) tem natureza estática e conta com um orçamento inicial que deve ser otimizado para reparar as estradas, onde cada estrada possui um custo proporcional ao dano sofrido, aproximando-se de um problema de empacotamento. Além disso, Duque e Sörensen (2011) utilizaram um subgrafo da cidade de Porto Príncipe após o terremoto de 2010 para analisar o cenário com orçamentos limitados a 25%, 50% e 75% do total necessário para reparar todas as estradas.

Aksu e Ozdamar (2014) apresentam dois modelos matemáticos para planejar a reparação de uma rede danificada. O primeiro calcula a planificação da reparação de caminhos predefinidos, onde o objetivo é reparar todos os caminhos o mais cedo possível. O segundo é um modelo tático que aloca as *work-troops* de forma equitativa entre as origens do grafo. O estudo de caso envolveu dois distritos de uma região da Turquia com 212 e 386 vias, das quais 49 e 79 vias estavam bloqueadas, respectivamente. Os caminhos a serem restaurados foram predefinidos como rotas de evacuação conhecidas.

Pramudita, Taniguchi e Qureshi (2014) tratam do problema de remover os detritos que bloqueiam as vias de uma cidade após a ocorrência de um evento calamitoso. Os autores apresentam o problema como uma variante do problema de roteamento de veículos capacitados, onde os caminhões que coletam os detritos fazem uma rota, removem os destroços das vias e retornam ao depósito. O trabalho propõe um modelo matemático e uma meta-heurística Busca Tabu para resolver instâncias do problema, onde a função objetivo é a minimização dos custos de viagem dos veículos. Os autores utilizaram a estimativa calculada por Hirayama et al. (2010) para o potencial de dano gerado por terremotos e inundações na área metropolitana de Tóquio, com 196 nós e 98 arcos para realizar experimentos.

Lu et al. (2016) apresentam o problema causado por condições de mau tempo e outros eventos menores de tráfego que causam a diminuição na capacidade do fluxo de uma rede urbana, onde a recuperação do fluxo é realizada por uma única equipe. As vias críticas da rede são identificadas como *links* e o objetivo é minimizar a soma dos custos de travessia

de toda a rede. O problema é resolvido de forma ótima utilizando um algoritmo guloso que aplica consecutivamente uma decisão localmente ótima em cada estágio, visando encontrar o ótimo global. O algoritmo prioriza os *links* de acordo com a economia nos custos de travessia proporcionado após sua recuperação. Como estudo de caso, realizaram um experimento utilizando o grafo de Sioux Falls, contendo 24 nós e 76 arestas. O experimento supôs que 8 *links* estavam danificados e a redução da capacidade das vias reduzida entre 75% e 80%. A principal diferença do problema tratado em Lu et al. (2016) para o WSP é de que a equipe de reparação não está fisicamente associada a nenhum nó; o tempo de reparação de uma via é sempre o mesmo, independente da sua deterioração da capacidade.

Duque, Dolinskaya e Sörensen (2016) tratam da reparação de emergência em uma rede viária rural danificada. O problema é definido como um grafo não-direcionado em que um subconjunto dos nós está danificado e necessita de reparação; nós podem possuir demanda que deve ser suprida a partir do nó de depósito. Os nós de demanda representam pequenas cidades ou assentamentos e a sua importância pode ser medida a partir da quantidade de população no local. A função objetivo é a minimização da soma dos momentos em que cada nó de demanda torna-se acessível ponderado pela demanda do nó. Os autores propõem um algoritmo de programação dinâmica exato e uma meta-heurística gulosa-aleatória baseado no GRASP para resolver o problema. Experimentos foram realizados utilizando dois conjuntos de instâncias como estudo de caso: o primeiro, de tamanho pequeno com até 41 nós; e o segundo, de tamanhos médio e grande, com até 401 nós. As instâncias menores do problema foram resolvidas de forma ótima utilizando o algoritmo exato. A meta-heurística desenvolvida foi capaz de encontrar o resultado ótimo em 92,8% dos casos para as instâncias pequenas. As principais diferenças entre o problema e o WSP é que o primeiro utiliza grafos esparsos de redes viárias rurais com pequenas cidades conectadas a um centro regional. A reparação da rede é realizada por uma única equipe, a partir de um único depósito. Além disto, o problema não exige que todos os nós sejam reparados, mas somente que nenhum ponto de demanda permaneça isolado a partir do depósito.

Kim et al. (2018) propõem um problema baseado em Duque, Dolinskaya e Sörensen (2016), mas adicionam aspectos dinâmicos no período imediato após um desastre (*short-term*). Danos adicionais podem acontecer, piorando a situação da rede viária. Os autores apresentam um modelo matemático de programação inteira mista que considera uma única equipe de reparo partindo de um nó origem. O modelo minimiza a soma ponderada dos danos e o tempo de reparo dos nós da rede. Diversas instâncias de testes com até 23

nós foram criadas. Uma meta-heurística Colônia de Formigas também foi aplicada para resolver o problema, onde as formigas começam a se mover a partir nó de origem e escolhem o próximo nó danificado de acordo com uma probabilidade, gerando a planificação de reparo. A principal diferença para o WSP é de que desastres adicionais podem acontecer, piorando a situação da população nos nós isolados.

Akbari e Salman (2017) apresentam um problema cujo objetivo é minimizar o tempo de caminhada no grafo que visite as vias bloqueadas e repare-as de forma que a conectividade da rede seja restaurada. Múltiplos depósitos e múltiplas *work-troops* são considerados. O autores utilizaram um modelo de programação inteira mista que é capaz de resolver de forma ótima instâncias de 40 nós e 2 *work-troops* em até 1 hora. Para resolver instâncias maiores, o trabalho propõe uma matheurística que resolve uma versão relaxada do problema. Em seguida, aplica-se um algoritmo de viabilidade capaz de transformar uma solução relaxada em uma solução para o problema original. Como estudo de caso, foram utilizados diversos conjuntos de instâncias e diferentes configurações de depósitos e quantidade de *work-troops* variando de 1 a 6. Três deles originados da rede viária de Istambul, com 74 nós e 179 arestas para o conjunto menor; 349 nós e 689 arestas para o conjunto detalhado e 250 nós e 539 arestas da região ocidental da cidade. A principal diferença para o WSP é de que uma via bloqueada é reparada por uma única *work-troop*. Também não é exigido que todas as arestas sejam reparadas, mas somente aquelas que recuperem a conectividade da rede.

Vodák, Bíl e Křivánková (2018) apresentam o problema de reparar um subconjunto das vias bloqueadas para que a conectividade do grafo seja restaurada. Para lidar com grafos grandes, os autores apresentam um algoritmo capaz de reduzir a rede viária em um grafo mais simples. A redução é feita agrupando diversos nós adjacentes em componentes independentes. A função objetivo é a minimização do tempo necessário para as unidades de reparo visitarem todos os componentes. Três cenários para o problema são apresentados: (i) as unidades de reparo devem visitar todos os componentes pelo menos uma vez e as vias bloqueadas podem ser transpassadas sem custo; (ii) as unidades de reparo devem visitar todos os componentes ao menos uma vez, porém as vias bloqueadas podem ser transpassadas se forem reparadas primeiro, adicionando um custo; (iii) mesmo que o primeiro cenário, porém os componentes possuem um fator de importância que deve ser levado em conta na ordem de visita. Os autores propõem uma meta-heurística Colônia de Formigas modificada para resolver o problema, onde as rotas são construídas por formigas até que pelo menos um nó em cada componente seja visitado. O estudo de caso utilizou o grafo de uma região de Zlín, na República Tcheca, com 723 nós e 974 arestas com 46

ruas bloqueadas e 16 componentes independentes. As principais diferenças do problema tratado em Vodák, Bíl e Křivánková (2018) para WSP é de que as unidades de reparo podem transpassar vias bloqueadas sem custo (em alguns cenários); as vias bloqueadas são reparadas somente por uma unidade e somente as vias que recuperam a conectividade da rede precisam ser restauradas.

2.2.1 Histórico do *Work-troops Scheduling Problem*

O WSP (Seção 2.1.2) é definido por Sakuraba, Santos e Prins (2015), como *"We defined the Work-troops Scheduling Problem (WSP) as generating a multi-period scheduling [...] to repair blocked edges in order to improve as fast as possible the accessibility of the population to the relief teams."*[1], em um trabalho que utilizou os dados da rede viária de Porto Príncipe, Haiti, após ser atingida por um terremoto de magnitude 7.0, em 2010. A rede de transportes da capital do Haiti contém milhares de vértices e arestas, e seus dados foram fornecidos pelo acordo internacional *The International Charter: Space and Major Disasters* (ICSMD, 2019).

Após o terremoto, as ruas da cidade foram parcialmente ou totalmente bloqueadas, como mostrada na Figura 3, impedindo que veículos pudessem acessar certas regiões da cidade.

Figura 3: Da esquerda para a direita: exemplo de ruas levemente, moderadamente e severamente bloqueadas.

Fonte: Sakuraba, Santos e Prins (2015).

Como forma de gerar uma programação de trabalho para as máquinas que fazem a reparação das vias bloqueadas, o trabalho de Sakuraba, Santos e Prins (2015) utilizou uma heurística gulosa. Instâncias de tamanho pequeno (20 nós) foram criadas para que

[1] Tradução: Definimos o *Work-troops Scheduling Problem* (WSP) como gerar uma programação de múltiplos períodos [...] para reparar as arestas bloqueadas e melhorar o mais rápido possível a acessibilidade da população e das equipes de socorro.

fosse possível comparar os resultados da heurística com os ótimos calculados pelo CPLEX, embora a formulação matemática não tenha sido apresentada no trabalho publicado. Estes foram os primeiros resultados heurísticos para instâncias do problema e, nas instâncias simuladas pelo CPLEX, a heurística apresentou resultados que se distanciam em até 8% do ótimo.

O WSP também foi tratado em Sakuraba, Santos e Prins (2016) em um trabalho realizado pelo projeto OLIC (*Optimisation de la Logistique d'Intervention pour les Catastrophes majeures*). Este trabalho utilizou os mesmos dados da rede viária da cidade de Porto Príncipe e também as instâncias de pequeno porte capazes de ser resolvidas pelo CPLEX, e apresentou uma formulação matemática para o WSP.

Ainda em Sakuraba, Santos e Prins (2016), duas novas heurísticas gulosas foram elaboradas para planificar a reparação das vias bloqueadas. Os autores também introduziram o conceito de acessibilidade, definida como o tempo necessário para estabelecer caminhos trafegáveis das origens (vértices de onde partem as equipes de socorro) até os acampamentos (pontos onde se encontram a população de pessoas atingidas pela catástrofe). O conceito de acessibilidade passou também a ser utilizado como uma métrica de comparação entre os resultados das heurísticas e do modelo exato. Em Sakuraba et al. (2016), os resultados das heurísticas de Ranqueamento, Economias e da Heurística de Classificação Lexicográfica foram compilados e comparados com os resultados ótimos calculados pelo CPLEX pela métrica de acessibilidade.

3 Heurísticas e meta-heurísticas para o *Work-troops Scheduling Problem*

Neste trabalho, as três heurísticas propostas em Sakuraba et al. (2016) foram reproduzidas e descritas na Seção 3.1. Na Seção 3.2, são apresentadas as meta-heurísticas GRASP (*Greedy Randomized Adaptive Search Procedure*) e ILS (*Iterated Local Search*) propostas neste trabalho para o WSP (*Work-troops Scheduling Problem*). Os detalhes para desses métodos são apresentados, incluindo seus respectivos pseudocódigos.

3.1 Heurísticas construtivas e gulosas para o WSP

Três heurísticas são propostas e utilizadas para resolver o WSP. Cada uma delas aplica um critério predefinido para classificar e priorizar as vias, alocando WT às arestas bloqueadas durante os períodos de reparação. Todas as heurísticas fazem uso do algoritmo de Dijkstra para calcular os caminhos mínimos entre origens e acampamentos, e também verificar se existe acessibilidade até a extremidade da aresta que se deseja reparar. As seções seguintes apresentam e explicam um pseudocódigo para cada uma destas heurísticas.

3.1.1 Heurística de Ranque

A heurística de Ranque (em inglês: *Ranking heuristic*) (SAKURABA et al., 2016) utiliza um critério guloso de avaliação de arestas para elaborar uma lista ranqueada em que o ranque de cada aresta é atribuído pela quantidade de vezes que esta aparece em caminhos mínimos entre origens e acampamentos. Para este procedimento, a heurística de Ranque computa os caminhos mínimos ótimos entre todos os pares (o, d), com $o \in O$ e $d \in D$ e conta a ocorrência de cada via bloqueada presente nestes caminhos. Depois da avaliação, as arestas são ordenadas pelo seu ranque em ordem decrescente.

O primeiro período de tempo é tratado de modo especial: devido a análises realizadas nos resultados produzidos pelo modelo matemático. Claramente, os resultados indicam que os caminhos com poucas reparações são priorizados para serem reparados no primeiro período de tempo. Por isto, no primeiro período de tempo da heurística de Ranque, os caminhos mínimos com uma única aresta bloqueada, onde $u_{ij} >= r_{ij}$ (largura da via maior ou igual à estimativa de tempo de reparo), são reparados. Em seguida, a alocação das WT às arestas é feita a partir da cabeça da lista ranqueada, configurando um comportamento puramente guloso. Em caso de arestas bloqueadas com um mesmo ranque, o critério de desempate é feito pela aresta de menor r_{ij}, pois exige menos tempo de trabalho.

O Algoritmo 1 apresenta o pseudocódigo da heurística de Ranque. As linhas 2 - 10 calculam o ranque das arestas bloqueadas nos caminhos mínimos (linha 5) entre todos os pares de origens e acampamentos. As linhas 6 e 7 apresentam o caso especial explicado no parágrafo anterior: os caminhos que possuem somente uma única aresta bloqueada são selecionados e adicionados a uma lista de prioridade LP. A instrução 11 ordena as arestas bloqueadas em ordem decrescente de ranque. As linhas 12 e 13 verificam se existe algum caminho mínimo em LP com somente uma aresta bloqueada e $u_{ij} \geq r_{ij}$ para repará-lo no primeiro período. Os passos 15 - 22 reparam as vias bloqueadas a partir do início da lista ranqueada. A linha 20 atualiza os caminhos mínimos do grafo quando uma aresta bloqueada é totalmente reparada.

A heurística de Ranque possui complexidade $O(|E|lg|V|)$ para classificar todos os candidatos inicialmente e $O(|B||E|lg|V|)$ para reparar todas as vias, onde E é o conjunto de arestas no grafo; V, o conjunto de vértices e $B \subset E$, o conjunto de arestas bloqueadas. O termo $|E|lg|V|$ é a complexidade assintótica do algoritmo de caminhos mínimos.

3.1.2 Heurística de Economias

A heurística de Economias (SAKURABA et al., 2016) é uma heurística que se assemelha a de Ranque, pois também utiliza uma lista ordenada. Mas, diferentemente desta, as vias bloqueadas são avaliadas pelo critério de economia obtida na função objetivo caso elas sejam reparadas. A economia de uma aresta é calculada pela redução das distâncias ponderadas entre origens e acampamentos no período. Para isso, para cada um dos períodos $t = 1, ..., T$, a heurística precisa recalcular as economias nas distâncias utilizando um algoritmo de caminhos mínimos e ordenar as arestas bloqueadas em ordem decrescente de economia.

O Algoritmo 2 apresenta o pseudocódigo para a heurística de Economias. A mesma

Algoritmo 1 Heurística de Ranque

```
1: procedimento RANKING(G = (V, E), O, D, B)
2:     para todo o ∈ O faça
3:         Calcule os caminho mínimos (CM) para os vértices usando r_ij como custo
4:         para todo d ∈ D faça
5:             Adicione 1 ao ranque de cada [i, j] ∈ B nos CM de d
6:             se CM até d possui somente uma aresta bloqueada então
7:                 Adicione esta aresta bloqueada à lista LP
8:             fim se
9:         fim para
10:    fim para
11:    Ordene as arestas bloqueadas em ordem decrescente de ranque
12:    se existe aresta em LP com u_ij ≥ r_ij então
13:        Aloque as WTs para estas arestas imediatamente
14:    fim se
15:    para t ← 1 até T faça
16:        para todo WT disponível faça
17:            Aloque a WT para a primeira via bloqueada a partir da cabeça da lista
18:        fim para
19:        se alguma [i, j] ∈ B tornou-se disponível então
20:            Atualize os CM utilizando a nova aresta disponível
21:        fim se
22:    fim para
23:    retorne A programação de WT e o valor da Função Objetivo
24: fim procedimento
```

estratégia empregada na heurística de Ranque para o primeiro período é utilizada na heurística de Economias, nas instruções das linhas 2 - 9. A linha 11 calcula as economias geradas por cada uma das arestas. A instrução da linha 13 ordena as arestas bloqueadas em ordem decrescente de economia. Os passos 17 - 28 reparam as arestas bloqueadas a partir do inicio da lista ordenada. A atualização dos caminhos mínimos do grafo quando uma aresta bloqueada é totalmente reparada é realizada na linha 22. A linha 24 recalcula as economias. A instrução na linha 26 reordena as arestas com os novos cálculos de economia sempre que o estado do grafo muda.

Algoritmo 2 Heurística de Economias

```
1:  procedimento SAVINGS(G = (V, E), O, D, B)
2:      para todo i ∈ O faça
3:          Calcule os caminhos mínimos (CM) para os vértices usando r_ij como custo
4:          para todo j ∈ D faça
5:              se CM para j possui somente uma aresta bloqueada então
6:                  Adicione esta aresta bloqueada à lista LP
7:              fim se
8:          fim para
9:      fim para
10:     para todo [i, j] ∈ B faça
11:         Calcule a economia se a aresta [i, j] for reparada
12:     fim para
13:     Ordene as arestas bloqueadas em ordem decrescente de economia
14:     se existe aresta em LP com u_ij ≥ r_ij então
15:         Aloque as WTs para estas arestas imediatamente
16:     fim se
17:     para t ← 1 até T faça
18:         para todo WT disponível faça
19:             Aloque a WT para a primeira aresta acessível a partir da cabeça da lista
20:         fim para
21:         se alguma [i, j] ∈ B tornou-se disponível então
22:             Atualize os CM utilizando a nova aresta disponível
23:             para todo [i, j] ∈ B faça
24:                 Calcule a economia se a aresta [i, j] for reparada
25:             fim para
26:             Ordene as arestas bloqueadas em ordem decrescente de economia
27:         fim se
28:     fim para
29:     retorne A programação de WT e valor da Função Objetivo
30: fim procedimento
```

A heurística de Economias possui complexidade $O(|B|^2|E|lg|V|)$ para reparar todas as vias, onde E é o conjunto de arestas no grafo; V, o conjunto de vértices e $B \subset E$, o conjunto de arestas bloqueadas. O termo $|E|lg|V|$ é atribuído a um algoritmo de caminhos

mínimos.

3.1.3 Heurística de Classificação Lexicográfica

A Heurística de Classificação Lexicográfica (em inglês: LCH – *Lexicographic Classification Heuristic*) (SAKURABA et al., 2016) classifica as vias em dois conjuntos. O primeiro deles, $S1$, é o conjunto de arestas bloqueadas que levam às zonas isoladas o mais rápido possível, isto é, arestas com o menor parâmetro r_{ij} e que, quando reparadas, desbloqueiam um caminho até os acampamentos; o conjunto $S2$ é de arestas que pertencem aos caminhos mínimos ótimos (em termos de distância), independentes de estarem bloqueadas ou não. A ordem de prioridade em que as arestas são reparadas é dada pela tripla $(S_1, S_2, \{a \in B \mid a \notin S_1 \; e \; a \notin S_2\})$ onde o conjunto S_1 tem prioridade sobre S_2.

O Algoritmo 3 apresenta o pseudocódigo para a LCH. As linhas 2 - 4 selecionam as arestas do conjunto S1 baseado no modelo do RNAP. O algoritmo de Dijkstra é utilizado para identificar as vias que podem ser reparadas o mais rápido possível e que desbloqueiam a passagem até os acampamentos. Na linha 5, o algoritmo de Dijkstra é utilizado novamente para calcular os caminhos ótimos utilizando o atributo d_{ij} da aresta. As arestas bloqueadas nestes caminhos são adicionadas ao conjunto $S2$. Por fim, as arestas no conjunto S1 são mantidas ordenadas pelo atributo r_{ij}.

Uma vez que os conjuntos $S1$ e $S2$ são construídos, o processo de reparação começa (linhas 6 - 21), e as WT são alocadas pela ordem: se houver algum acampamento inalcançável, a WT é alocada para a primeira aresta bloqueada em $S1$ (linhas 8 - 10). Se todos os acampamentos podem ser alcançados, ou não for possível alocar a WT a uma aresta em $S1$, a WT é então enviada para reparar a primeira aresta do conjunto $S2$ (linhas 11 - 13). O último critério de prioridade é se o conjunto $S2$ estiver vazio, por exemplo, caso todas as arestas bloqueadas estiverem reparadas, ou se não for possível reparar alguma aresta do conjunto, então aloca-se a WT a alguma aresta bloqueada de forma aleatória (linhas 14 - 16). Por fim, a linha 19 atualiza os caminhos mínimos do grafo quando uma aresta bloqueada é totalmente reparada.

A LCH possui complexidade $O(|E|lg|V|)$ para classificar as arestas no conjunto $S1$, $O(|E|lg|V|)$ para classificar arestas no conjunto $S2$ e $O(|B||E|lg|V|)$ para reparar todas as vias, onde E é o conjunto de arestas no grafo; V, o conjunto de vértices e $B \subset E$, o subconjunto de arestas bloqueadas. O termo $|E|lg|V|$ é atribuído a um algoritmo de caminhos mínimos.

Algoritmo 3 Heurística de Classificação Lexicográfica

procedimento LCH($G = (V, E), O, D, B$)
    para todo $i \in O$ **faça**
        Resolva o modelo do RNAP para construir o conjunto S1
    fim para
    Calcule os caminhos mínimos (CM) da raiz artificial para todo $i \in D$ utilizando o parâmetro d_{ij} para construir S2
    para $i \leftarrow 1$ até T **faça**
        para todo WT disponível **faça**
            se Existe um acampamento que não pode ser alcançado **então**
                Aloque a WT para a primeira aresta acessível em S1
            fim se
            se WT ainda não foi alocada **então**
                Aloque a WT para a primeira aresta acessível em S2
            fim se
            se WT ainda não foi alocada **então**
                Aloque a WT aleatoriamente para alguma aresta bloqueada
            fim se
        fim para
        se alguma $[i, j] \in B$ tornou-se disponível **então**
            Atualize os CM utilizando a nova aresta disponível
        fim se
    fim para
    retorne A programação de WT e valor da Função Objetivo
fim procedimento

3.2 Meta-heurísticas aplicadas ao WSP

Os métodos GRASP (FEO; RESENDE, 1989) e ILS (LOURENÇO; MARTIN; STÜTZLE, 2003) foram escolhidos para serem aplicados ao WSP porque foram utilizados com sucesso em muitos problemas de otimização combinatória. Além disso, essas duas meta-heurísticas têm poucos parâmetros a serem calibrados. Também, algumas das heurísticas presentes na literatura foram adaptadas para serem usadas em ambas meta-heurísticas propostas.

As seções seguintes detalham os métodos GRASP e ILS para o WSP. Inicialmente, esses métodos são apresentados e, em seguida, a sua aplicação para o WSP é descrita.

3.2.1 *Greedy Randomized Adaptive Search Procedure*

O GRASP é um método iterativo e multipartida que combina as boas características dos algoritmos puramente gulosos e dos procedimentos aleatórios na sua fase de construção de soluções; posteriormente, um procedimento de busca local é utilizado para refinar a solução inicial (FEO; RESENDE, 1989). O papel da primeira fase é de criar soluções utilizando dois paradigmas distintos: *gula* e *aleatoriedade*. O equilíbrio entre estes paradigmas é feito pelo parâmetro α (alfa). A segunda fase de uma iteração é realizada por um algoritmo de busca local, que faz melhoria da solução criada pela fase de construção. Usualmente, várias iterações são realizadas e a melhor solução gerada dentre todas é guardada e exibida como resultado final (FESTA; RESENDE, 2002; RESENDE; FESTA, 2008).

O critério de parada do GRASP é um parâmetro que pode ser definido como um número máximo de iterações, um tempo máximo de execução ou um número máximo de iterações sem melhoria da melhor solução encontrada. Este último leva em consideração quantas fases de construção e busca local ocorreram sem que uma melhor solução tenha sido encontrada. Quase sempre, o tempo de execução não varia muito de iteração para iteração e o tempo total de processamento é previsível, aumentando linearmente conforme o número de iterações (RESENDE; RIBEIRO, 2003).

Em geral, construir uma solução é computacionalmente barato e o tempo gasto na fase de construção do GRASP utiliza apenas uma fração do tempo de processamento total. Por outro lado, a fase de busca local é a que demanda um maior tempo de execução e sua eficácia depende de aspectos como a estrutura de vizinhança definida; a técnica de busca na vizinhança, o processo de avaliação das soluções vizinhas; e também da solução de partida. É nesse momento que uma boa fase de construção se mostra importante, pois elaborar soluções de qualidade para serem utilizadas como ponto de partida pela busca

local pode implicar em menos tempo de processamento para alcançar um ótimo local (RESENDE; RIBEIRO, 2003; BINATO et al., 2000).

Considerando um problema de minimização, uma iteração do GRASP inicia com uma solução vazia e, durante a sua construção, os elementos (candidatos) disponíveis para serem adicionados à solução sendo construída são mantidos em uma lista de candidatos C ($C = [c_1, c_2, ..., c_n]$) em uma ordem decrescente de classificação definida por uma função gulosa $g(c)$, i.e., $g(c_1) \geq g(c_2) \geq ... \geq g(c_n)$ (RESENDE; RIBEIRO, 2003).

Durante uma iteração, a solução é construída candidato a candidato, com aqueles que já foram utilizados removidos da listas de candidatos. Com isso, em um dado momento da construção, C contém somente os candidatos que estão prontamente disponíveis para serem inseridos à solução parcial, mas que ainda não foram (MOUZON; YILDIRIM, 2008).

O α é um parâmetro que pertence ao intervalo $[0, 1]$ e é responsável por restringir a lista de candidatos, criando a *lista restrita de candidatos* (LRC). Existem duas estratégias para criar a LRC. Uma baseia-se na cardinalidade da LRC e a outra na qualidade dos elementos que a compõem. Sejam $G_{min} = min\{g(c)|c \in C\}$ e $G_{max} = max\{g(c)|c \in C\}$ o menor e maior custo incremental dos candidatos de acordo com a função gulosa $g(c)$, onde C é a lista de todos os possíveis candidatos a compor a solução. De acordo com Lima (2009), tem-se:

- *Estratégia de construção da LRC com base na cardinalidade*: ordena-se C em ordem decrescente de benefício segundo a função $g(c)$, de modo que para o primeiro candidato $g(c) = G_{min}$, e para o último, $g(c) = G_{max}$. A partir desse ponto, inclui-se os p primeiros elementos de C na LRC, sendo o valor de p calculado por:

$$p = 1 + \alpha(N - 1) \tag{3.1}$$

  Onde N é o total de candidatos em C.

- *Estratégia de construção da LRC com base na qualidade dos candidatos*: a LRC é construída considerando os valores $g(c)$ associados a cada candidato, de forma que o conjunto de elementos que irão compor a LRC é caracterizado por:

$$LRC = \{c \in C|g(c) \leq G_{min} + \alpha(G_{max} - G_{min})\} \tag{3.2}$$

  Os elementos de C que irão compor LRC serão apenas aqueles que o valor de $g(c)$ respeita a condição imposta por 3.2.

O α influencia o aspecto probabilístico do GRASP. Normalmente realiza-se experimentos de calibração para analisar qual valor de α apresenta melhores soluções para o problema empregado (FESTA; RESENDE, 2011; LIMA, 2009).

Antes de se incorporar um novo elemento à solução parcial, repete-se o processo de gerar a lista restrita de candidatos e escolhe-se um elemento aleatoriamente a partir desta. Uma vez que o candidato é adicionado à solução, a lista de candidatos é atualizada e os candidatos são reavaliados pela função gulosa g. O processo se repete até que se tenha uma solução completa (RESENDE; RIBEIRO, 2003).

Todas a versões da meta-heurística GRASP utilizadas neste trabalho têm a estratégia de construção da LRC de candidatos com base na cardinalidade.

O Algoritmo 4 apresenta o pseudocódigo da meta-heurística GRASP adaptado de Martí et al. (2015) e Resende e Ribeiro (2003). Os parâmetros α e $maxIts$ são passados como entrada (linha 1). Em seguida, cria-se uma variável, inicialmente vazia, para armazenar a melhor solução (linha 2). As linhas 3 - 7 realizam o processo de iterações. Cada iteração começa com um chamado à heurística construtiva (linha 4), seguida pela fase de busca local (linha 5). Após as duas fases, a melhor solução é atualizada, se a solução retornada pela busca local for melhor que a melhor solução encontrada até o momento (linha 6). O procedimento finaliza quando o critério de parada é alcançado, retornando a melhor solução dentre todas as iterações (linha 8).

Algoritmo 4 GRASP

```
1: procedimento GRASP(α, maxIts)
2:     s⁺ ← ∅
3:     para it ← 1 até maxIts faça
4:         s ← CONSTROISOLUÇÃO(α)
5:         s′ ← BUSCALOCAL(s)
6:         s⁺ ← MIN(s′, s⁺)
7:     fim para
8:     retorne s⁺
9: fim procedimento
```

O Algoritmo 5 apresenta um pseudocódigo para uma heurística de construção genérica mencionada na linha 4 do Algoritmo 4, também adaptado de Festa e Resende (2011). O pseudocódigo utiliza a estratégia de LRC baseada na cardinalidade. Os elementos que serão usados para gerar a solução são passados por parâmetro, assim como o parâmetro α que controla a gula e a aleatoriedade da solução (linha 1). A construção começa com uma solução inicialmente vazia (linha 2), e então se dá até que esteja completa (linha 8). A linha 5 seleciona os p melhores elementos da C para compor a LRC de acordo com

a equação 3.1, e então um candidato aleatório é sorteado para ser adicionado à solução sendo construída (linha 6). Após este processo, os candidatos são novamente avaliados de acordo com uma função gulosa (7). Quando se obtém uma solução completa, a heurística retorna a solução construída (linha 9).

Algoritmo 5 Heurística de construção

```
1: procedimento ConstroiSoluçao(candidatos, α)
2:     s ← ∅
3:     C ← candidatos avaliados de acordo com uma função gulosa g(c)
4:     enquanto s não for uma solução completa faça
5:         LRC ← p melhores elementos de C
6:         selecione um elemento da LRC aleatoriamente e adicione a s
7:         C ← candidatos reavaliados de acordo com uma função gulosa g(c)
8:     fim enquanto
9:     retorne s
10: fim procedimento
```

As heurísticas não garantem a geração de soluções ótimas, portanto recomenda-se que as soluções construídas passem por uma fase chamada de busca local. Esta fase representa a etapa de melhoria, pois investiga a vizinhança da solução construída até encontrar um mínimo local (RESENDE; RIBEIRO, 2003).

O processo de busca local é influenciado pela qualidade da solução construída pela heurística de construção, usualmente é importante iniciar o procedimento a partir de soluções que já possuam uma boa qualidade. Geralmente, soluções que foram construídas com um método guloso possuem qualidade melhor do que aquelas que são totalmente aleatórias. Isto significa que o procedimento de busca local consumirá possivelmente menos tempo para convergir para um ótimo local (LOURENÇO; MARTIN; STÜTZLE, 2003).

A busca local pode ser implementada utilizando duas estratégias. Na estratégia primeiro aprimorante (em inglês: *first-improving*), a solução corrente é substituída pela primeira solução aprimorante encontrada. Na estratégia melhor aprimorante (em inglês: *best-improving*), toda a vizinhança é testada e a solução corrente é substituída pela solução que apresenta a maior melhoria da função objetivo (LOURENÇO; MARTIN; STÜTZLE, 2003; RESENDE; RIBEIRO, 2003).

O Algoritmo 6, obtido de Resende e Ribeiro (2003), apresenta o pseudocódigo para o procedimento de busca local. Inicialmente, uma solução inicial e um modelo de vizinhança são passados por parâmetro (linha 1). O laço das linhas 2 - 5 realiza o processo de refinamento da solução inicial. Na linha 3, a vizinhança é examinada utilizando o critério de primeiro aprimorante ou melhor aprimorante. Na linha 4, a variável s que servirá de

solução base para a próxima iteração é atualizada. A busca para se nenhuma solução melhor é encontrada. A melhor solução é retornada na linha 6.

Algoritmo 6 Procedimento de busca local

```
1: procedimento BuscaLocal(s, N)
2:     enquanto s não é um ótimo local faça
3:         Encontre s' ∈ N(s), tal que f(s') < f(s)
4:         s ← s'
5:     fim enquanto
6:     retorne s
7: fim procedimento
```

Nas seções seguintes, as aplicações de dois métodos GRASP são apresentadas. Duas das heurísticas de Sakuraba et al. (2016) são adaptadas como fases de construção do GRASP. Também é descrito um movimento de vizinhança viável para um procedimento de busca local para WSP.

3.2.1.1 Fase construtiva

Duas heurísticas construtivas foram utilizadas separadamente na fase construtiva do GRASP. As heurísticas de Ranque e Economias descritas respectivamente nas seções 3.1.1 e 3.1.2. Esta escolha se deu porque essas heurísticas podem ser diretamente adaptadas para serem usadas como fase de construção do GRASP, incluindo o conceito de *LRC*. A LCH pode ser vista como um "*framework*" que utiliza listas imbricadas para construir uma solução. O fato de trabalhar com múltiplas listas, não permite uma utilização direta no GRASP, como é o caso das heurísticas de Ranque e Economias. As versões do GRASP incluindo as heurísticas de Ranque e Economias adaptadas como fases construtivas são referidas respectivamente como GRASP-R e GRASP-S.

O pseudocódigo para a heurística de Ranque como fase construtiva do GRASP-R é descrito no Algoritmo 7. As linhas 2 - 7 calculam o ranque das arestas de acordo com a ocorrência nos caminhos mínimos (CM). A linha 8 cria a lista C ordenando as arestas bloqueadas em ordem decrescente de ranque. Na linha 10, para cada período de reparação, os candidatos da *LRC* são escolhidos de acordo com a equação 3.1. O laço das linhas 11 - 13 sorteia as arestas bloqueadas aleatoriamente da *LRC* e as alocam para as WTs. O sorteio é realizado entre as arestas que são acessíveis. A instrução na linha 16 atualiza os caminhos mínimos quando uma aresta é totalmente reparada. Na linha 17, a aresta completamente reparada é adicionada à solução s, que é uma representação da solução e servirá como entrada para o procedimento de busca local. A linha 18 retorna a solução s

e o valor da função objetivo.

Algoritmo 7 Fase construtiva do GRASP-R

```
1: procedimento RANKING(G = (V, E), O, D, B, α)
2:     para todo i ∈ O faça
3:         Calcule os CM para os vértices usando r_ij como custo
4:         para todo j ∈ D faça
5:             Adicione 1 ao ranque de cada [i, j] ∈ B nos CM
6:         fim para
7:     fim para
8:     C ← arestas bloqueadas em ordem decrescente de ranque
9:     para t ← 1 até T faça
10:        LRC ← p melhores elementos de C
11:        para todo WT disponível faça
12:            Aloque a WT aleatoriamente para uma aresta acessível em LRC
13:        fim para
14:        se alguma [i, j] ∈ B tornou-se disponível então
15:            Atualize os CM utilizando a nova aresta disponível
16:            Adicione [i, j] à solução s
17:        fim se
18:    fim para
19:    retorne s e valor da Função Objetivo
20: fim procedimento
```

O pseudocódigo para a heurística de Economias como fase construtiva do GRASP-S é descrito no Algoritmo 8. As linhas 2 - 4 calculam as economias das arestas de acordo com a função objetivo. A linha 5 cria a lista C ordenando as arestas bloqueadas em ordem decrescente de economia. Na linha 7, para cada período de reparação, os candidatos que comporão a LRC são escolhidos de acordo com a equação 3.1. O laço das linhas 8 - 10 sorteia as arestas bloqueadas aleatoriamente da LRC e as alocam para as WTs. O sorteio é realizado entre as arestas que são acessíveis. A instrução na linha 12 atualiza os caminhos mínimos (CM) quando uma aresta é totalmente reparada. Na linha 13, a aresta completamente reparada é adicionada à solução s. Esta solução s servirá como entrada para o procedimento de busca local. A instrução da linha 15 recalcula as economias das arestas e a linha 17 recria C de acordo com os novos valores de economia. A linha 20 retorna a solução s e o valor da função objetivo.

3.2.1.2 Busca local para o WSP

As heurísticas construtivas representam uma solução s como uma lista ordenada de arestas bloqueadas. Por exemplo, considere a solução $s = [e_1, e_2, e_3, ..., e_n]$, onde o índice

Algoritmo 8 Fase construtiva do GRASP-S

```
1: procedimento SAVINGS(G = (V, E), O, D, B, α)
2:     para todo [i, j] ∈ B faça
3:         Calcule a economia se a aresta [i, j] for reparada
4:     fim para
5:     C ← arestas bloqueadas em ordem decrescente de economia
6:     para t ← 1 até T faça
7:         LRC ← p melhores elementos de C
8:         para todo WT disponível faça
9:             Aloque a WT aleatoriamente para uma aresta acessível em LRC
10:        fim para
11:        se alguma [i, j] ∈ B tornou-se disponível então
12:            Atualize os CM utilizando a nova aresta disponível
13:            Adicione [i, j] à solução s
14:            para todo [i, j] ∈ B faça
15:                Calcule a economia se a aresta [i, j] for reparada
16:            fim para
17:            C ← arestas bloqueadas em ordem decrescente de economia
18:        fim se
19:    fim para
20:    retorne s e valor da Função Objetivo
21: fim procedimento
```

de uma aresta na lista indica a ordem de reparação[1] válida que respeita as restrições de acessibilidade do problema. A busca local realiza modificações nesta lista ordenada. Posteriormente, um algoritmo de avaliação é aplicado à solução modificada.

O procedimento de busca local proposto neste trabalho utiliza uma heurística de refinamento iterativa do tipo $2-opt$ (CROES, 1958), que recebe uma solução s como parâmetro de entrada e tem o papel de melhorá-la buscando por soluções vizinhas $s^{'} \in S(s)$, onde S é uma função que enumera a vizinhança de s a partir de um movimento predefinido. O procedimento de busca local é implementado utilizando a estratégia de primeiro aprimorante. Esta decisão foi tomada pelo motivo de que avaliar cada elemento $s^{'} \in S(s)$ possui um custo razoável. Assim, espera-se que a busca local convirja para um ótimo local mais rapidamente (RESENDE; RIBEIRO, 2003).

A busca local utiliza o modelo de vizinhança descrita na Equação 3.3 e utiliza o movimento $swap(s, e_1, e_2)$, que troca a posição das arestas e_1, e_2 na solução s. Com este modelo de vizinhança, cada solução s possui até $|B|^2$ soluções vizinhas, onde B é o conjunto de

[1] Também existem outras informações: o período que a arestas começou a ser reparada, o período de finalização, as máquinas que reparam a arestas etc. Porém, a busca local modifica a ordem das arestas para gerar soluções levemente modificadas.

arestas bloqueadas.

$$S(s) = \{swap(s, e_1, e_2) \mid e_1, e_2 \in B\} \tag{3.3}$$

Diz-se que $swap(s, e_1, e_2)$ é um movimento viável, pois a modificação na ordem das arestas é verificada posteriormente pelo algoritmo de avaliação, que recalcula as rotas de reparação baseada na nova ordem das máquinas. Os testes de acessibilidade realizados pelo algoritmo de avaliação levam em consideração a ordem de cada aresta dentro da solução, mas asseguram que sejam fisicamente acessíveis antes que sejam alocadas às máquinas.

A vizinhança $S(s)$ pode ser ilustrada utilizando o grafo da Figura 4, com 1 origem (nó 1) e 1 acampamento (nó 4). A origem possui uma máquina, enquanto as arestas estão danificadas em $r = 1$. Considere que $s = \{[1,4],[1,2],[2,4],[3,4],[1,3]\}$ é uma solução para o grafo. A Figura 5 apresenta a vizinhança $S(s)$ com o movimento $swap(s, e_1, e_2)$ para todos os pares $(e_1, e_2) \in s$.

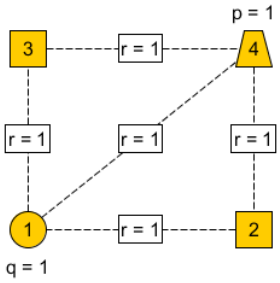

Figura 4: Exemplo de grafo $G = (V, E)$.

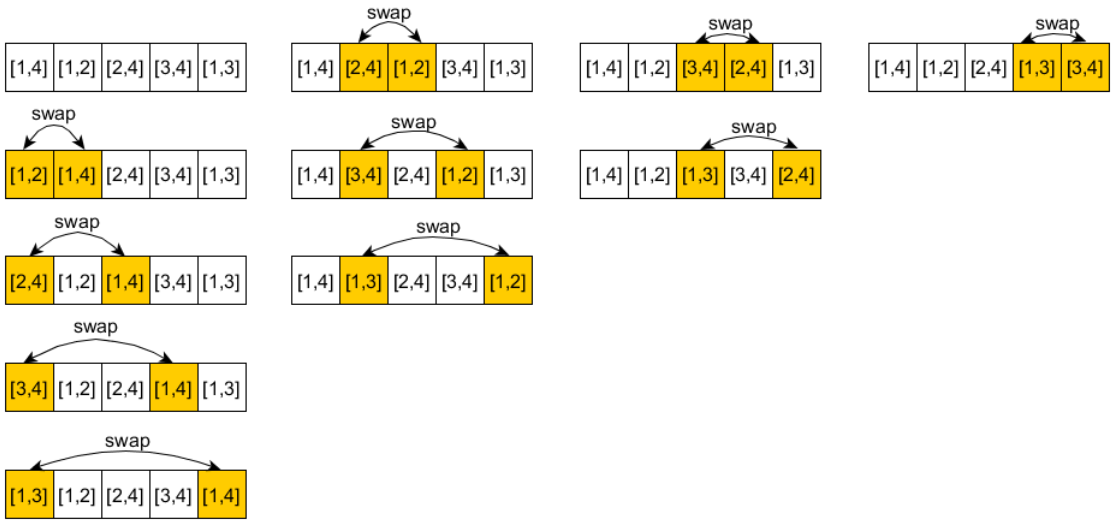

Figura 5: Exemplo de vizinhança S(s).

A Figura 6 ilustra o processo de avaliação para a solução s ={[1, 4], [1, 2], [2, 4], [3, 4], [1, 3]}. O algoritmo de avaliação escaneia s e tenta alocar a *work-troop* para as arestas a partir da cabeça da lista. A etiqueta w' indica o período em que a aresta é reparada. No primeiro período, a aresta bloqueada de menor índice [1, 4] é reparada a partir do nó 1; no segundo período, a *work-troop* utiliza [1, 4] recém reparada para retornar ao nó 1 e reparar a aresta [1, 2]; na mesma sequência, as arestas [2, 4], [3, 4], [1, 3] são reparadas no terceiro, quarto e quinto períodos. É possível notar que as arestas são reparadas na mesma ordem em que estão dispostas na solução.

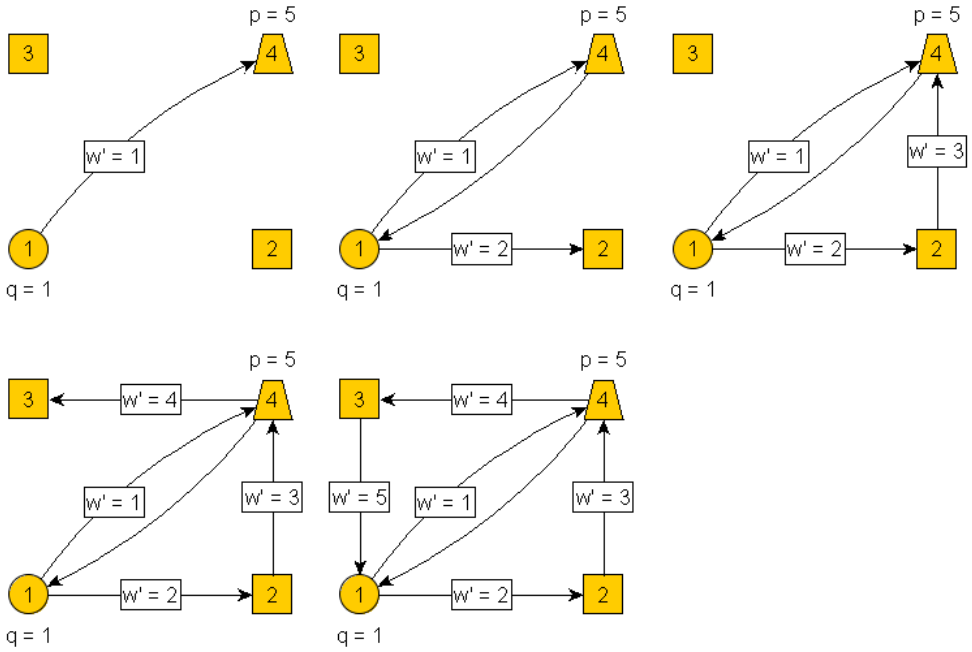

Figura 6: Solução $s = \{[1, 4], [1, 2], [2, 4], [3, 4], [1, 3]\}$.

A Figura 7 apresenta uma solução $s' = \{[2, 4], [1, 2], [1, 4], [3, 4], [1, 3]\}$. Considere que $s' = swap(s, [1, 4], [2, 4])$. A aresta [2, 4] está na cabeça da lista, mas não é acessível a partir da origem no primeiro período. O algoritmo de avaliação escaneia a solução à procura da próxima aresta acessível, alocando a *work-troop* para [1, 2] no primeiro período. Em seguida, aresta [2, 4] ainda é a cabeça da lista e pode finalmente ser reparada a partir do nó 2 no segundo período; isto é possível porque, com a reparação de [1, 2], existe agora um caminho transitável até o nó 2. O algoritmo de avaliação segue escaneando a solução, reparando as arestas [1, 4], [3, 4] e [1, 3] em sequência no terceiro, quarto e quinto períodos.

O Algoritmo 9 apresenta o pseudocódigo para o algoritmo de avaliação. Inicialmente, os dados do grafo e uma solução s são passados como parâmetro (linha 1), em seguida os caminhos mínimos (CM) são calculados utilizando as arestas transitáveis. Os caminhos

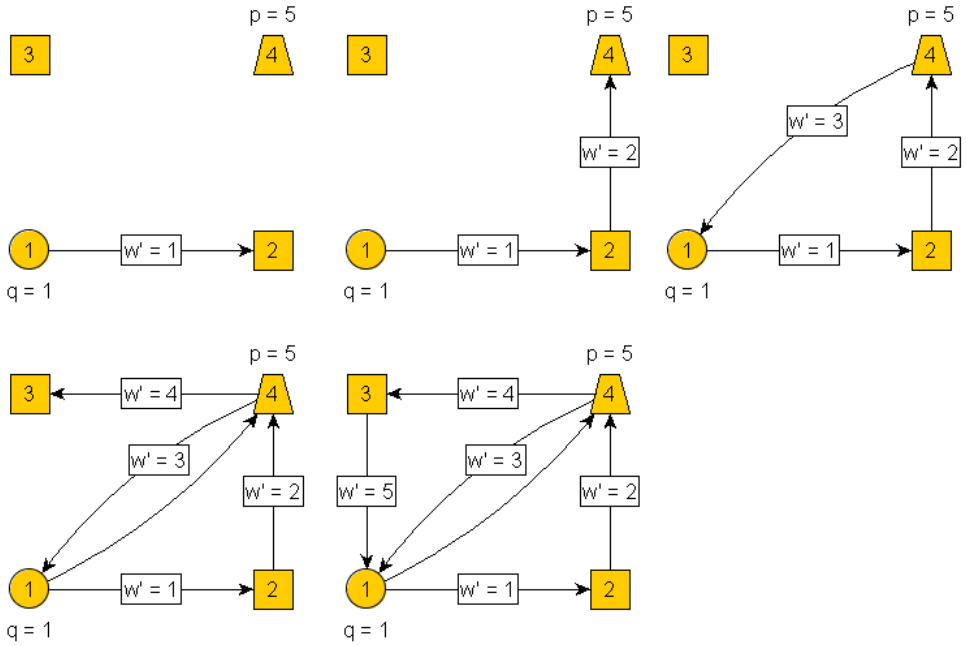

Figura 7: Solução $s' = \{[2,4],[1,2],[1,4],[3,4],[1,3]\}$.

mínimos são obtidos pelo algoritmo de Dijkstra e usados para realizar os testes de acessibilidade até as arestas. O laço das linhas 3 - 14 faz a planificação das máquinas entre todos os períodos. As linhas 4 - 10 iteram entre as máquinas que estão disponíveis, enquanto o laço das linhas 5 - 9 escaneia a lista ordenada das arestas em s. A linha 6 testa se a máquina possui acessibilidade até a aresta utilizando os caminhos mínimos calculados anteriormente. Se sim, a máquina é finalmente alocada; caso contrário, o teste é realizado para a próxima aresta na solução. A linha 12 atualiza os caminhos mínimos sempre que uma aresta se torna disponível. Por fim, a solução processada com todos os testes de acessibilidade realizados é retornada na linha 15.

O movimento $swap(s, e_1, e_2)$ pode resultar em soluções com uma aresta totalmente isolada na cabeça da lista, impossibilitando que as máquinas acessem a aresta para reparála. Porém, a busca local só finaliza depois que todas a modificações são passadas pelo algoritmo de avaliação, que corrige estas irregularidades e sempre retorna uma solução válida. Além do mais, o algoritmo de avaliação é determinístico, o que implica uma mesma solução sempre terá a mesma avaliação.

3.2.2 *Busca Local Iterada*

A Busca Local Iterada (do inglês: ILS – *Iterated Local Search*) tem como ideia geral guardar sub-sequências de ótimos locais, além de utilizar perturbações em partes da

Algoritmo 9 Algoritmo de avaliação

```
1: função AVALIA(G = (V, E), O, D, s)
2:     Calcule os CM utilizando as arestas não-bloqueadas em G = (V, E)
3:     para t ← 1 até T faça
4:         para todo WT disponível faça
5:             para todo [i, j] ∈ s faça
6:                 se a WT consegue acessar [i, j] então
7:                     Aloque a WT para reparar a [i, j] no período t
8:                 fim se
9:             fim para
10:        fim para
11:        se alguma [i, j] ∈ B tornou-se disponível então
12:            Atualize os CM utilizando a nova aresta disponível
13:        fim se
14:    fim para
15:    retorne Planificação das WTs
16: fim função
```

solução para sair de ótimos locais. Uma perturbação é um procedimento que modifica parcialmente uma solução de forma aleatória, permitindo que uma nova vizinhança seja explorada pela busca local. A ILS possui três componentes principais: a construção de uma solução viável, a perturbação e a busca local. Inicialmente, uma solução inicial é gerada e a busca local é utilizada. Em seguida, iterações sucessivas são aplicadas alternando perturbação e busca local (LOURENÇO; MARTIN; STÜTZLE, 2003). A Figura 8 ilustra este processo: partindo de um mínimo local s^*, aplica-se uma perturbação probabilística que leva s^* em s'; após refinar s' com uma busca local, encontra-se um novo ótimo local $s^{*\prime}$, que pode ser melhor que s^*. Este processo pode ser realizado diversas vezes.

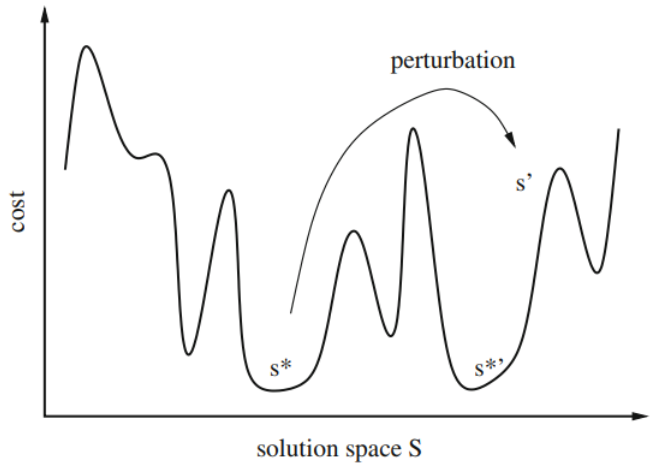

Figura 8: Representação do procedimento ILS.

Fonte: Lourenço, Martin e Stützle (2003).

A ideia da ILS é de construir uma caminhada aleatória/probabilística através do espaço de soluções existentes, que pode ou não ser finito. Cada "passo" é determinado por um procedimento de busca local que é capaz de encontrar um ótimo local na vizinhança de uma solução e uma função de pertubação que é aplicado às soluções para "saltar" para um novo ótimo local já otimizado(LOURENÇO; MARTIN; STÜTZLE, 2003).

Uma implementação da meta-heurística ILS é representada pelo Algoritmo 10, onde quatro procedimentos são especificados: (linha 2) um procedimento ou função para criar uma solução inicial, onde pode-se utilizar uma construção puramente aleatória ou uma heurística gulosa; (linhas 3 e 6) um procedimento de busca local para refinar as soluções de acordo com um ou mais modelos de vizinhança; (linha 5) uma função de perturbação, que aceita uma solução como entrada e aplica uma modificação parcial e aleatória; e (linha 7) um critério pré-definido de aceitação, que determina se uma solução $s^{*\prime}$ é aceita como um melhor ótimo local. Um exemplo de um critério de aceitação simples é verificar se o custo de $s^{*\prime}$ é o melhor até o momento (PENNA; SUBRAMANIAN; OCHI, 2013).

Algoritmo 10 *Busca Local Iterada*

```
1:  procedimento ILS
2:      s_0 ← ConstroiSolução
3:      s* ← BuscaLocal(s_0)
4:      para it ← 1, maxIts faça
5:          s' ← Perturbação(s*)
6:          s*' ← BuscaLocal(s')
7:          s* ← CriterioDeAceitação(s*, s*')
8:      fim para
9:      retorne s*
10: fim procedimento
```

O laço de repetição nas linhas 4 - 8 é executado até um número máximo de iterações, onde cada iteração faz uso de uma função de perturbação, da busca local e do critério de aceitação.

A função de perturbação na linha 5 é utilizada pela ILS em conjunto com o critério de aceitação na linha 7 para guiar as iterações pelo espaço de soluções possíveis. A força da perturbação (em inglês, *pertubation strength*) é medida pelo número ou percentual da solução que é alterada. Se esta força é muito alta, então o procedimento é semelhante a um reinício aleatório, pois grande parte da solução é modificada. Por outro lado, em perturbações sutis, o procedimento é capaz de gerar soluções que pertencem à outras vizinhanças, mas que preservam parte das características do ótimo local anterior (BESTEN; STÜTZLE; DORIGO, 2001).

O critério de aceitação na linha 7 pode utilizar diferentes estratégias para decidir se aceita ou não a solução $s^{*\prime}$ como um melhor ótimo local. Uma forma simples de implementar este critério é pela aceitação somente de ótimos locais com melhor função objetivo, como na função:

$$melhor(s^*, s^{*\prime}) = \begin{cases} s^{*\prime}, \text{ se } f(s^{*\prime}) < f(s^*) \\ s^*, \text{ caso contrário} \end{cases} \tag{3.4}$$

Onde s^* é um ótimo local, possivelmente alcançado na iteração anterior, e $s^{*\prime}$ é um outro ótimo local alcançado a partir de uma perturbação aplicada à s^*, seguida de uma busca local.

A ILS necessita de uma heurística para gerar uma solução inicial viável. Nenhuma condição particular é imposta para essa geração inicial. Desse modo, a construção da solução inicial é vista como uma caixa-preta e pode ser realizada por uma heurística específica para o problema ou por um procedimento puramente aleatório, onde a primeira é preferível sobre a segunda (LOURENÇO; MARTIN; STÜTZLE, 2003).

Como uma meta-heurística composta de 3 componentes independentes, é possível implementar a ILS para o WSP reutilizando componentes já existentes. Neste trabalho, a construção da solução inicial utiliza a heurística de Ranque, por possuir a menor complexidade computacional entre as heurísticas construtivas. Além disto, o mesmo procedimento de busca local descrito anteriormente é utilizado (ver seção 3.2.1.2). A perturbação aplica movimentos *swaps* entre arestas aleatórias na solução. A diferença do uso do *swap* na busca local é que na perturbação, os movimentos são aceitos mesmo se pioram a solução gerada. A força de perturbação é limitada a um total de 30% de modificação da solução original. Este parâmetro é uma *estimativa* de um bom valor para a força de perturbação e é inspirado em estudos de comportamento de pertubações realizados por Lourenço, Martin e Stützle (2003). O critério de aceitação consiste em substituir a solução se ela melhora a melhor solução conhecida na iteração corrente, seguindo assim o método original da ILS (ver Equação 3.4).

4 Experimentos computacionais

Neste capítulo são apresentados os resultados dos experimentos computacionais realizados com as heurísticas de Ranqueamento e de Classificação Lexicográfica (em inglês: LCH – *Lexicographic Classification Heuristic*) reproduzidas de Sakuraba et al. (2016), e as meta-heurísticas GRASP-R e ILS desenvolvidas neste trabalho. Todos os algoritmos foram implementados em Python 3.7 e os experimentos foram realizados em um *Desktop* Windows 7 64*bits* com processador Intel Core *i*7 a 3.4GHz de *clock* e 16GB de memória RAM. O objetivo dos experimentos é calibrar parâmetros, avaliar o desempenho e eficiência de cada método, compará-los, testar com vários grupos de instâncias descritos na seção 4.1.

Os resultados dos métodos são comparadas pela função objetivo (Equação 4.1) do modelo do WSP (*Work-troops Scheduling Problem*), em que p_i é a população no nó i (i é um acampamento, i.e., $i \in D$) e s_i^t é a distância entre nó i e a raiz artificial em um dado período de tempo t.

$$Z = \sum_{i \in D} \sum_{t=1}^{T} p_i s_i^t \tag{4.1}$$

Na seção 4.1 são descritos as instâncias e o formato de saída das soluções dos algoritmos. Em seguida, na seção 4.2, os experimentos de calibração são apresentados. Os resultados gerais são detalhados na seção 4.3. Por fim, a seção 4.4 apresenta gráficos TTTPlots para 9 instâncias que comparam a robustez das meta-heurísticas GRASP-R e ILS. Devido a complexidade da meta-heurística GRASP-S ser mais elevada do que a GRASP-R, optou-se por não utiliza-la nos experimentos.

4.1 Instâncias simuladas

Este trabalho utiliza quatro grupos de instâncias. Os grupos de 1 a 3 possuem 80 instâncias cada, totalizando 240. No primeiro deles, originalmente publicado em Sakuraba et al. (2016) as vias do grafo possuem largura de rua (u_{ij}) que varia entre 1 e 2. Os outros dois grupos foram derivados do primeiro e possuem respectivamente $u_{ij} = 1$ (grupo 2) e $u_{ij} = 2$ (grupo 3) para todas as vias. Estas instâncias foram originadas sobre um grafo com $|V| = 10$, $|E| = 20$, $|O| = 2$ e $|D| = 3$. Dez configurações aleatórias foram criadas para cada valor de $|B| \in \{5, 10, 15, 20\}$. Para cada configuração, duas instâncias foram geradas com $R = \sum_{[i,j]\in B} r_{ij} = 40$ e 45. Utiliza-se $10 \leq T \leq 13$ (limite máximo de períodos) para todas as instâncias simuladas e seu valor foi definido por testes preliminares realizados.

O grupo 4 de instâncias foi criado utilizando o mesmo gerador de instâncias, mas utilizando grafos maiores. Um total de 6 instâncias que variam entre 50 nós e 85 arestas até 100 nós e 180 arestas. Estas instâncias foram utilizadas, além do comparativo de qualidade de solução, para se ter uma ideia da escalabilidade dos algoritmos para grafos maiores.

4.2 Calibração dos parâmetros

Os experimentos de calibração foram realizados essencialmente com o GRASP-R para identificar o melhor valor de α. Os testes de calibração foram realizados em um conjunto de 8 instâncias com diferentes configurações, mas todas com 10 nós; 20 arestas e 4 *work-troops* distribuídas entre 2 origens. As instâncias de testes pertencem ao grupo 1. Em cada uma das instancias, executou-se 1000 amostras para cada valor de $\alpha \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Os gráficos examinam uma destas instâncias, mas para efeito de simplicidade, os gráficos mostram somente $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$.

Os gráficos na Figura 9 apresentam a distribuição do custo da solução de uma instância, uma vez que este é um problema de minimização, o valor de $\alpha = 0$ indica uma solução puramente gulosa, e $\alpha = 1$ indica uma solução puramente aleatória. Nota-se que a Figura 9 a) não possui variância e a fase de construção se comporta exatamente como a heurística de Ranqueamento gulosa. Como esperado, ao aumentar gradativamente o valor do α, o espaço de distribuição das soluções construídas também aumenta. Nota-se que $\alpha \in [0, 0.4]$ não consegue construir soluções ótimas, indicada pela linha tracejada. Somente a partir de $\alpha \geq 0.6$ que se obtém amostras de soluções ótimas.

Os gráficos na Figura 10 apresentam o resultado de se aplicar a busca local a cada uma das 1000 soluções construídas. A Figura 11 compara, em termos de diversidade e qualidade de solução, a fase de construção e a fase de construção com busca local. Os gráficos mostram os valores de α para 0 (construção puramente gulosa); 0.2; 0.4; 0.6; 0.8 e 1 (construção puramente aleatória). A linha tracejada indica a localização do ótimo global.

Durante os experimentos de calibração realizados constatou-se também que o tempo de execução do GRASP-R não é afetado pelo parâmetro α, todos os valores possuem virtualmente o mesmo tempo de processamento.

4.3 Resultados dos experimentos

Os experimentos desta seção apresentam os resultados dos grupos de instâncias. Para os grupos de 1 a 3, os resultados estão reunidos em conjuntos: 10 conjuntos numerados de 0 a 9 com 8 instâncias cada. Os resultados são apresentados assim para que o(a) leitor(a) possa ter uma visão mais sucinta da comparação entre os métodos desenvolvidos e as heurísticas da literatura. Para a comparação detalhada dos experimentos, incluindo os resultados individuais de cada uma das 240 instâncias, o(a) leitor(a) pode referir-se ao Apêndice B.

Nas Tabelas de 3 a 5, cada linha representa um conjunto de 8 instâncias e as colunas indicam: **Conj.** mostra o nome de cada conjunto de instância; os resultados ótimos são apresentados na coluna **CPLEX**; seguida das colunas das heurísticas de Ranqueamento, LCH, LCH + busca local, GRASP-R e ILS que indicam os valores obtidos por esse método para cada instância. O parâmetro α do GRASP-R foi fixado a 0.9. As meta-heurísticas tiveram um total de 20 execuções para cada instância, o critério de parada foi fixado a 10 iterações por execução. As heurísticas simples LCH de Ranqueamento foram executados apenas uma vez. Os resultados são definidos pela **Média** da função objetivo (Equação 4.1) obtida dentro do conjunto e pelo ***Gap*** (diferença percentual) do resultado ótimo. Os resultados ótimos obtidos pelos métodos estão indicados em negrito.

A Tabela 3 apresenta os resultados para o grupo de instâncias originais de Sakuraba et al. (2016), e as Tabelas 4 e 5 são de instâncias derivadas do grupo 1 e 2. A largura de rua desempenha uma restrição importante, pois impacta diretamente nas decisões de alocações que os algoritmos tomam: alocar todas as máquinas disponíveis para uma única aresta para repará-la mais rapidamente; ou distribuí-las entre diferentes arestas,

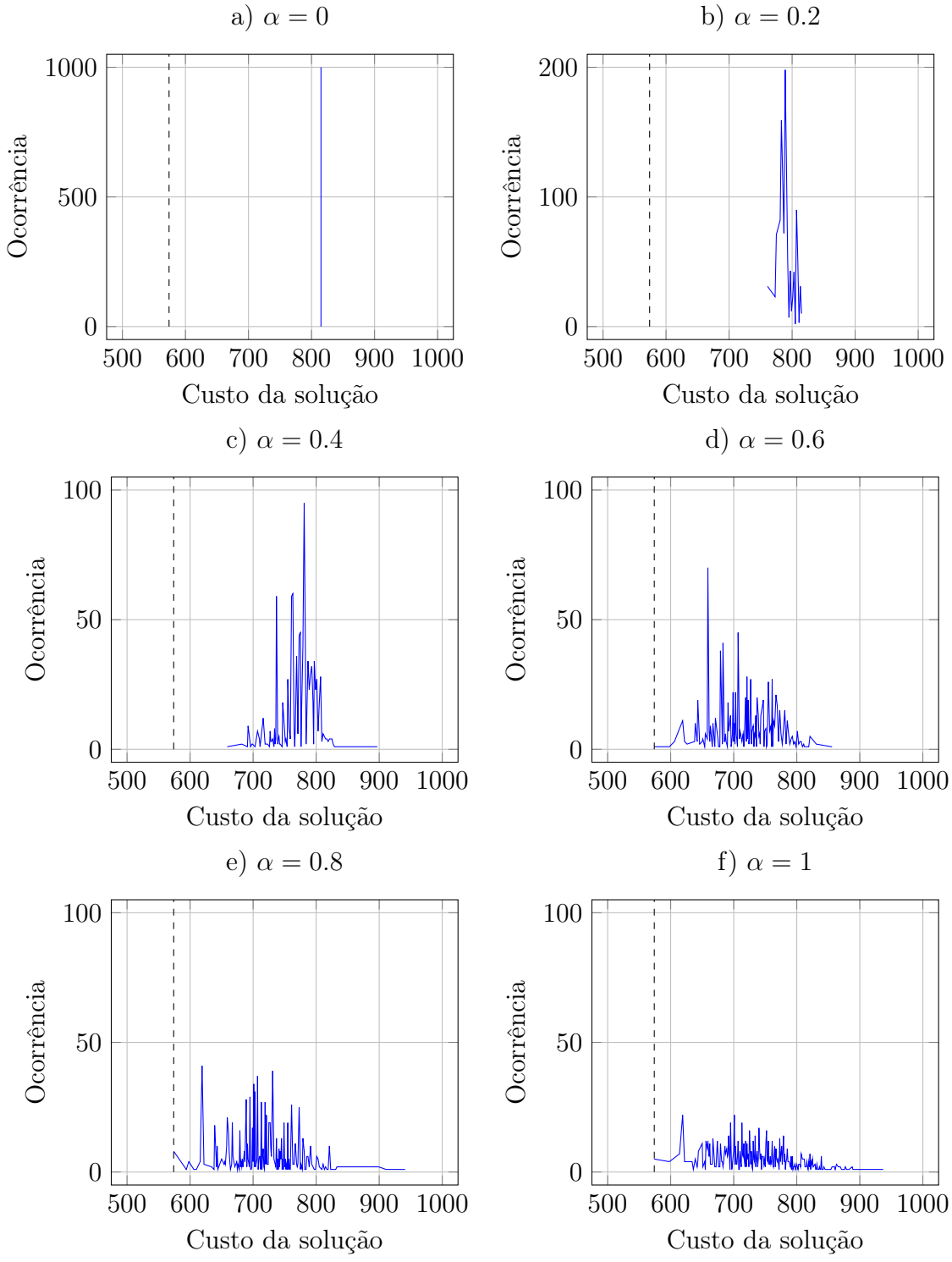

Figura 9: Distribuição da qualidade das soluções construídas variando o parâmetro α.

Fonte: Autoria própria.

necessitando de maior tempo para a reparação. Algumas vias da Tabela 3 e todas as vias da Tabela 5 permitem até 4 máquinas trabalhando ao mesmo tempo na reparação (duas em cada extremidade). Enquanto a Tabela 4 comporta até 2 máquinas simultâneas.

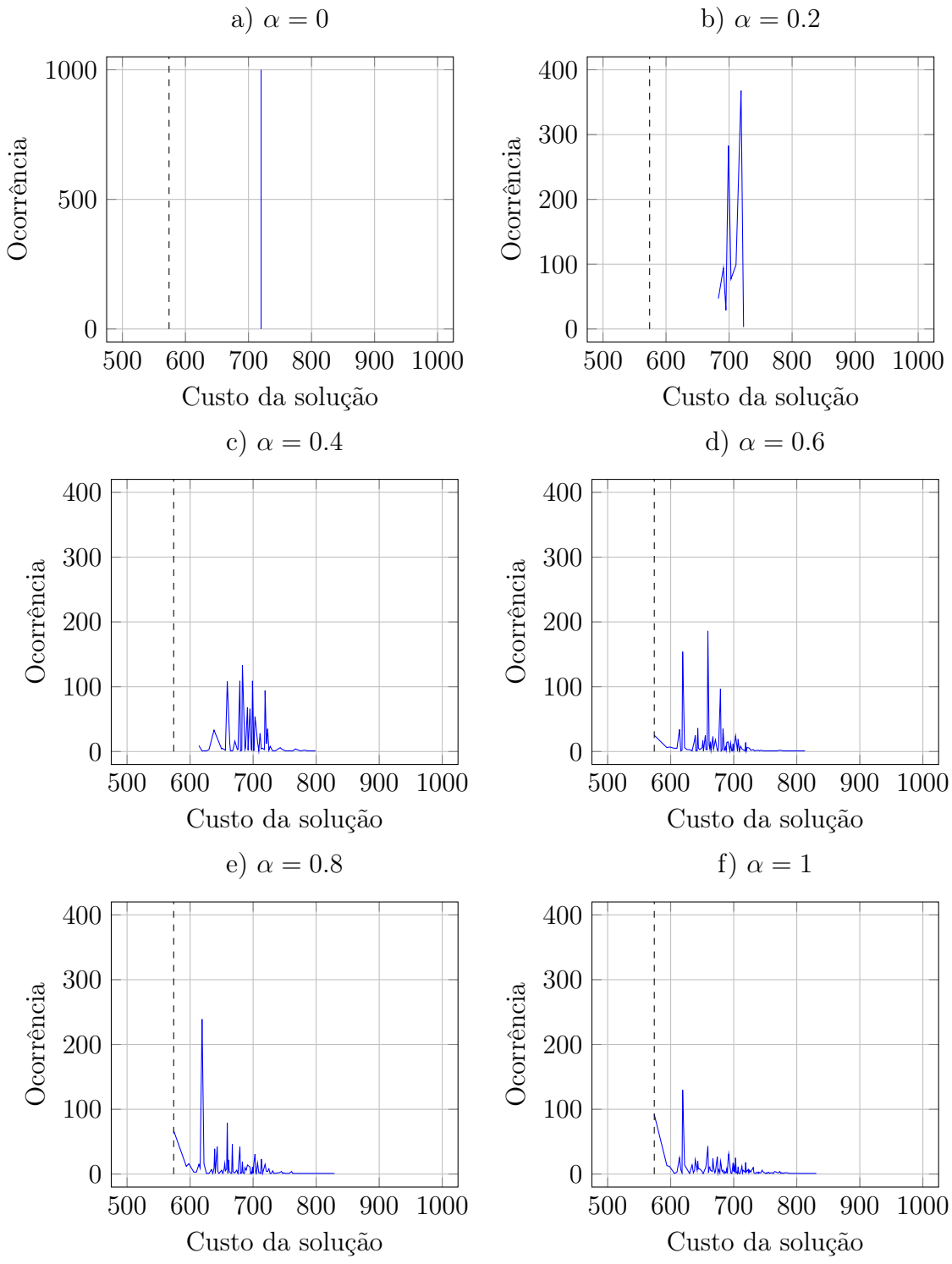

Figura 10: Distribuição da qualidade das soluções construídas variando o parâmetro α + busca local.

Fonte: Autoria própria.

Os resultados da LCH são um exemplo de como a busca local desempenha um papel importante. A adição direta desta à LCH proporciona resultados expressivamente melho-

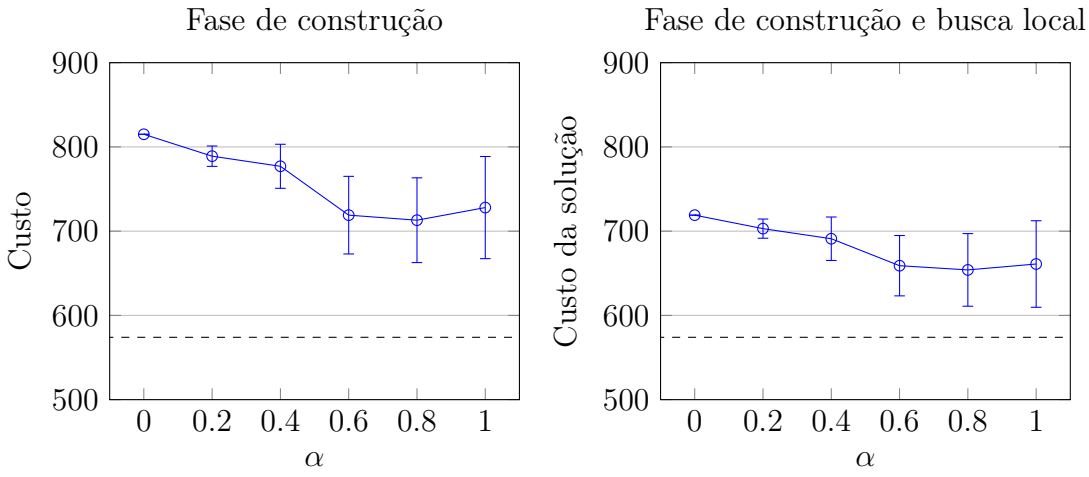

Figura 11: Média e desvio padrão das soluções construídas.

Fonte: Autoria própria.

res: O *gap* médio para os conjuntos do grupo 1 (Tabela 8) diminui de 15,41% para 0,02%; do grupo 2 (Tabela 4), de 5,6% para 0,02%; e do grupo 3 (Tabela 5), de 5,78% para 0,01%. O número de soluções ótimas por grupo que a LCH encontrou foi de 24/80, 29/80 e 45/80. A LCH + busca local atingiu 63/80, 53/80 e 69/80 soluções ótimas por grupo. Para a LCH, nenhuma instância levou mais de 0,05 segundo para executar. Na LCH + busca local, nenhuma instância levou mais que 5 segundos.

O GRASP-R apresenta os primeiros resultados de uma meta-heurística para o WSP. O valor de α foi definido para 0,9, de acordo com os experimentos de calibração realizados. Os resultados obtidos possuem um *gap máximo* de 0,03%. O GRASP-R foi capaz de encontrar soluções ótimas para 72/80, 71/80 e 78/80 das instâncias em cada um dos grupos. A média do tempo de execução do GRASP-R foi de 26,6 segundos e nenhuma instância demorou mais do que 62,2 segundos para executar.

A ILS foi utilizada como uma estratégia para "fugir" dos ótimos locais que o GRASP-R ficou estagnado, utilizando um movimento de perturbação (*swaps* aleatórios, ver Seção 3.2.1.2) seguido de uma busca local, sem ter que reconstruir uma solução a cada iteração. A ILS atingiu 76/80, 79/80 e 80/80 dos resultados ótimos para os grupos de instâncias, e os não-ótimos possuem um *gap máximo* de 0,01%. O tempo de processamento médio da ILS é de 13,1 segundos, e nenhuma instância demorou mais do 32,55 segundos para executar.

Tabela 3: Comparação dos resultados ótimos e heurísticos obtidos nos experimentos das instâncias do grupo 1.

Conj.	CPLEX	Ranqueamento		LCH		LCH com busca local		GRASP-R		ILS	
	Média	Média	Gap	Média	Gap	Média	Gap	Média	Gap	Média	Gap
0	**25538,25**	40637,63	59,12%	25634,75	0,38%	25545,38	0,03%	25539,75	0,01%	25539,75	0,01%
1	**40589,00**	94395,50	132,56%	48134,50	18,59%	40594,50	0,01%	40590,50	0,00%	40590,50	0,00%
2	**58028,25**	94395,50	62,67%	71805,25	23,74%	58033,88	0,01%	**58028,25**	0,00%	**58028,25**	0,00%
3	**58028,25**	54306,00	13,19%	54274,13	13,13%	**47976,00**	0,00%	**47976,00**	0,00%	**47976,00**	0,00%
4	**76809,25**	88176,88	14,80%	76861,75	0,07%	76814,25	0,01%	76811,25	0,00%	**76811,25**	0,00%
5	**28021,00**	43127,38	53,91%	38048,75	35,79%	28029,88	0,03%	**28021,00**	0,00%	**28021,00**	0,00%
6	**57985,50**	65539,25	13,03%	73021,50	25,93%	57991,13	0,01%	**57985,50**	0,00%	**57985,50**	0,00%
7	**40547,50**	48097,25	18,62%	44318,50	9,30%	40551,13	0,01%	40551,25	0,01%	**40547,50**	0,00%
8	**21764,50**	60612,50	178,49%	25535,00	17,32%	21770,13	0,03%	21770,13	0,03%	**21764,50**	0,00%
9	**38007,00**	45537,38	19,81%	41777,25	9,92%	38012,63	0,01%	**38007,00**	0,00%	**38007,00**	0,00%

Tabela 4: Comparação dos resultados ótimos e heurísticos obtidos nos experimentos das instâncias do grupo 2.

Conj.	CPLEX	Ranqueamento		LCH		LCH com busca local		GRASP-R		ILS	
	Média	Média	Gap	Média	Gap	Média	Gap	Média	Gap	Média	Gap
0	**40590,38**	40666,13	0,19%	40649,38	0,15%	40602,75	0,03%	40596,00	0,01%	**40590,38**	0,00%
1	**55611,25**	94539,00	70,00%	55672,75	0,11%	55621,75	0,02%	55613,50	0,00%	**55611,25**	0,00%
2	**119242,50**	155573,00	30,47%	119278,38	0,03%	119254,38	0,01%	119245,50	0,00%	**119242,50**	0,00%
3	**54243,38**	54295,13	0,10%	54255,00	0,02%	54249,00	0,01%	54249,00	0,01%	**54243,38**	0,00%
4	**76829,75**	88174,38	14,77%	76866,38	0,05%	76842,38	0,02%	76835,38	0,01%	**76829,75**	0,00%
5	**41761,00**	41872,88	0,27%	41787,50	0,06%	41772,50	0,03%	41766,63	0,01%	**41761,00**	0,00%
6	**121734,00**	129275,63	6,20%	121781,25	0,04%	121746,00	0,01%	121741,13	0,01%	121735,50	0,00%
7	**70534,25**	70598,25	0,09%	70545,88	0,02%	**70537,25**	0,00%	**70534,25**	0,00%	**70534,25**	0,00%
8	**39279,25**	73106,00	86,12%	39289,75	0,03%	39289,38	0,03%	39284,88	0,01%	**39279,25**	0,00%
9	**41765,25**	49294,88	18,03%	41787,75	0,05%	41770,88	0,01%	**41765,25**	0,00%	**41765,25**	0,00%

Tabela 5: Comparação dos resultados ótimos e heurísticos obtidos nos experimentos das instâncias do grupo 3.

Conj.	CPLEX	Ranqueamento		LCH		LCH com busca local		GRASP-R		ILS	
	Média	Média	Gap	Média	Gap	Média	Gap	Média	Gap	Média	Gap
0	**20535,75**	40592,25	97,67%	20583,00	0,23%	20541,38	0,03%	**20535,75**	0,00%	**20535,75**	0,00%
1	**35579,50**	77010,00	116,44%	35602,38	0,06%	35581,13	0,00%	35580,75	0,00%	**35579,50**	0,00%
2	**48015,25**	95618,75	99,14%	51789,13	7,86%	48025,88	0,02%	**48015,25**	0,00%	**48015,25**	0,00%
3	**47973,00**	58018,88	20,94%	47997,00	0,05%	**47973,00**	0,00%	**47973,00**	0,00%	**47973,00**	0,00%
4	**51801,00**	81910,00	58,12%	51824,88	0,05%	51803,25	0,00%	**51801,00**	0,00%	**51801,00**	0,00%
5	**18006,25**	38106,00	111,63%	18017,88	0,06%	18011,88	0,03%	**18006,25**	0,00%	**18006,25**	0,00%
6	**40500,00**	66746,25	64,81%	55494,38	37,02%	40505,63	0,01%	**40500,00**	0,00%	**40500,00**	0,00%
7	**30566,25**	43123,25	41,08%	34317,00	12,27%	30567,38	0,00%	30567,38	0,00%	**30566,25**	0,00%
8	**11760,00**	51884,00	341,19%	11763,25	0,03%	**11760,00**	0,00%	**11760,00**	0,00%	**11760,00**	0,00%
9	**18001,50**	50530,50	180,70%	18010,13	0,05%	18007,13	0,03%	**18001,50**	0,00%	**18001,50**	0,00%

A LCH se comporta melhor para os grupos de instâncias que possuem uma largura de rua fixa (grupos 2 e 3), do que o grupo de largura de rua mista (grupo 1). Uma possível explicação para isso é que a estratégia gulosa de construir os conjuntos S1 e S2 da heurística (ver Seção 3.1.3) tem dificuldades em lidar com vias de atributos u_{ij} variáveis.

Nove resultados não-ótimos do GRASP-R se deram pela necessidade de um período a mais de reparação para finalizar. Para os outros resultados, as soluções ficaram estagnadas em ótimos locais relativamente próximos ao global, mas que a busca local não foi capaz de encontrar. As 5 soluções não-ótimas da ILS são equivalentes as do GRASP-R, possivelmente indicando que alguns ótimo locais são mais difíceis de sair. Isto demonstra que talvez uma outra vizinhança precisa ser desenvolvida e incluída na busca local.

A Tabela 6 apresenta instâncias do grupo 4, de tamanhos médio e grande para analisar a performance dos algoritmos desenvolvidos em grafo maiores. A coluna **Inst.** indica o nome da instância. As informações sobre o número de nós, arestas, dano total no grafo, número de arestas bloqueadas são indicadas no nome, por exemplo: a instância G4_n50e85r100b25 possui 50 nós; 85 arestas; dano total no grafo 100 e 25 arestas bloqueadas. Em seguida, as colunas **LCH**, **LCH + busca local**, **GRASP-R** e **ILS** comparam os melhores resultados (coluna **Melhor**) e o tempo de execução (coluna **t(s)**) obtidos pelos métodos aplicados às instâncias. Assim como nos experimentos anteriores, 20 execuções foram realizadas para cada instância. As meta-heurísticas GRASP-R e ILS foram limitadas a duas iterações por execução. Como os resultados ótimos não são conhecidos, o melhor valor encontrado é indicado em negrito.

Os experimentos da Tabela 6 mostram que o GRASP-R e a ILS alcançaram os melhores resultados entre os métodos, porém com a ILS apresentando maior tempo de execução. Para a instância G4_n100e180r208b52, com 100 nós, 180 arestas e 52 arestas bloqueadas, a ILS consumiu 1050,55 segundos, 2/3 do tempo de execução do GRASP-R. Os melhores tempos de execução são da LCH, conseguindo resolver todas as instâncias em até de 0,22 segundo, mas os resultados apresentam *gap* de até 55,2% comparados com as meta-heurísticas. A LCH + busca local pode ser vista como uma alternativa mais rápida às meta-heurísticas levando, em média, metade do tempo de execução do GRASP-R e ILS. O maior *gap* entre a LCH com busca local e as meta-heurísticas foi de 0,04%, na instância G4_n60e104r112b28, indicando um ótimo local próximo aos encontrados pelo GRASP-R e ILS em um tempo menor.

Tabela 6: Comparação dos resultados heurísticos obtidos nos experimentos das instâncias do grupo 4.

Inst.	LCH		LCH + busca local		GRASP-R		ILS	
	Melhor	t(s)	Melhor	t(s)	Melhor	t(s)	Melhor	t(s)
G4_n50e85r100b25	177764	0,05	**98272**	16,69	**98272**	38,48	**98272**	54,28
G4_n60e104r112b28	106232	0,05	105352	42,71	**104892**	85,97	**104892**	100,91
G4_n70e123r160b40	636780	0,22	635660	42,71	**631820**	213,37	**631820**	276,18
G4_n80e142r152b38	315672	0,16	314512	119,52	**313472**	205,81	**313472**	229,58
G4_n90e161r180b45	121152	0,16	**120912**	191,14	**120912**	530,88	**120912**	620,57
G4_n100e180r208b52	452920	0,22	292776	457,51	**291696**	1050,55	**291696**	1589,28

4.4 Experimentos de probabilidade de tempo para alcançar um valor alvo

Esta seção apresenta os experimentos TTTPlots (*Time to target plots*) como metodologia para a comparação das meta-heurísticas GRASP-R e ILS desenvolvidas. Para realizar estes experimentos, as meta-heurísticas foram programadas para encerrar suas execuções tão logo encontrem uma solução com custo menor ou igual a um valor alvo estabelecido. O valor alvo foi definido como a média encontrada pelo GRASP-R nos experimentos apresentados no Apêndice B. As meta-heurísticas foram executadas 200 vezes para cada instância, utilizando-se sementes diferentes para o gerador de números aleatórios. Estes gráficos indicam uma aproximação da distribuição da probabilidade da variável tempo para alcançar uma solução menor ou igual ao valor alvo. Quanto mais à esquerda está o algoritmo, mais rapidamente ele converge para o valor alvo. Os gráficos apresentam os TTTPlots de 9 instâncias, 3 para cada grupo. Todas as instâncias possuem 10 nós e 20 arestas, mas diferem na configuração do grafo e no número de ruas bloqueadas.

As Figuras 12 - 14 apresentam os TTTPlots de três instâncias com grafos semelhantes, com 5 arestas bloqueadas; os prefixos G1, G2 e G3 indicam o grupo que a instância pertence. A instância G1_0n10r40B5 da Figura 12 possui custo ótimo de 582, o alvo foi definido como 594 (*gap* de 2%). As meta-heurísticas são capazes de convergir para o alvo em menos de 1,5 segundo com uma probabilidade de 100%, porém a ILS apresenta maior consistência entre as execuções. A instância G2_0n10r40B5 da Figura 13 possui custo ótimo de 654, o valor alvo foi definido como 699 (*gap* de 6,8%). O resultado das duas meta-heurísticas são semelhantes, com a ILS apresentando uma probabilidade mais alta de atingir o ótimo local mais cedo, mas a convergência com 100% de probabilidade acontece simultaneamente aos 1,6 segundo de execução. A instância G3_0n10r40B5 da Figura 14

possui custo ótimo de 570, o alvo foi definido para o valor ótimo. As meta-heurísticas também apresentam resultados semelhantes, com uma probabilidade de atingir o ótimo em até 1,1 segundo.

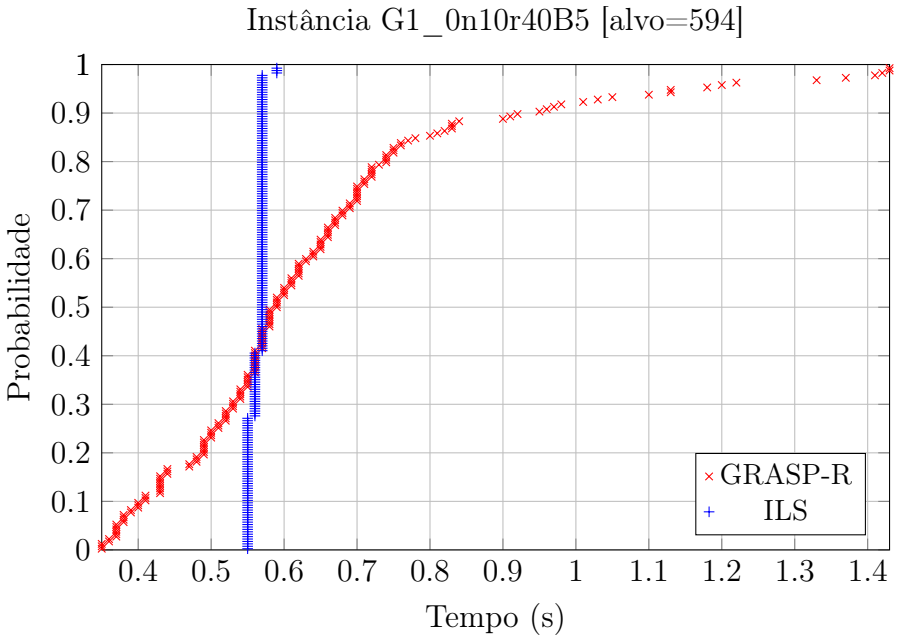

Figura 12: TTTPlot da instância G1_0n10r40B5.

As Figuras 15 - 17 apresentam TTTPlots de instâncias com 15 arestas bloqueadas. Estas instâncias estão entre as que o CPLEX levou mais tempo para finalizar a execução. A instância G1_4n10r45B15 da Figura 15 possui custo ótimo de 80660, o alvo foi definido para 80708 (*gap* de 0,06%); as duas meta-heurísticas têm uma performance equivalente, mas a ILS atinge o alvo com 100% de probabilidade cerca de 2 segundos mais rápido. A instância G2_4n10r45B15 da Figura 16 tem custo ótimo de 80718, o alvo foi definido para 80800 (*gap* de 0,1%); a ILS mostra-se mais consistente entre as execuções e leva a metade do tempo para atingir o alvo com probabilidade de 100%. A instância G1_4n10r45B15 da Figura 17 possui custo ótimo de 80640, o alvo foi definido para 80658 (*gap* de 0,02%); nesta instância, o GRASP-R mostrou-se consistentemente melhor que a ILS, consumindo cerca de 38 segundos para atingir o alvo com probabilidade de 100%, enquanto a ILS levou mais de 65 segundos.

As Figuras 18 - 20 também apresentam TTTPlots de instâncias com 15 arestas bloqueadas. Os alvos para estas instâncias foram definidos para o valor ótimo. A instância G1_6n10r45B15 da Figura 18 mostra que a ILS possui uma convergência para o alvo ótimo de 60504 em até 0,6 segundo, enquanto o GRASP-R leva até 1,1 segundo. As instâncias G2_6n10r45B15 e G3_6n10r45B15 das Figuras 19 e 20 mostram que a ILS possui

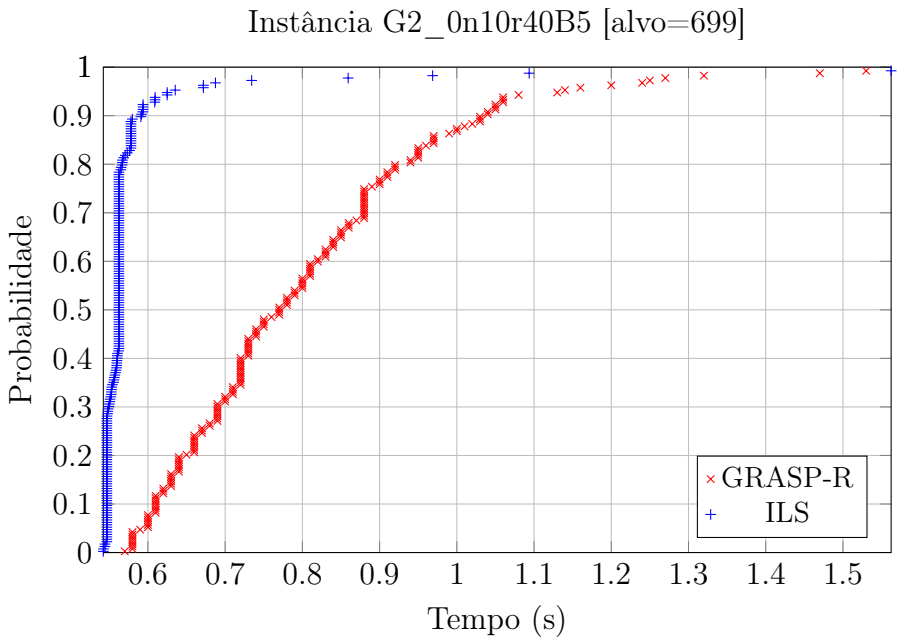

Figura 13: TTTPlot da instância G2_0n10r40B5.

variância muito pequena entre as execuções, e consegue achar o ótimo com 100% de probabilidade em menos de 1 segundo, enquanto o GRASP-R leva um tempo maior para alcançar o mesmo resultado probabilístico.

Os gráficos de TTTPlots indicam que meta-heurística ILS possui convergência para o alvo mais rápida do que o GRASP-R. Nas instâncias G1_0n10r40B5, G2_0n10r40B5 e G3_0n10r40B5, as meta-heurísticas conseguem atingir o alvo com 100% de probabilidade em menos de 2 segundos, mas com a ILS apresentando tempos melhores. Para a instância G1_4n10r45B15, o comportamentos das duas meta-heurísticas é praticamente o mesmo e as curvas se sobrepõe durante as execuções. Nas instâncias G1_6n10r45B15, G2_6n10r45B15 e G3_6n10r45B15 o alvo foi definido como o valor ótimo; é possível ver que a ILS possui variância inferior ao GRASP-R durante as execuções, sendo capaz de atingir o alvo com 100% de probabilidade em menos de 1 segundo, indicando uma robustez maior.

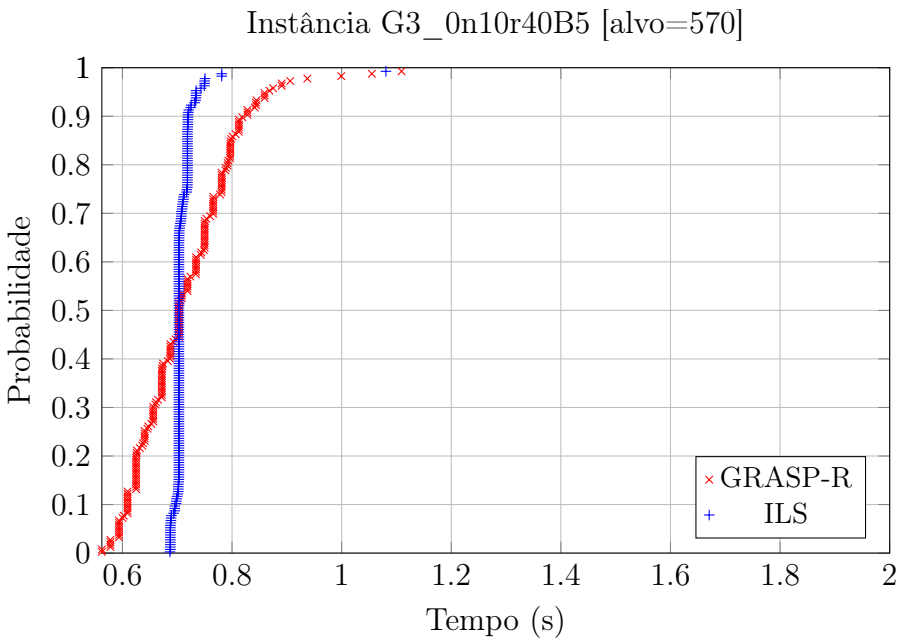

Figura 14: TTTPlot da instância G3_0n10r40B5.

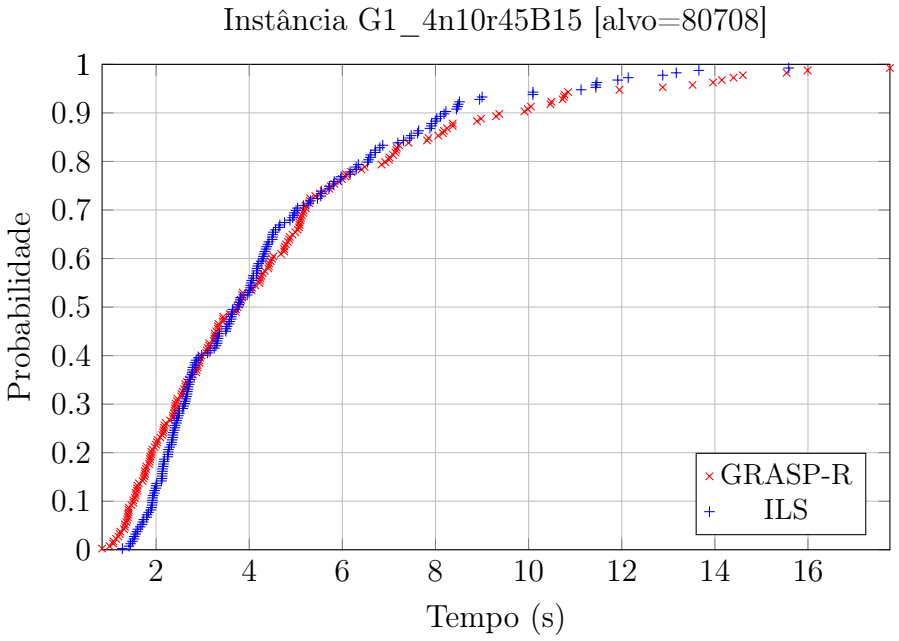

Figura 15: TTTPlot da instância G1_4n10r45B15.

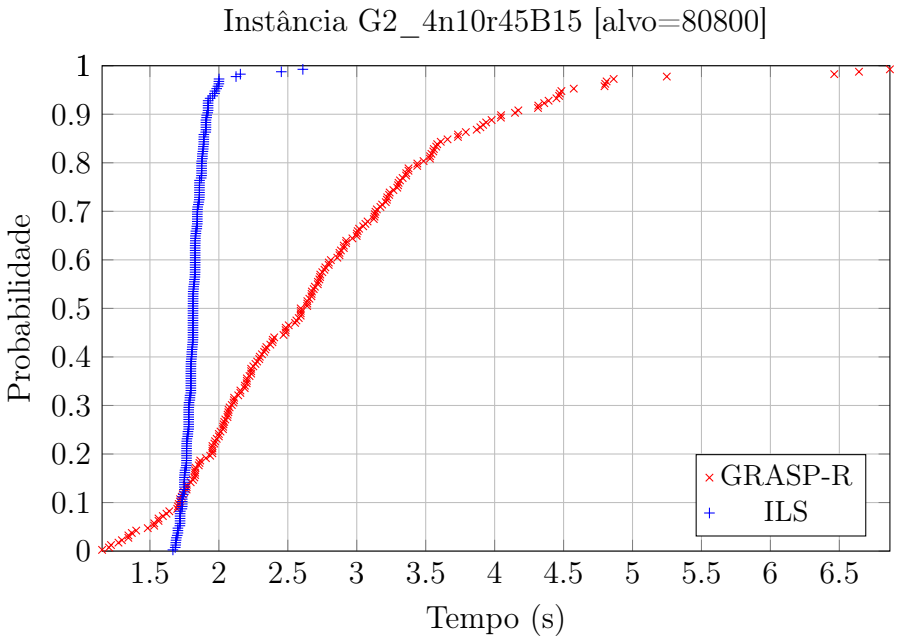

Figura 16: TTTPlot da instância G2_4n10r45B15.

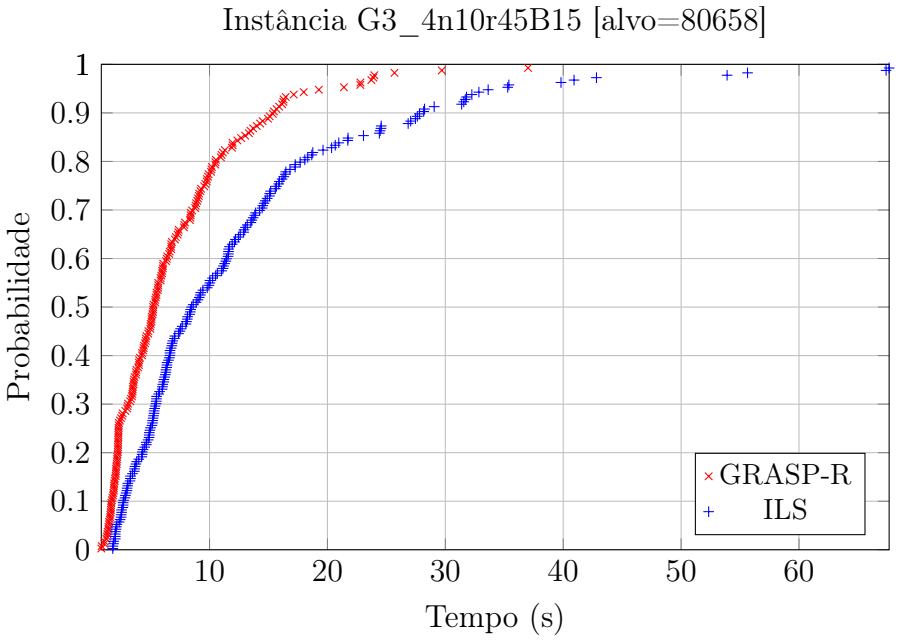

Figura 17: TTTPlot da instância G3_4n10r45B15.

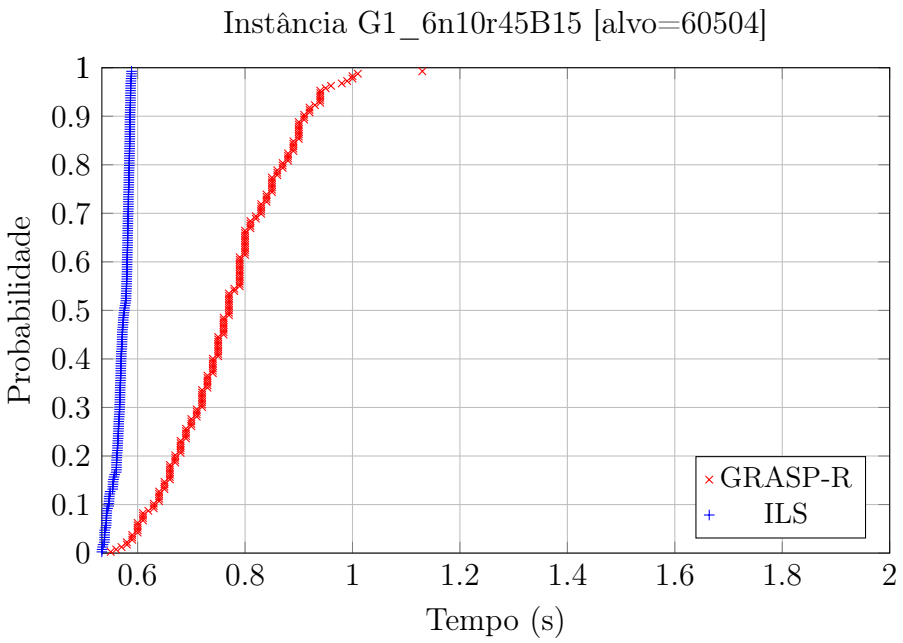

Figura 18: TTTPlot da instância G1_6n10r45B15.

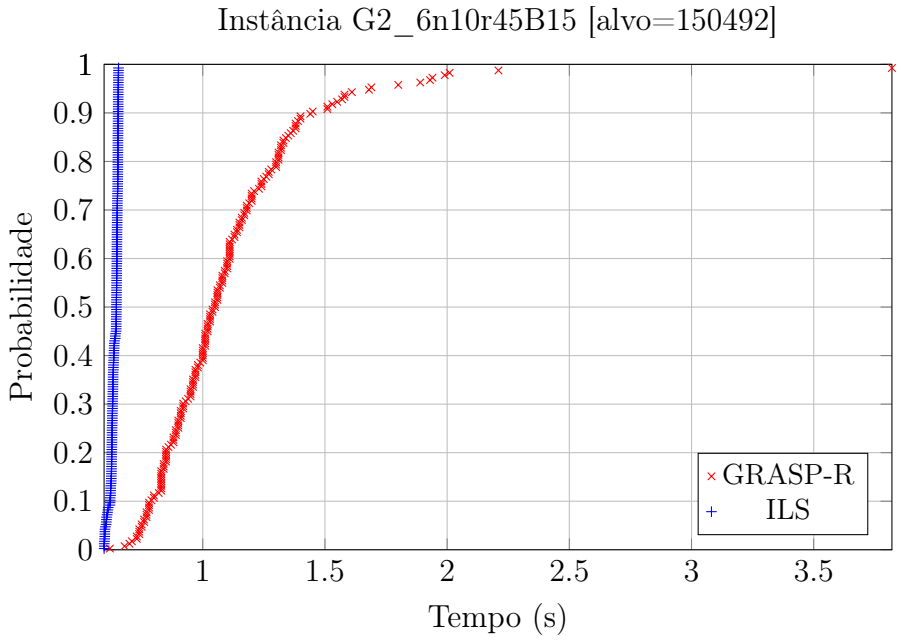

Figura 19: TTTPlot da instância G2_6n10r45B15.

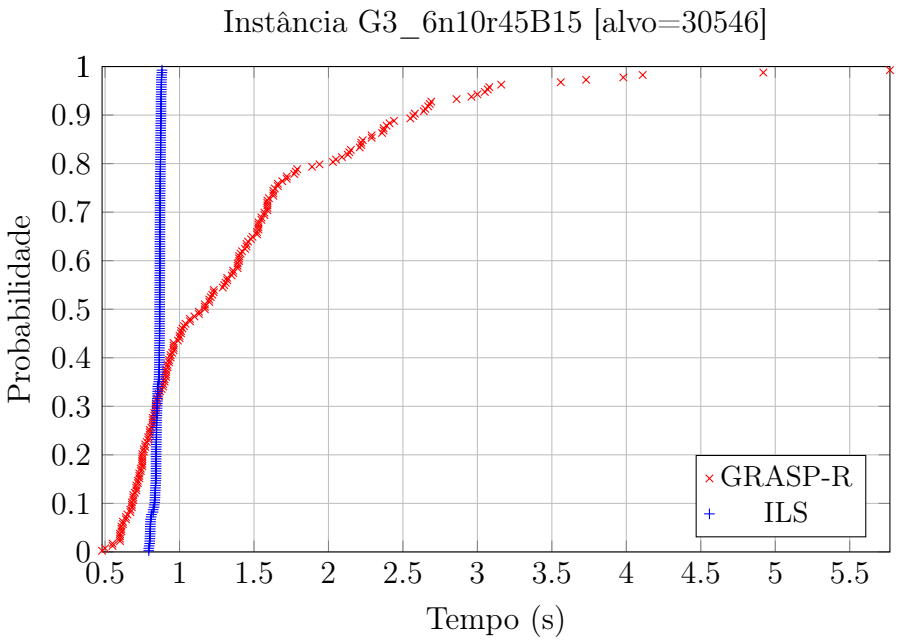

Figura 20: TTTPlot da instância G3_6n10r45B15.

5 Conclusões

Neste trabalho de mestrado, o problema de acessibilidade associado à reparação de vias após desastres de grandes proporções foi tratado. Este problema, inicialmente apresentado por Sakuraba, Santos e Prins (2015), consiste em encontrar a melhor programação para recuperar uma rede viária danificada. O problema de decisão associado é NP-difícil, portanto métodos heurísticos foram desenvolvidos. Até onde se sabe, este é o primeiro trabalho a aplicar meta-heurísticas a este problema, e isso implicou em alguns desafios.

O problema pode englobar grafos de milhares de vértices e arestas (tamanhos de cidades reais). Uma grande dificuldade para criar algoritmos é que a avaliação das soluções é realizada por um algoritmo de caminhos mínimos que adiciona pelo menos $O(nlgn)$ ao tempo de execução total, devido a característica dinâmica do problema que faz que com a rede viária mude de estado conforme as vias são reparadas. É preciso sempre garantir que exista uma matriz de caminhos mínimos com as distâncias atualizadas e em sincronia com o estado atual do grafo. Uma falha neste requisito pode gerar situações inválidas que não poderiam acontecer de acordo com as restrições do problema.

Duas meta-heurísticas de sucesso na literatura foram aplicadas: GRASP e ILS. As três heurísticas gulosas presentes na literatura foram analisadas para identificar a que melhor se encaixaria como fase construtiva para o GRASP. Apesar da Heurística de Classificação Lexicográfica ser a que apresenta os melhores resultados dentre as três, ela não é compatível com o modelo de listas de candidatos do GRASP e não pôde ser utilizada como fase de construção. A escolha da heurística de Ranque sobre a de Economias se deu pela diferença de complexidade, onde a primeira é mais aplicável à grafos maiores devido a sua baixa complexidade computacional. Uma meta-heurística GRASP utilizando a heurística de Ranque como fase construtiva foi desenvolvida e chamada de GRASP-R.

Os algoritmos utilizam um formato de soluções baseado em uma lista ordenada de arestas representando as vias bloqueadas do grafo. A ordem da lista indica a ordem de reparo das vias, com informações sobre os períodos de início e fim e as *work-troops* alocadas

para realizar o reparo. Um método de busca local foi idealizado sobre este formato. A busca local utiliza o movimento $swap(s, e_1, e_2)$ que troca as posições das arestas e_1 e e_2 na solução, implicando em uma nova ordem de reparação. Para garantir que a nova ordem respeita todas as restrições do problema, a busca local aplica um algoritmo de avaliação determinístico que recalcula as rotas de reparação das máquinas e avalia as soluções modificadas de acordo com a função objetivo.

Uma segunda meta-heurística ILS foi desenvolvida. Diferentemente da GRASP-R, a ILS não precisa recriar uma nova solução em cada iteração. A ILS foi desenvolvida utilizando componentes já existentes: a heurística de Ranque foi utilizada para gerar a solução de partida; o movimento de busca local $swap(s, e_1, e_2)$ foi usado para gerar perturbações; as perturbações alteram aleatoriamente até 30% da solução e são aceitas mesmo se piorarem o valor da função objetivo; a mesma busca local do GRASP-R foi utilizada para refinar as soluções perturbadas.

Os resultados das meta-heurísticas comparados com a Heurística de Classificação Lexicográfica se mostraram superiores em até 50% para algumas instâncias, com grande parte atingindo o valor ótimo: para a GRASP-R, 221 das 240 instâncias; a ILS atingiu 235 das 240, enquanto os resultados não ótimos estão a menos de 0,01%.

As principais contribuições deste trabalho podem ser resumidas como: (i) dois métodos meta-heurísticos que geram soluções de melhor qualidade do que as heurísticas gulosas da literatura; (ii) desenvolvimento de um modelo de vizinhança $2 - opt$ e busca local para o formato de soluções proposto; (iii) derivação de dois grupos de instâncias a partir do grupo original de Sakuraba et al. (2016), assim como a criação de um grupo de instâncias de tamanho médio.

O comparativo de tempo de GRASP-R em relação à ILS indicam que a criação de soluções a cada iteração é custoso e implica em mais tempo de processamento, embora nem sempre lide a soluções melhores. A ILS é capaz de utilizar uma única solução inicial e aplicar sucessivas perturbações + busca local e atingir resultados melhores do que o GRASP-R. O modelo de vizinhança da busca local é simples, mas é viável mesmo com o grande número de restrições que o problema possui. Uma das expectativas de trabalhos futuros é a elaboração de outros movimentos de busca local, mais sofisticados. Com esta elaboração, aconselha-se a aplicação dessas novas funções à ILS e também à meta-heurísticas baseadas em tipos parecidos de transformações, como a *Variable Neighborhood Search* (MLADENOVIĆ; HANSEN, 1997).

Existem diversas perspectivas futuras para este trabalho. Por exemplo, novas vizi-

nhanças podem ser desenvolvidas para melhorar ainda mais os resultados. Métodos emergentes como matheurísticas (*local branching*, *relax-and-fix*, etc) podem ser desenvolvidos. Além disto, a questão da escala real pode ser analisada. Os grafos reais de cidades possuem milhares de nós e de arestas. Isto necessita uma otimização das estruturas de dados e uma atenção particular à complexidade de implementação dos algoritmos. Esse foi o ponto que levou à escolha da heurística de Ranque como fase construtiva para as duas meta-heurísticas. Enfim, o problema pode ainda ser estendido para integrar outras características como o roteamento da transferência de destroços até as áreas de descarga e a integração de drones para coordenar as operações, entre outros.

Referências

AKBARI, V.; SALMAN, F. S. Multi-vehicle synchronized arc routing problem to restore post-disaster network connectivity. *European Journal of Operational Research*, v. 257, n. 2, p. 625 – 640, 2017.

AKSU, D. T.; OZDAMAR, L. A mathematical model for post-disaster road restoration: Enabling accessibility and evacuation. *Transportation Research Part E: Logistics and Transportation Review*, v. 61, p. 56 – 67, 2014.

BESTEN, M.; STÜTZLE, T.; DORIGO, M. Design of iterated local search algorithms. In: BOERS, E. J. W. (Ed.). *Applications of Evolutionary Computing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001. p. 441–451.

BINATO, S. et al. A grasp for job shop scheduling. *AT&T Labs Research Technical Report: 00.6.1*, 2000.

BRILHAM, R. Lessons from the haiti earthquake. *Nature*, Nature Publishing Group, v. 463, 2010.

COOK, S. A. The complexity of theorem-proving procedures. In: *Proceedings of the Third Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 1971. (STOC '71), p. 151–158.

CROES, G. A. A method for solving traveling-salesman problems. *Operations Research*, v. 6, n. 6, p. 791–812, 1958.

DIJKSTRA, E. W. A note on two problems in connexion with graphs. *Numerische Mathematlk*, v. 1, n. 1, 1959.

DOERNER, K. F.; GUTJAHR, W. J.; WASSENHOVE, L. V. Special issue on optimization in disaster relief. *OR Spectrum*, v. 33, n. 3, p. 445–449, 2011.

DUQUE, P. A. M.; DOLINSKAYA, I. S.; SÖRENSEN, K. Network repair crew scheduling and routing for emergency relief distribution problem. *European Journal of Operational Research*, v. 248, n. 1, p. 272 – 285, 2016.

DUQUE, P. M.; SÖRENSEN, K. A GRASP metaheuristic to improve accessibility after a disaster. *OR Spectrum*, Springer-Verlag, n. 33, 2011.

FENG, C.-M.; WANG, T.-C. Highway emergency rehabilitation scheduling in post-earthquake 72 hours. *Journal of the Eastern Asia Society for Transportation Studies*, v. 5, n. 1, 2003.

FEO, T. A.; RESENDE, M. G. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, v. 8, n. 2, p. 67 – 71, 1989.

FESTA, P.; RESENDE, M. G. C. Grasp: An annotated bibliography. In: . [S.l.: s.n.], 2002.

FESTA, P.; RESENDE, M. G. C. GRASP: basic components and enhancements. *Telecommunication Systems*, v. 46, n. 3, p. 253–271, 2011.

HIRAYAMA, N. et al. Establishment of disaster debris management based on quantitative estimation using natural hazard maps. *WIT Transactions on Ecology and the Environment*, v. 140, 2010.

HU, Y.; LIU, X.; JIANG, Y. Overviews of failure mode and reconstruction of road traffic facilities in wenchuan earthquak-stricken areas. *Procedia Environmental Sciences*, v. 12, p. 615 – 627, 2012. 2011 International Conference of Environmental Science and Engineering.

ICSMD. *International Charter "Space and Major Disasters"*. 2019. Disponível em: <`https://www.disasterscharter.org`>. Acesso em Março 14, 2019.

KARP, R. Reducibility among combinatorial problems. In: MILLER, R.; THATCHER, J. (Ed.). *Complexity of Computer Computations*. [S.l.]: Plenum Press, 1972. p. 85–103.

KIM, S. et al. Network repair crew scheduling for short-term disasters. *Applied Mathematical Modelling*, v. 64, p. 510 – 523, 2018.

LIMA, F. C. J. *Algoritmo Q-Learning como Estratégia de Exploração e/ou Explotação para as Metaheurísticas GRASP e Algoritmo Genético*. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2009.

LOURENÇO, H. R.; MARTIN, O. C.; STÜTZLE, T. Handbook of metaheuristics. In: _____. [S.l.]: Kluwer Academic Publisher, 2003. (International Series in Operations Research & Management Science), cap. Iterated Local Search, p. 321–353.

LU, G. et al. An Optimal Schedule for Urban Road Network Repair Based on the Greedy Algorithm. *PLOS ONE*, Public Library of Science, v. 11, n. 10, p. 1–15, 10 2016.

LU, X.; BENGTSSON, L.; HOLME, P. Predictability of population displacement after the 2010 haiti earthquake. *Proceedings of the National Academy of Sciences*, National Academy of Sciences, v. 109, n. 29, p. 11576–11581, 2012.

MARTÍ, R. et al. Multiobjective GRASP with Path Relinking. *European Journal of Operational Research*, v. 240, n. 1, p. 54 – 71, 2015.

MLADENOVIĆ, N.; HANSEN, P. Variable neighborhood search. *Computers & Operations Research*, v. 24, n. 11, p. 1097 – 1100, 1997.

MOUZON, G.; YILDIRIM, M. B. A framework to minimize total energy consumption and total tardiness on a single machine. *International Journal of Sustainable Engineering*, 2008.

PALLARDY, R. Haiti earthquake of 2010. *Encyclopædia Britannica*, Encyclopædia Britannica, inc., 2019.

PENNA, P. H. V.; SUBRAMANIAN, A.; OCHI, L. S. An Iterated Local Search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, v. 19, n. 2, p. 201–232, 2013.

PRAMUDITA, A.; TANIGUCHI, E.; QURESHI, A. G. Location and routing problems of debris collection operation after disasters with realistic case study. *Procedia - Social and Behavioral Sciences*, v. 125, p. 445 – 458, 2014. Eighth International Conference on City Logistics 17-19 June 2013, Bali, Indonesia.

RESENDE, M. G. C.; FESTA, P. An annotated bibliography of grasppart i: Algorithms. *AT&T Labs Research Technical Report*, 2008.

RESENDE, M. G. C.; RIBEIRO, C. C. Greedy randomized adaptive search procedures. In: _____. *Handbook of Metaheuristics*. [S.l.]: Kluwer Academic Publisher, 2003. cap. 8, p. 219–249.

SAKURABA, C.; SANTOS, A.; PRINS, C. Optimizing network accessibility: a case study on the haiti earthquake. In: *Anais do XLVII Simpósio Brasileiro de Pesquisa Operacional*. [S.l.: s.n.], 2015. p. 8.

SAKURABA, C. S.; SANTOS, A. C.; PRINS, C. Work-troop scheduling for road network accessibility after a major earthquake. *Electronic Notes in Discrete Mathematics*, v. 52, p. 317 – 324, 2016.

SAKURABA, C. S. et al. Road network emergency accessibility planning after a major earthquake. *EURO Journal on Computational Optimization*, v. 4, n. 3, p. 381–402, 2016.

SANTOS, A. C. Advances in network accessibility and reconstruction after major earthquakes. In: _____. *The Palgrave Handbook of Humanitarian Logistics and Supply Chain Management*. London: Palgrave Macmillan UK, 2018. p. 547–565.

SHANKAR, G. *Post Disaster Management, Poverty and Food*. mar. 2011. Mar., 2011. Disponível em: <`https://earthzine.org/post-disaster-management-poverty-and-food/`>. Acesso em Março 27, 2018.

ULLMAN, J. D. Np-complete scheduling problems. *Journal of Computer and System Science*, n. 10, 1975.

VODÁK, R.; BÍL, M.; KŘIVÁNKOVÁ, Z. A modified ant colony optimization algorithm to increase the speed of the road network recovery process after disasters. *International Journal of Disaster Risk Reduction*, v. 31, p. 1092 – 1106, 2018.

YÜCEL, E.; SALMAN, F.; ARSIK, I. Improving post-disaster road network accessibility by strengthening links against failures. *European Journal of Operational Research*, v. 269, n. 2, p. 406 – 422, 2018.

ZHENG, Y.-J.; CHEN, S.-Y.; LING, H.-F. Evolutionary optimization for disaster relief operations: A survey. *Applied Soft Computing*, v. 27, p. 553 – 566, 2015.

APÊNDICE A – O *Work-troops Scheduling Problem* é NP-Difícil

Este capítulo apresenta uma prova para o WSP (*Work-troops Scheduling Problem*) ser um problema NP-Difícil. Define-se o problema de decisão associado ao WSP como: dado um grafo $G = (V, E)$ e um inteiro positivo K, é possível reparar um conjunto de arestas que estejam bloqueadas de forma que a soma das distâncias da raiz artificial até os acampamentos seja menor ou igual a K?

A prova apresentada neste capítulo busca mostrar que o problema de decisão associado ao WSP é NP-Completo. Para isso, é preciso mostrar que este está em NP, e é NP-Difícil. A última implica que instâncias de qualquer problema em NP podem ser reduzidas para instâncias do WSP em tempo polinomial (KARP, 1972).

Limitações: A prova apresentada neste capítulo considera somente o primeiro período de reparação do WSP ($t = 1$). Embora seja possível generaliza-la para $t = 1, .., T$, optou-se manter esta limitação para preservar a simplicidade.

Um problema está em NP se, dada uma solução, é possível verificar se esta solução é aceita como correta ou é rejeitada em tempo polinomial (KARP, 1972; ULLMAN, 1975). O problema de decisão associado ao WSP está em NP porque é possível, dado um grafo $G = (V, E)$ e um inteiro positivo K, calcular a soma das distâncias da raiz artificial até os acampamentos utilizando um algoritmo de caminhos mínimos e verificar se esta é menor ou igual a K em tempo polinomial.

Para provar que o problema de decisão associado ao WSP é NP-Difícil, é necessário mostrar que um problema já provado ser NP-Completo pode ser reduzido em tempo polinomial para o WSP (ULLMAN, 1975). Assim, mostra-se que o problema de 3-SAT (3 - SATISFATIBILIDADE), um dos 21 problemas NP-Completos de (KARP, 1972), pode ser reduzido para o WSP por uma função que executa em tempo polinomial.

Teorema 1: Dado um grafo $G = (V, E)$ e um inteiro positivo K, é NP-Difícil decidir se

é possível reparar um conjunto de arestas bloqueadas de forma que a soma das distâncias da raiz artificial até os acampamentos seja menor ou igual a K.

A prova apresentada para o Teorema 1 utiliza os Axiomas 1 - 3:

Axioma 1: A distância até um vértice que não pode ser alcançado a partir da raiz artificial é igual a uma constante $M > 2$.

Axioma 2: A menor distância possível, se existir um caminho desbloqueado, entre a raiz artificial e um acampamento qualquer é 2.

Axioma 3: Uma *work-troop* não fica ociosa e é sempre alocada a uma aresta que esteja pronta para ser reparada.

Prova: Suponha que uma instância 3-SAT, ϕ, contém m variáveis e l cláusulas. Utiliza-se o seguinte processo para realizar a redução de ϕ para uma instância do WSP. Para cada variável em ϕ, utiliza-se o *gadget*[1] de variável da Figura 21, que consiste em uma origem i com uma *work-troop* ($q_i = 1$) conectada a um literal x e sua negação $\neg x$ por arestas bloqueadas que levam, cada uma, $r = 1$ período de tempo para ser reparada e possuem comprimento físico $d = 1$.

Para cada cláusula em ϕ, utiliza-se o *gadget* de cláusula na Figura 22. Observa-se que cada cláusula contêm exatamente três literais e, para cada literal, conecta-se a cláusula ao vértice equivalente no *gadget* de variável com arestas *transitáveis* ($r = 0$) e comprimento $d = 1$. Agora conecta-se a raiz artificial a todas as origens com arcos unidirecionais de comprimento $d = 0$ e $r = 0$. Por último, define-se $K = 2l$. Por exemplo, se $\phi = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (x \vee \neg y \vee \neg z)$ com $m = 3$, $l = 3$ e $K = 6$, o grafo resultante é o representado na Figura 23. Uma solução possível é $x = falso$, $y = verdade$ e $z = falso$ (destacado em azul).

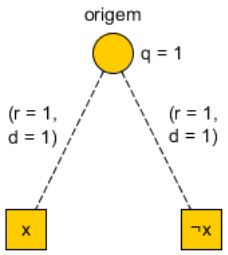

Figura 21: Exemplo do *gadget* de variável com os literais x e $\neg x$.

Fonte: Autoria própria.

[1]Um fragmento de grafo que possui certas propriedades.

No grafo da Figura 23 é possível ver que todos os caminhos entre a raiz artificial e os acampamentos estão bloqueados, assim, de acordo com o Axioma 1, a distância para cada acampamento é igual a M, então $Z = M + ... + M$ é a soma das l distâncias. Cada aresta bloqueada que conecta uma origem i a x_i e sua negação $\neg x_i$ possui atributo $r = 1$, que significa que uma *work-troop* necessita de 1 período de tempo para reparála. Para decidir se é possível que $Z = K$, é necessário determinar o conjunto de arestas bloqueadas que devem ser reparadas para abrir um caminho da raiz artificial até cada um dos acampamentos, assim a soma das distâncias deixará de ser $Z = M + ... + M$ e passará a ser $Z = 2 + ... + 2$ (Axioma 2), logo, como Z é a soma de l termos, então $Z = 2l = K$.

Figura 22: Exemplo do *gadget* de cláusula representando um acampamento com os seus respectivos literais $(x_1 \vee x_2 \vee x_3)$.

Fonte: Autoria própria.

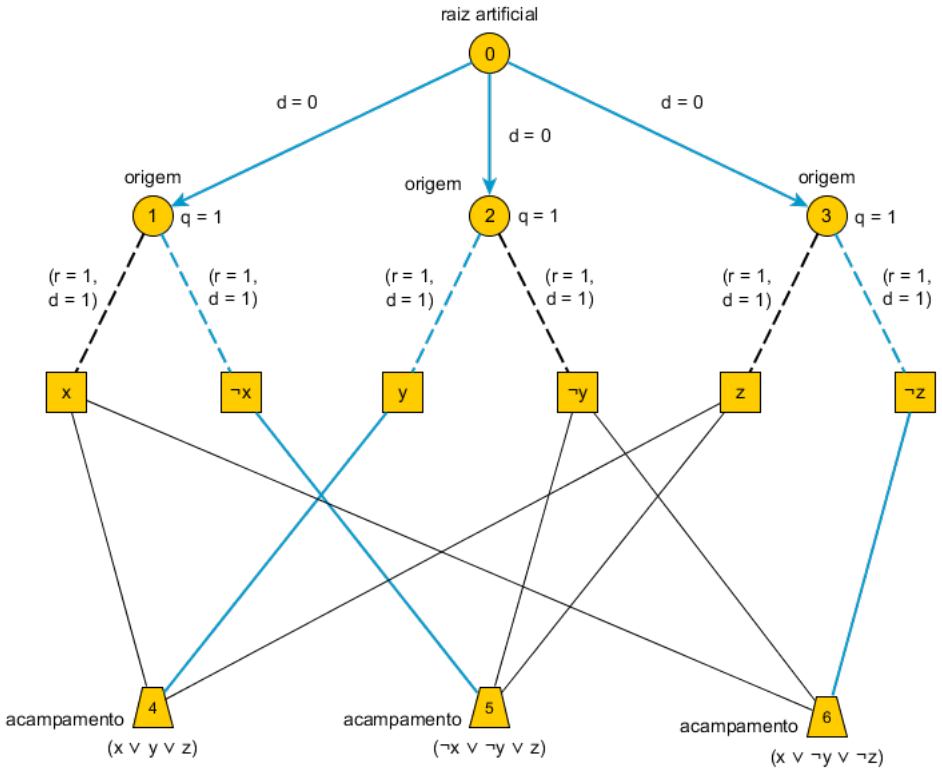

Figura 23: Exemplo de redução de 3-SAT para WSP com $\phi = (x \vee y \vee z) \wedge (\neg x \vee \neg y \vee z) \wedge (x \vee \neg y \vee \neg z)$.

Fonte: Autoria própria.

Para reparar G, a *work-troop* situada em cada origem i deve escolher exclusivamente entre desbloquear o caminho para o vértice x_i ou $\neg x_i$, pois não é possível reparar as duas arestas ao mesmo tempo. Uma *work-troop sempre* reparará uma das arestas (Axioma 3) e, caso repare o caminho até x_i, o caminho até $\neg x_i$ continua bloqueado e vice-versa. Desbloquear um caminho até x_i ou $\neg x_i$ é equivalente a atribuir valores *verdadeiro* ou *falso* à variável x_i. E também que os valores de x_i e $\neg x_i$ não podem ser iguais, ou seja, a expressão de *ou exclusivo* $x_i \oplus \neg x_i$ é sempre satisfeita. A Figura 24 apresenta um grafo para a solução $x = falso$, $y = verdadeiro$ e $z = falso$, da Figura 23, a *work-troop* situada na origem 1 reparará a aresta que leva ao vértice $\neg x$, com isso, desbloqueando o caminho para o acampamento $(\neg \mathbf{x} \vee \neg y \vee z)$. A segunda *work-troop*, na origem 2, desbloqueará o caminho até o acampamento $(x \vee \mathbf{y} \vee z)$ passando pelo vértice y; e a terceira, até $(x \vee \neg y \vee \neg \mathbf{z})$ reparando a aresta que leva a $\neg z$.

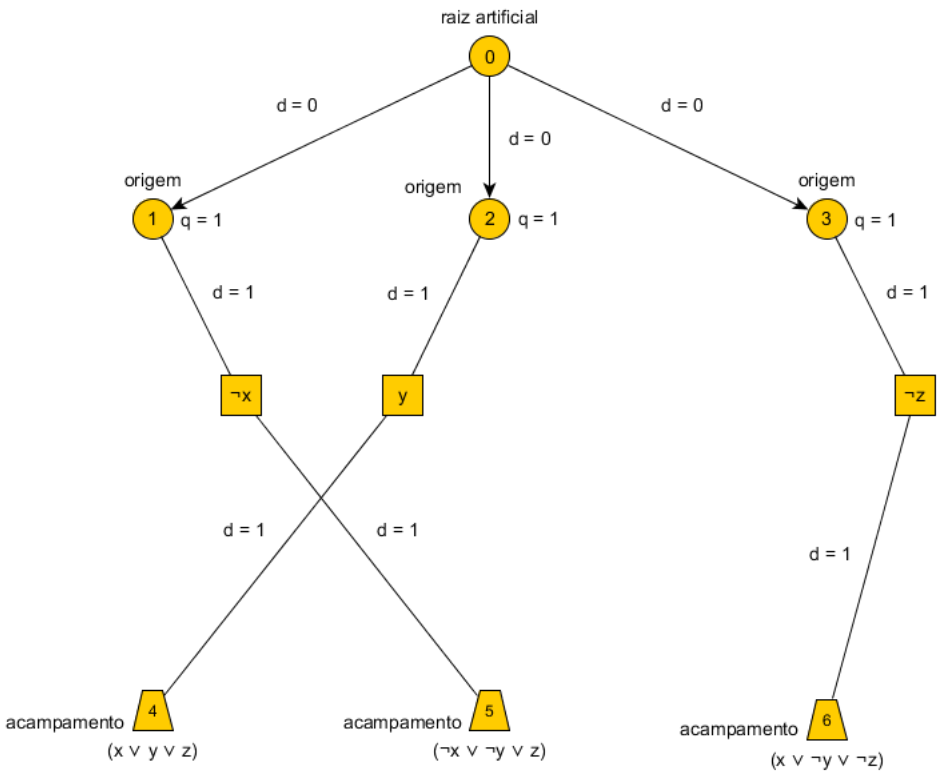

Figura 24: Árvore de caminhos mínimos para a solução $x = falso$, $y = verdade$ e $z = falso$ do grafo da Figura 23.

Fonte: Autoria própria.

$\Rightarrow$ Supõe-se agora que exista um atribuição de valores verdadeiro e falso que satisfaça ϕ. Isto significa que, para cada cláusula j em ϕ, existe pelo menos um literal em j com valor verdadeiro. Consequentemente, para cada origem i existe uma aresta que foi reparada (x_i ou $\neg x_i$) e abriu um caminho transitável até j. Se, para cada um dos acampamentos, existe

um caminho transitável a partir da raiz artificial, então, implica que existem l caminhos, cada um com custo 2 (Axioma 2), e a soma das distâncias é $Z = 2l$, então $Z = K$.

$\Leftarrow$ A segunda parte da prova mostra se $Z = 2l$ é a soma das distâncias da raiz artificial até os acampamentos, então existe uma atribuição de literais que satisfaz ϕ. Utilizando uma prova por contradição, supõe-se que $Z = 2l$, mas *não existe* uma atribuição de valores às variáveis que satisfaz ϕ. Nota-se que $Z = 2l$ pode ser escrita como:

$$\begin{aligned} Z &= 2l \\ &= 2 + ... + 2, \ \textit{com } l \textit{ termos} \end{aligned} \tag{A.1}$$

Porém, se ϕ não é satisfeita, então existe pelo menos uma cláusula que possui os três literais com valor falso. Por consequência, para algum acampamento, não existe um caminho desbloqueado a partir da raiz artificial, e a distância deixa de ser 2 e passa a ser uma constante M (Axioma 1), então Z deve ser reescrita como:

$$\begin{aligned} Z &= 2 + ... + 2 + M, \ \textit{com } l \textit{ termos} \\ &= 2(l-1) + M \end{aligned} \tag{A.2}$$

Para qualquer valor de $M > 2$, tem-se que as Equações A.1 e A.2 causam uma contradição:

$$2l = 2(l-1) + M \tag{A.3}$$

Com isso, se existe uma soma das distâncias entre a raiz artificial e os acampamentos $Z = 2l$, então tem de existir uma atribuição de valores às variáveis que satisfaz ϕ. Assim, $Z = K$ se, e somente se, existir uma atribuição às variáveis que satisfaz ϕ.

□

O processo de *redução* apresentado na prova para o Teorema 1 é polinomial. É possível reduzir qualquer instância ϕ de 3-SAT para uma instância do WSP com um algoritmo que executa em tempo polinomial, o que torna WSP pelo menos tão difícil quanto 3-SAT, ou seja, 3-SAT $\leq_p$ WSP. Se, em algum momento, for possível determinar um conjunto de arestas bloqueadas que podem ser reparadas e tornam a soma das distâncias a partir

da raiz artificial menor ou igual a um inteiro positivo K em tempo polinomial, então é possível determinar a satisfatibilidade de ϕ também em tempo polinomial (COOK, 1971; KARP, 1972).

APÊNDICE B – Resultados das instâncias simuladas

Resultados dos experimentos computacionais em três grupos de instâncias: grupo 1 (Tabela 8), instâncias originais de Sakuraba et al. (2016), que possuem largura de rua mista; grupo 2 (Tabela 9), instâncias derivadas do grupo 1, com todas as larguras de rua iguais a 1; e grupo 3 (Tabela 10), também derivadas do grupo 1, com todas as larguras de rua iguais a 2. A Tabela 7 apresenta a estrutura de um grupo de instâncias, onde |V| é o número de vértices, |E| é o número de arestas, R é o somatório dos danos das arestas bloqueadas (atributo r) e |B| é a quantidade de arestas bloqueadas.

Tabela 7: Configuração de um grupo de instâncias.

|V|	|E|	R	|B|	Nº de instâncias
10	20	40	5	10
			10	10
			15	10
			20	10
10	20	45	5	10
			10	10
			15	10
			20	10

O resultados compreendem o ótimo global calculado pelo CPLEX, e os resultados heurísticos dos métodos GRASP-R e ILS. Para cada instância, 20 execuções do GRASP-R e do ILS foram realizadas, e o melhor resultado da função objetivo e o tempo de cálculo disponibilizados nas tabelas abaixo.

Tabela 8: Comparação dos resultados ótimos e heurísticos obtidos nos experimentos das instâncias do grupo 1.

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
0n10r40B5	**582**	61,43	594	17,57	594	8,82
0n10r40B10	**501**	4,67	**501**	24,08	**501**	11,74
0n10r40B15	**40434**	10,62	**40434**	24,45	**40434**	11,99
0n10r40B20	**30456**	3,52	**30456**	25,43	**30456**	12,56
0n10r45B5	**684**	69,25	**684**	18,63	**684**	9,42
0n10r45B10	**591**	8,89	**591**	22,06	**591**	13,49
0n10r45B15	**40536**	19,55	**40536**	31,11	**40536**	15,68
0n10r45B20	**90522**	215,69	**90522**	33,72	**90522**	16,7
1n10r40B5	**502**	4,32	**502**	23,83	**502**	7,7
1n10r40B10	**552**	16,24	**552**	36,39	**552**	11,33
1n10r40B15	**702**	9,35	**702**	49,55	**702**	21,2
1n10r40B20	**120390**	10,45	**120390**	24,44	**120390**	13,53
1n10r45B5	**632**	8,3	638	22,64	638	9,25
1n10r45B10	**676**	117,25	**676**	34,84	**676**	14,08
1n10r45B15	**40794**	124,24	40800	40,62	40800	22,15
1n10r45B20	**160464**	53,79	**160464**	28,28	**160464**	16,44
2n10r40B5	**40494**	1,59	**40494**	18,64	**40494**	7,8
2n10r40B10	**522**	7,5	**522**	36,44	**522**	15,32
2n10r40B15	**40494**	12,48	**40494**	43,4	**40494**	22,31
2n10r40B20	**110400**	17,78	**110400**	25,28	**110400**	12,74
2n10r45B5	**80556**	5,62	**80556**	17,86	**80556**	9,26
2n10r45B10	**40686**	50,04	**40686**	39,22	**40686**	21,11
2n10r45B15	**40584**	19,58	**40584**	53,08	**40584**	24,94
2n10r45B20	**110490**	238,09	**110490**	28,85	**110490**	16,24
3n10r40B5	**450**	2,78	**450**	15,99	**450**	7,59
3n10r40B10	**456**	3,99	**456**	21,95	**456**	10,09
3n10r40B15	**30432**	7,13	**30432**	19,05	**30432**	10,49
3n10r40B20	**160386**	67,72	**160386**	30,42	**160386**	14,18
3n10r45B5	**540**	4,82	**540**	14,3	**540**	9,12
3n10r45B10	**546**	13,1	**546**	21,72	**546**	12,37
3n10r45B15	**30522**	147,59	**30522**	23,45	**30522**	13,24
3n10r45B20	**160476**	103,24	**160476**	41,16	**160476**	16,76

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
4n10r40B5	**80498**	3,9	**80498**	20,67	**80498**	8,47
4n10r40B10	**594**	168,18	**594**	32,95	**594**	14,19
4n10r40B15	**80526**	139,67	**80526**	52,24	**80526**	20,14
4n10r40B20	**70460**	30,75	**70460**	44,31	**70460**	17,47
4n10r45B5	**80608**	1,94	**80608**	21,14	**80608**	8,67
4n10r45B10	**40672**	503,4	**40672**	17,57	**40672**	21,67
4n10r45B15	**80660**	897,65	80676	62,61	80676	17,82
4n10r45B20	**180456**	40,12	**180456**	28	**180456**	13,67
5n10r40B5	**456**	3,74	**456**	19,39	**456**	7,79
5n10r40B10	**468**	5,99	**468**	28,23	**468**	11,41
5n10r40B15	**538**	52,24	**538**	37,36	**538**	19,84
5n10r40B20	**110424**	7,58	**110424**	28,6	**110424**	13,06
5n10r45B5	**546**	5,16	**546**	16,01	**546**	8,08
5n10r45B10	**558**	23,99	**558**	22,39	**558**	13,42
5n10r45B15	**664**	19,25	**664**	34,95	**664**	17,87
5n10r45B20	**110514**	35,63	**110514**	31,73	**110514**	18,35
6n10r40B5	**450**	2,2	**450**	17,17	**450**	7,05
6n10r40B10	**60450**	45,57	**60450**	21,91	**60450**	11,72
6n10r40B15	**60414**	121,07	**60414**	25,16	**60414**	11,81
6n10r40B20	**110400**	6,96	**110400**	28,01	**110400**	12,09
6n10r45B5	**540**	3,88	**540**	14,55	**540**	8,51
6n10r45B10	**60636**	119,31	**60636**	24,19	**60636**	13,05
6n10r45B15	**60504**	2417,1	**60504**	26,45	**60504**	12,83
6n10r45B20	**110490**	111,29	**110490**	28,85	**110490**	13,45
7n10r40B5	**528**	3,98	540	18,41	**528**	8,06
7n10r40B10	**40446**	7,07	**40446**	31,31	**40446**	19,44
7n10r40B15	**540**	19,36	549	19,72	**540**	12,4
7n10r40B20	**120390**	7,72	**120390**	22,01	**120390**	13,28
7n10r45B5	**624**	8,41	**624**	23,64	**624**	9,85
7n10r45B10	**40643**	133,65	**40643**	37,19	**40643**	17,1
7n10r45B15	**729**	41,68	738	24,37	**729**	13,68
7n10r45B20	**120480**	301,16	**120480**	28	**120480**	13,63

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
8n10r40B5	**492**	6,32	537	18,1	**492**	7,67
8n10r40B10	**450**	4,17	**450**	27,62	**450**	9,68
8n10r40B15	**474**	6,29	**474**	25,7	**474**	12,99
8n10r40B20	**70440**	209,73	**70440**	28,99	**70440**	13,59
8n10r45B5	**582**	17,61	**582**	20,09	**582**	8,05
8n10r45B10	**626**	30,58	**626**	26,52	**626**	13,02
8n10r45B15	**564**	71,28	**564**	26,47	**564**	14,01
8n10r45B20	**100488**	173,16	**100488**	27,36	**100488**	14,19
9n10r40B5	**522**	2,78	**522**	16,45	**522**	7,51
9n10r40B10	**486**	5,63	**486**	23,75	**486**	9,98
9n10r40B15	**80418**	103,62	**80418**	23,95	**80418**	12,12
9n10r40B20	**70416**	60,42	**70416**	23,8	**70416**	12,13
9n10r45B5	**612**	4,13	**612**	19,08	**612**	7,27
9n10r45B10	**588**	9,42	**588**	24,18	**588**	11,39
9n10r45B15	**80508**	7,21	**80508**	19,39	**80508**	15,41
9n10r45B20	**70506**	8,89	**70506**	23,78	**70506**	15,24

Tabela 9: Comparação dos resultados ótimos e heurísticos obtidos nos experimentos das instâncias do grupo 2.

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
0n10r40B5	**654**	18,85	699	9,14	**654**	7,3
0n10r40B10	**543**	4,26	**543**	11,47	**543**	10,8
0n10r40B15	**40476**	9,61	**40476**	11,97	**40476**	9,95
0n10r40B20	**90441**	9,92	**90441**	16,71	**90441**	11,65
0n10r45B5	**786**	53,73	**786**	9,69	**786**	7,29
0n10r45B10	**633**	5,41	**633**	14,81	**633**	12,55
0n10r45B15	**40677**	10,55	**40677**	20,55	**40677**	12,75
0n10r45B20	**150513**	3,03	**150513**	17,97	**150513**	13,52
1n10r40B5	**502**	2,5	520	8,43	**502**	6,68
1n10r40B10	**574**	13,39	**574**	19,61	**574**	8,87
1n10r40B15	**714**	90,83	**714**	33,59	**714**	19,62
1n10r40B20	**180390**	9,63	**180390**	18,63	**180390**	12,21
1n10r45B5	**638**	3,82	**638**	10,91	**638**	7,98
1n10r45B10	**712**	44,35	**712**	19,37	**712**	13,07
1n10r45B15	**40896**	51,46	**40896**	36,18	**40896**	32,55
1n10r45B20	**220464**	7,19	**220464**	22,32	**220464**	15,51
2n10r40B5	**120450**	0,95	120474	10,29	**120450**	6,73
2n10r40B10	**40470**	2,46	**40470**	24,22	**40470**	12,41
2n10r40B15	**80466**	23,67	**80466**	31,06	**80466**	11,7
2n10r40B20	**110400**	30,72	**110400**	12,45	**110400**	10,09
2n10r45B5	**200502**	1,15	**200502**	10,37	**200502**	7,93
2n10r45B10	**120624**	23,32	**120624**	18,14	**120624**	14,1
2n10r45B15	**80556**	178,31	**80556**	26,7	**80556**	18,24
2n10r45B20	**200472**	49,92	**200472**	19,81	**200472**	14,96
3n10r40B5	**450**	1,59	495	8,4	**450**	6,32
3n10r40B10	**462**	3,26	**462**	13,93	**462**	8,75
3n10r40B15	**30441**	6,83	**30441**	16,09	**30441**	10,99
3n10r40B20	**170448**	6,47	**170448**	26,12	**170448**	15,8
3n10r45B5	**540**	2,37	**540**	9,52	**540**	6,89
3n10r45B10	**558**	11,11	**558**	11,67	**558**	10,33
3n10r45B15	**60510**	11,51	**60510**	18,28	**60510**	13,46
3n10r45B20	**170538**	26,97	**170538**	28,01	**170538**	15,8

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
4n10r40B5	**80498**	2,01	80543	9,66	**80498**	6,71
4n10r40B10	**598**	2223,65	**598**	19,04	**598**	11,87
4n10r40B15	**80526**	355,62	**80526**	26,83	**80526**	18,25
4n10r40B20	**70460**	10,28	**70460**	26,59	**70460**	15,01
4n10r45B5	**80608**	2,62	**80608**	11,5	**80608**	7,43
4n10r45B10	**40750**	512177,6	**40750**	26,56	**40750**	14,67
4n10r45B15	**80718**	76373,28	**80718**	32,05	**80718**	18,62
4n10r45B20	**180480**	420,84	**180480**	21,05	**180480**	12,53
5n10r40B5	**462**	1,45	507	7,81	**462**	6,37
5n10r40B10	**468**	8,19	**468**	15,53	**468**	9,35
5n10r40B15	**538**	39,27	**538**	21,9	**538**	15,22
5n10r40B20	**150384**	4,26	**150384**	15,27	**150384**	11,19
5n10r45B5	**558**	1,62	**558**	8,65	**558**	7,5
5n10r45B10	**558**	6,65	**558**	13,8	**558**	11,77
5n10r45B15	**664**	904,65	**664**	18,4	**664**	20,02
5n10r45B20	**180456**	314,2	**180456**	16,9	**180456**	14,16
6n10r40B5	**450**	1,53	495	8,29	**450**	6,41
6n10r40B10	**120534**	257,51	**120534**	14,92	**120534**	9,91
6n10r40B15	**120414**	90,14	**120414**	34,62	**120414**	15,55
6n10r40B20	**200364**	6,8	**200364**	18,53	**200364**	11,59
6n10r45B5	**540**	2,28	**540**	8,44	**540**	6,82
6n10r45B10	**180624**	13048,41	180636	14,24	180636	10,13
6n10r45B15	**150492**	77,39	**150492**	21,27	**150492**	11,39
6n10r45B20	**200454**	52,62	**200454**	19,11	**200454**	12,64
7n10r40B5	**600**	8,88	**600**	11,16	**600**	6,79
7n10r40B10	**40454**	3,57	**40454**	25,25	**40454**	12,48
7n10r40B15	**540**	80,45	**540**	18,69	**540**	14
7n10r40B20	**210348**	12,15	**210348**	13,48	**210348**	9,9
7n10r45B5	**690**	2,54	**690**	11,57	**690**	8,2
7n10r45B10	**70526**	5,04	**70526**	30,59	**70526**	15,28
7n10r45B15	**30678**	4574,48	**30678**	16,79	**30678**	14,15
7n10r45B20	**210438**	148,92	**210438**	16,74	**210438**	12,6

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
8n10r40B5	**576**	3,54	621	8,09	**576**	6,88
8n10r40B10	**450**	3,49	**450**	14,57	**450**	8,93
8n10r40B15	**40446**	86,57	**40446**	18,11	**40446**	10,47
8n10r40B20	**70440**	98,28	**70440**	15,97	**70440**	13,31
8n10r45B5	**666**	5,55	**666**	9,69	**666**	7,39
8n10r45B10	**626**	20	**626**	14,73	**626**	10,32
8n10r45B15	**40536**	105,55	**40536**	21,1	**40536**	14,26
8n10r45B20	**160494**	5,23	**160494**	18,92	**160494**	14,72
9n10r40B5	**522**	6,02	**522**	9,27	**522**	7,87
9n10r40B10	**528**	5,52	**528**	13,8	**528**	12,07
9n10r40B15	**80418**	14,12	**80418**	11,84	**80418**	13,25
9n10r40B20	**70416**	10,67	**70416**	13,6	**70416**	13,08
9n10r45B5	**612**	2,73	**612**	10,53	**612**	6,89
9n10r45B10	**630**	5,86	**630**	13,63	**630**	13,07
9n10r45B15	**80508**	3	**80508**	9,9	**80508**	13,6
9n10r45B20	**100488**	234,33	**100488**	11,41	**100488**	15

Tabela 10: Comparação dos resultados ótimos e heurísticos obtidos nos experimentos das instâncias do grupo 3.

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
0n10r40B5	**570**	61,43	**570**	9,7	**570**	10,3
0n10r40B10	**501**	4,67	**501**	12,34	**501**	13,46
0n10r40B15	**450**	10,62	**450**	11,15	**450**	15,29
0n10r40B20	**30456**	3,52	**30456**	15,91	**30456**	10,22
0n10r45B5	**672**	69,25	**672**	9,4	**672**	10,3
0n10r45B10	**591**	8,89	**591**	13,35	**591**	15,9
0n10r45B15	**40524**	19,55	**40524**	14,06	**40524**	21,05
0n10r45B20	**90522**	215,69	**90522**	19,51	**90522**	13,72
1n10r40B5	**482**	4,32	**482**	9,29	**482**	9,6
1n10r40B10	**532**	16,24	**532**	15,94	**532**	14,92
1n10r40B15	**702**	9,35	712	19,73	**702**	26,78
1n10r40B20	**120390**	10,45	**120390**	14,88	**120390**	12
1n10r45B5	**598**	8,3	**598**	11,29	**598**	9,45
1n10r45B10	**664**	117,25	**664**	17,25	**664**	15,76
1n10r45B15	**40788**	124,24	**40788**	26,1	**40788**	31,27
1n10r45B20	**120480**	53,79	**120480**	18,17	**120480**	13,41
2n10r40B5	**40458**	1,59	**40458**	10,27	**40458**	9,26
2n10r40B10	**498**	7,5	**498**	15,75	**498**	18,84
2n10r40B15	**40478**	12,48	**40478**	24,77	**40478**	19,22
2n10r40B20	**70416**	17,78	**70416**	15,4	**70416**	13,05
2n10r45B5	**80532**	5,62	**80532**	9,99	**80532**	9,73
2n10r45B10	**40662**	50,04	**40662**	21,69	**40662**	24,1
2n10r45B15	**40572**	19,58	**40572**	30	**40572**	18,99
2n10r45B20	**70506**	238,09	**70506**	17,81	**70506**	13,61
3n10r40B5	**450**	2,78	**450**	8,16	**450**	7,17
3n10r40B10	**456**	3,99	**456**	10,7	**456**	9,79
3n10r40B15	**30432**	7,13	**30432**	11,55	**30432**	10,02
3n10r40B20	**160374**	67,72	**160374**	18,06	**160374**	15,72
3n10r45B5	**540**	4,82	**540**	8,64	**540**	7,56
3n10r45B10	**546**	13,1	**546**	12,77	**546**	11,46
3n10r45B15	**30522**	147,59	**30522**	13,17	**30522**	11,49
3n10r45B20	**160464**	103,24	**160464**	18,95	**160464**	17,96

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
4n10r40B5	**40474**	3,9	**40474**	9,22	**40474**	8,8
4n10r40B10	**554**	168,18	**554**	16,1	**554**	15,49
4n10r40B15	**80482**	139,67	**80482**	29,61	**80482**	21,71
4n10r40B20	**70460**	30,75	**70460**	19,02	**70460**	14,44
4n10r45B5	**40564**	1,94	**40564**	9,9	**40564**	9,21
4n10r45B10	**746**	503,4	**746**	18,67	**746**	18,48
4n10r45B15	**80640**	897,65	**80640**	39,48	**80640**	23,97
4n10r45B20	**100488**	40,12	**100488**	18,22	**100488**	14,4
5n10r40B5	**456**	3,74	**456**	8,62	**456**	8,41
5n10r40B10	**459**	5,99	**459**	13,02	**459**	12,15
5n10r40B15	**514**	52,24	**514**	19,59	**514**	20,26
5n10r40B20	**70416**	7,58	**70416**	14,3	**70416**	12,1
5n10r45B5	**546**	5,16	**546**	9,18	**546**	8,51
5n10r45B10	**549**	23,99	**549**	13,53	**549**	13,62
5n10r45B15	**604**	19,25	**604**	23,54	**604**	26,1
5n10r45B20	**70506**	35,63	**70506**	16,59	**70506**	13,87
6n10r40B5	**450**	2,2	**450**	8,32	**450**	7,21
6n10r40B10	**60450**	45,57	**60450**	14,15	**60450**	12,82
6n10r40B15	**30456**	121,07	**30456**	16,07	**30456**	12,22
6n10r40B20	**70416**	6,96	**70416**	13,32	**70416**	12,2
6n10r45B5	**540**	3,88	**540**	8,19	**540**	7,67
6n10r45B10	**60636**	119,31	**60636**	15,01	**60636**	14,91
6n10r45B15	**30546**	2417,1	**30546**	14,71	**30546**	15,76
6n10r45B20	**70506**	111,29	**70506**	16,67	**70506**	13,82
7n10r40B5	**528**	3,98	**528**	9,9	**528**	10,09
7n10r40B10	**569**	7,07	**569**	16,89	**569**	15,18
7n10r40B15	**540**	19,36	549	12,3	**540**	11,06
7n10r40B20	**120390**	7,72	**120390**	13,79	**120390**	12,2
7n10r45B5	**624**	8,41	**624**	10,46	**624**	9,46
7n10r45B10	**679**	133,65	**679**	18,16	**679**	18,4
7n10r45B15	**720**	41,68	**720**	16,03	**720**	13,11
7n10r45B20	**120480**	301,16	**120480**	15,51	**120480**	14,19

Inst.	CPLEX		GRASP-R		ILS	
	Obj.	Tempo (s)	Obj.	Tempo (s)	Obj.	Tempo (s)
8n10r40B5	**492**	6,32	**492**	8,12	**492**	8,69
8n10r40B10	**450**	4,17	**450**	13,35	**450**	13,99
8n10r40B15	**462**	6,29	**462**	14,14	**462**	17,38
8n10r40B20	**30456**	209,73	**30456**	14,9	**30456**	11,01
8n10r45B5	**582**	17,61	**582**	9,1	**582**	9,64
8n10r45B10	**582**	30,58	**582**	14,68	**582**	15,51
8n10r45B15	**552**	71,28	**552**	16,57	**552**	17,3
8n10r45B20	**60504**	173,16	**60504**	15,64	**60504**	13,68
9n10r40B5	**486**	2,78	**486**	8,71	**486**	8,39
9n10r40B10	**474**	5,63	**474**	12,64	**474**	14,55
9n10r40B15	**40434**	103,62	**40434**	15,58	**40434**	10,01
9n10r40B20	**30432**	60,42	**30432**	14,68	**30432**	13,3
9n10r45B5	**576**	4,13	**576**	9,23	**576**	8,75
9n10r45B10	**564**	9,42	**564**	15,13	**564**	15,95
9n10r45B15	**40524**	7,21	**40524**	14,38	**40524**	12,77
9n10r45B20	**30522**	8,89	**30522**	18,53	**30522**	12,38