



Universidade Estadual de Campinas
Instituto de Computação



Carlos Victor Dantas Araújo

Formulações e Heurísticas para o Problema de Máximo Atendimento em Roteamento Multicast com Restrições de QoS

CAMPINAS
2021

Carlos Victor Dantas Araújo

**Formulações e Heurísticas para o Problema de Máximo
Atendimento em Roteamento Multicast com Restrições de QoS**

Dissertação apresentada ao Instituto de
Computação da Universidade Estadual de
Campinas como parte dos requisitos para a
obtenção do título de Mestre em Ciência da
Computação.

Orientador: Prof. Dr. Fábio Luiz Usberti
Coorientador: Prof. Dr. Cid Carvalho de Souza

Este exemplar corresponde à versão da
Dissertação entregue à banca antes da
defesa.

CAMPINAS
2021

Na versão final esta página será substituída pela ficha catalográfica.

De acordo com o padrão da CCPG: “Quando se tratar de Teses e Dissertações financiadas por agências de fomento, os beneficiados deverão fazer referência ao apoio recebido e inserir esta informação na ficha catalográfica, além do nome da agência, o número do processo pelo qual recebeu o auxílio.”

e

“caso a tese de doutorado seja feita em Cotutela, será necessário informar na ficha catalográfica o fato, a Universidade conveniente, o país e o nome do orientador.”

Na versão final, esta página será substituída por outra informando a composição da banca e que a ata de defesa está arquivada pela Unicamp.

Resumo

Este trabalho aborda o Problema de Máximo Atendimento em Roteamento Multicast com restrições de Qualidade de Serviço (com a sigla em inglês MS-MRP-QoS) aplicado no contexto de redes veiculares ad-hoc, onde é fornecida uma rede veículos para qual deseje-se enviar dados partindo de um nó raiz com destino à um conjunto de nós terminais. A utilização de todos os nós não é obrigatória e cada conexão entre o nó raiz e um nó terminal deve satisfazer a qualidade de serviço conforme limites estabelecidos para cada métrica. O objetivo é maximizar o número de terminais atendidos em acordo com as métricas de qualidade de serviço da rede.

Nesta dissertação, foram desenvolvidas formulações de programação inteira mista e quatro relaxações lagrangianas, visando obter limitantes primais e duais de boa qualidade. Também foi desenvolvido um conjunto de métodos heurísticos, dentre eles uma busca local em arborescências, aplicada durante as resoluções das relaxações lagrangianas, e três meta-heurísticas BRKGA com variações no método de decodificação, um deles incorporando informações dos multiplicadores de Lagrange. As metodologias propostas foram submetidas a experimentos computacionais com um conjunto de instâncias geradas com características de redes veiculares ad-hoc. Análises estatísticas foram realizadas para comparar os desempenhos entre as metodologias. Os resultados destacam a eficácia da formulação mista em obter limitantes primais e duais de alta qualidade para todas as instâncias do conjunto. Já a formulação inteira, relaxações lagrangianas e BRKGA obtiveram resultados de boa qualidade e estatisticamente semelhantes.

Abstract

Lista de Figuras

1.1	Shorter figure caption	12
1.2	Métodos usuais para transmissão de dados em redes	13
1.3	Exemplo de Instância do MS-MRP-QoS	14
4.1	Exemplo de aplicação da busca local	49
4.2	Exemplo de representação gráfica do teste Nemenyi.	51
4.3	Parte do mapa da cidade de Washington no <i>Open Street Maps</i>	53
4.4	Parte do mapa ampliado da cidade de Washington no SUMO	53
4.5	Fotografia dos enlaces da rede.	54
4.6	Fluxograma do processo de geração das instâncias.	55
5.1	Testes de Iman-Davenport e Nemenyi para as diferentes RLs	72
5.2	Testes de Imand-Davenport e Nemenyi para os decodificadores das versões do BRKGA.	75
5.3	Testes de Iman-Davenport e Nemenyi para os Limitantes Inferiores.	77
5.4	Testes de Iman-Davenport e Nemenyi para os Limitantes Superiores.	79

Lista de Tabelas

5.1	Variáveis removidas pelos pré-processamentos.	57
5.2	Resultados do modelo DMFM-MRP $_{\Lambda, \Xi}$	59
5.3	Resultados do modelo DMFM-MRP $_{1,1}$	60
5.4	Resultados do modelo DMFM-MRP $_{1,1}$ sem a utilização de pré-processamentos.	62
5.5	Resultados modelo AB-MRP $_{1,1}$	63
5.6	Contagem de soluções dos modelos.	64
5.7	Resultados da Relaxação RL-SP e suas variações.	66
5.8	Resultados da relaxação RL-SPRC $_{\lambda}$ e suas variações.	67
5.9	Resultados da relaxação RL-SPRC $_{\xi}$ e suas variações.	69
5.10	Resultados da relaxação RL-SPRC2 e suas variações.	70
5.11	Contagem de vitórias entre as RLs.	72
5.12	Resultados das versões do BRKGA.	74
5.13	Comparação dos melhores LIs entre as metodologias.	76
5.14	Comparação dos melhores LSs entre as metodologias.	78

Sumário

1	Introdução	11
2	Conceitos Preliminares e Formulações do Problema	16
2.1	Notações e Definições	16
2.2	Modelo de PLI DMFM-MRP	18
2.3	Modelo de Programação Inteira Mista - AB-MRP	20
2.3.1	Possíveis valores para os parâmetros M	21
2.4	Relaxação Lagrangiana	23
2.5	Meta-heurísticas	26
2.5.1	Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA) . . .	27
3	Revisão Bibliográfica	29
4	Metodologias	33
4.1	Fixação de Variáveis nos Modelos Matemáticos	33
4.2	Relaxações Lagrangianas	36
4.2.1	Relaxação Lagrangiana - Caminho Mínimo	36
4.2.2	Relaxação Lagrangiana - Caminho Mínimo com Restrição de <i>delay</i> .	37
4.2.3	Relaxação Lagrangiana - Caminho Mínimo com Restrição de <i>jitter</i> .	38
4.2.4	Relaxação Lagrangiana - Caminho Mínimo com Restrição de Múltiplos Recursos	39
4.2.5	Decisões de Implementação	40
4.2.6	Ciclos Negativos	41
4.3	Biased Random-Key Genetic Algorithm	41
4.3.1	Função de Aptidão F1	43
4.3.2	Função de Aptidão F2	43
4.3.3	Função de Aptidão F3	44
4.3.4	Novo procedimento para geração das chaves aleatórias	46
4.4	Heurística de Busca Local	46
4.5	Análise Estatística de Resultados	50
4.6	Geração de Instâncias	52
5	Resultados Computacionais	56
5.1	Pré-processamentos	56
5.2	Modelos DMFM-MRP e AB-MRP	58
5.3	Relaxações Lagrangianas	64
5.3.1	RL-SP	65
5.3.2	RL-SPRC $_{\lambda}$	66
5.3.3	RL-SPRC $_{\xi}$	68

5.3.4	RL-SPRC2	69
5.3.5	Análise geral	71
5.4	BRKGA	73
5.5	Avaliação Geral dos Limitantes Inferiores	76
5.6	Avaliação Geral dos Limitantes Superiores	77
6	Considerações Finais	80
	Referências Bibliográficas	82

Capítulo 1

Introdução

Os avanços nas tecnologias de redes sem fio contribuíram para o surgimento da rede ad hoc móvel, do inglês *Mobile Ad hoc Network* (MANET). MANET é um tipo de rede autoconfigurável composta por um conjunto de nós móveis independentes que são conectados uns aos outros utilizando redes sem fio. Cada nó em uma MANET pode se mover livremente, logo, é comum que suas conexões se alterem com frequência. Estes nós normalmente são dispositivos tais como computadores pessoais, pequenos dispositivos móveis, sensores e telefones celulares.

Há diversas pesquisas interessadas em desenvolver projetos de sistemas de transporte inteligentes, visando uma melhoria na segurança em estradas [40, 25]. Um dos mais importantes sistemas de transporte inteligente é um tipo particular de MANET, conhecido como *Vehicular Ad hoc Network* (VANET) [6]. Nas VANETs, o conjunto de nós da rede é composto por veículos que se movem de acordo com padrões restritos baseados em fatores como sentido da via, regulamentações de trânsito e tráfego [33]. Uma VANET provê comunicação entre veículos, que pode ser de duas maneiras distintas: apenas entre veículos (V2V - *Vehicle to vehicle*) e entre veículos e infraestruturas fixas distribuídas ao longo da estrada como estações base, também chamados de RSU (Roadside Unit), (V2I - *Vehicle to Infrastructure*).

Segundo Karim [23], a aplicação de VANETs é mais indicada para comunicação de redes veiculares por possuir uma sequência de vantagens, tais como o baixo custo de implantação, a possibilidade de comunicação em locais onde não existem infraestruturas fixas e baixa latência na entrega de pacotes de dados, quando comparada com demais tecnologias como redes 3G/4G e *infostation*¹.

As aplicações em VANETs são utilizadas principalmente para auxiliar a comunicação e coordenação entre condutores de veículos, com o objetivo de orientá-los para evitar eventos como acidentes na estrada, engarrafamentos, estradas em más condições, assim como auxiliar no controle do limite máximo de velocidade, levar informações detalhadas sobre acidentes para as equipes de resgate e permitir passagem livre de veículos de emergência. Outra categoria de aplicação é a de entretenimento e conforto para passageiros e motoristas, com acesso à Internet, chats, jogos interativos, reconhecimento de locais livres para estacionamento, dados sobre o preço do combustível, entre outros. Um dos cenários de

¹*Infostation* é um tipo de rede que oferece grande cobertura geográfica e em alta velocidade, contanto que as aplicações tenham tolerância a atrasos significativos na entrega.

funcionamento de VANETs, utilizando comunicação V2V e V2I, é apresentada na Figura 1.1.

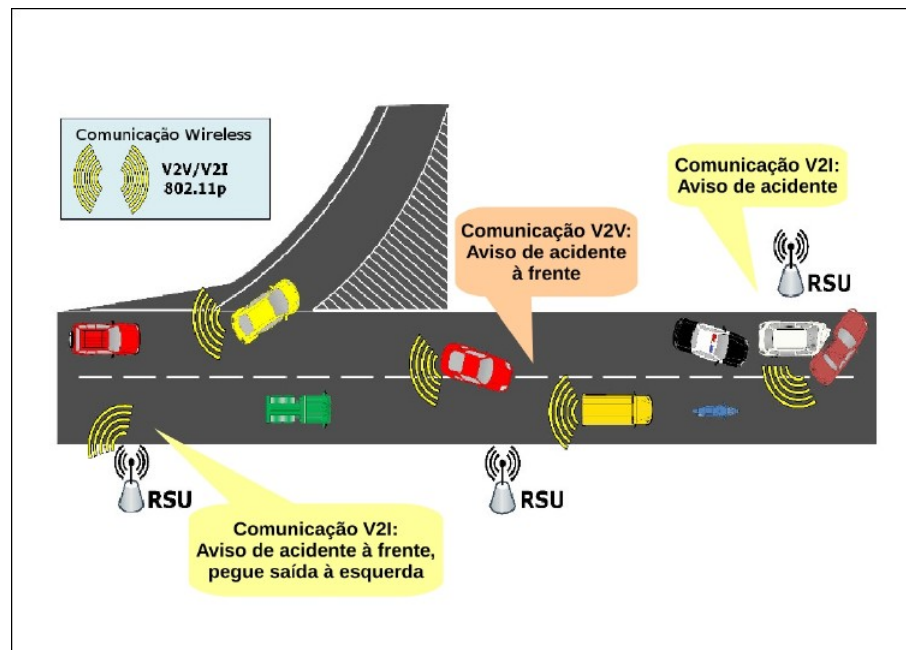


Figura 1.1: Exemplo de funcionamento de uma rede VANET. (Fonte [2])

A maior diferença entre redes MANETs e VANETs está no padrão de mobilidade dos nós, pois as VANETs apresentam uma mobilidade mais alta e seus nós realizam movimentos pré definidos e limitados pelas estradas. O principal desafio em VANETs consiste em manter o roteamento estável, uma vez que as redes geradas pelas conexões entre veículos são muito dinâmicas [29]. Segundo Taleb et al. [41], tendo em vista a alta mobilidade dos nós, os protocolos de roteamento desenvolvidos para MANETs precisam ser devidamente adaptados para atender efetivamente um roteamento otimizado em VANETs.

Segundo Peterson et al. [32], em uma rede de comunicação existem três métodos fundamentais para a transmissão de dados: *unicast*, *broadcast* e *multicast*. O *unicast* é a forma de roteamento onde a comunicação é realizada de um para um, isto é, em cada transmissão realizada um nó age como origem e outro como destino. A abordagem *broadcast* realiza comunicação de um para todos, ou seja, um nó age como origem e todos os outros nós da rede serão destinos. Por fim, o *multicast* realiza a comunicação de um nó que age como origem para um subconjunto de nós que são destinos. A abordagem *multicast* é o método mais eficiente para a comunicação em grupo, reduzindo o desperdício de recursos da rede e tornando mais barato e rápido o custo de comunicação, uma vez que as transmissões para o conjunto de nós destino são realizadas com base em um caminho lógico definido, sem obrigatoriedade de replicação da informação para todos os nós da rede, independente deles estarem interessados ou não na mensagem, como acontece no *broadcast*. Uma desvantagem do *unicast* é que o método gera uma cópia da informação para cada um dos nós pelos quais o pacote de dados deve passar. A Figura 1.2 ilustra as três abordagens de transmissão.

Em VANETs, assim como em outros tipos de redes, as aplicações possuem requisitos especiais em relação aos recursos da rede. A qualidade de serviço, do inglês Quality

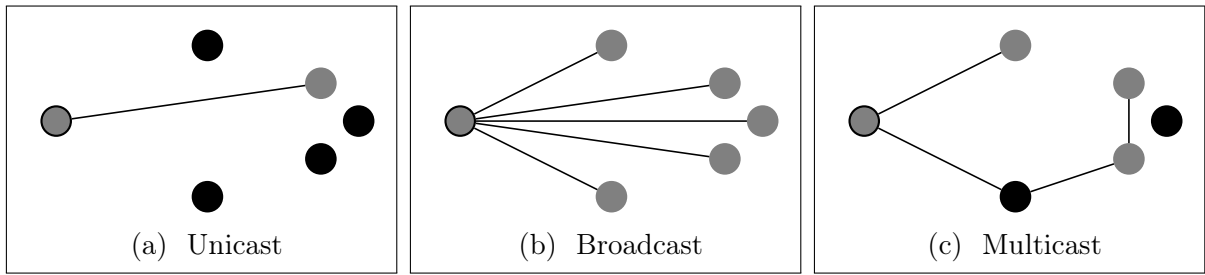


Figura 1.2: Métodos usuais para transmissão de dados em redes

of Service (QoS), está diretamente relacionada com o uso de demandas de recursos da rede, uma vez que em muitas aplicações é extremamente importante manter o serviço da rede com qualidade alta. Por exemplo, aplicações relacionadas à segurança, tais como mensagens de aviso de colisão, acidentes ou más condições da estrada, são sensíveis ao atraso na entrega das mensagens, bem como aplicações de multimídia são sensíveis à largura de banda. Dentre as métricas existentes, vale ressaltar algumas que são de extrema importância para VANETs: atraso fim a fim (*end-to-end delay*), *jitter*, ou seja, variação no atraso entre as mensagens para o mesmo destino, variação do atraso fim a fim entre diferentes destinos, largura de banda (*bandwidth*), quantidade de saltos da mensagem, ou seja, a quantidade de nós pelos quais o pacote passa até o destino final, e estimativa de duração da conexão entre veículos. Neste trabalho serão consideradas as quatro primeiras métricas citadas, explicadas em mais detalhes a seguir.

Entende-se *delay* fim a fim como o tempo gasto para que um pacote seja enviado da origem até o destino. O tempo total consiste na soma dos atrasos do processamento nos nós da rede e o atraso da propagação ao longo do meio de transmissão. O *jitter* é uma medida da variação do atraso na entrega entre sucessivos pacotes de dados em uma conexão. Para exemplificar, se um pacote enviado de uma origem até um destino tem atraso de 15 ms e o envio de outro pacote entre a mesma origem e destino tem atraso de 17 ms, isso significa, de maneira simplificada, que o *jitter* dessa conexão é igual a 2 ms. Segundo Biradar e Manvi [5], é possível reduzir o valor do *jitter* com a utilização de *buffers*, entretanto, isso aumenta consideravelmente a quantidade de memória necessária, fazendo com que seja preferível ter o *jitter* controlado pela própria rede.

A variação do *delay* entre os diferentes destinos pode ser definida como a diferença entre o atraso do caminho que conecta a origem com qualquer par de destinos. Limitar essa variação é essencial para sincronização entre os vários receptores, garantindo que não haja diferença de muitos pacotes de uma mesma mensagem para os vários nós destinatários durante o tempo de vida de uma sessão *multicast* [36]. Por exemplo, assumindo um limite de variação de atraso entre diferentes destinos como 3 ms, se um pacote leva um tempo de 17 ms para ir da origem para um destino, um outro destino deve receber o mesmo pacote com o atraso no intervalo de [14, 20] ms. Por fim, a largura de banda é a medida de capacidade de transmissão de um determinado meio, conexão ou rede, determinando a velocidade máxima em que os dados podem ser transmitidos no enlace.

De maneira mais informal, o problema de roteamento *multicast*, do inglês Multicast Routing Problem (MRP), pode ser descrito como um problema onde dados os custos de transmissão de mensagens entre veículos, busca-se minimizar os gastos para transferir

mensagens nó raiz até um determinado subconjunto de nós terminais. Essa transmissão pode utilizar outros nós da rede, mas não precisa passar por todos eles. Considerar restrições de Qualidade de Serviço, do inglês QoS, trata-se de garantir que os caminhos percorridos para realizar a transmissão da raiz para cada nó terminal obedeçam às restrições de QoS. Essas considerações resultam no Multicast Routing Problem with Quality of Service constraints (MRP-QoS).

Uma propriedade essencial para garantir que uma instância do MRP-QoS seja viável é que exista pelo menos uma solução na qual todos os nós terminais possam ser atendidos. Tipicamente em VANETs os veículos podem ocasionalmente se desconectarem da rede em virtude de suas elevadas mobilidades, ou seja, nem sempre é possível garantir que todos os veículos serão atendidos a todo instante. Sendo assim, foi proposta uma nova variante do MRP-QoS, denominada Máximo Atendimento em MRP com restrições de QoS, do inglês Maximum Service Multicast Routing Problem with Quality of Service Constraints (MS-MRP-QoS), onde nem todos os nós terminais da rede podem ser atendidos. Se o caminho gerado da raiz até um terminal respeita todas as restrições de QoS, esse terminal atendido, caso contrário, o terminal não é atendido. O objetivo do MS-MRP-QoS é maximizar o serviço, i.e., atender o maior número de nós terminais de acordo com as métricas de QoS impostas pela rede.

A Figura 1.3 apresenta um exemplo do processo que constitui a resolução de uma instância do MS-MRP-QoS. A Figura 1.3a representa o grafo inicial, tal que os nós retangulares (azuis) representam o conjunto de veículos terminais (destinos), o nó com círculo duplo (vermelho) é o nó raiz, enquanto os demais nós são opcionais. Todas as conexões (links) tem arcos bidirecionais, ou seja, se por exemplo os nós 1 e 4 estão conectados é possível encaminhar mensagens na direção (1,4) e também (4,1). Cada arco contém 3 métricas associadas, λ (*delay*), ξ (*jitter*) e ω (largura de banda), uma quarta métrica utilizada pelo MS-MRP-QoS é a de variação de *delays* entre terminais. No entanto, ela não se aplica aos arcos individualmente e sim a todos os pares de caminhos utilizados para atender os terminais. A Figura 1.3b apresenta uma solução do MS-MRP-QoS, onde foram gerados caminhos, representados pelos arcos ressaltados (azuis), que atingem todos os terminais. Nesse caso os terminais com retângulo duplo representam veículos atendidos e o retângulo simples os não atendidos. Nesse exemplo, dos quatro terminais, apenas três são atendidos.

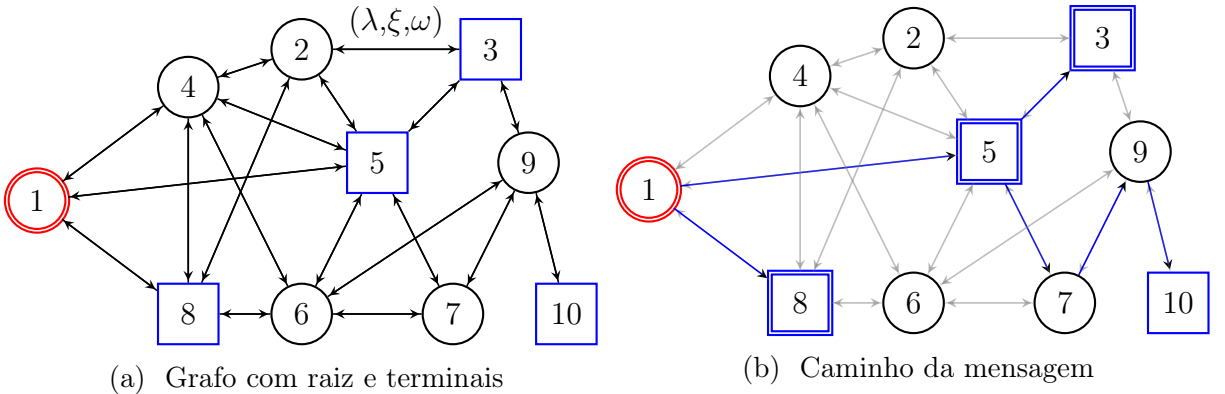


Figura 1.3: Exemplo de Instância do MS-MRP-QoS

Neste trabalho investigamos o MS-MRP-QoS visando encontrar metodologias eficazes para resolvê-lo. Para alcançar esse objetivo, inicialmente resolvemos o MS-MRP-QoS de maneira exata utilizando Programação Linear Inteira Mista (PLIM). Desenvolvemos um conjunto de quatro relaxações lagrangianas para obtenção de limitantes inferiores e superiores com base na dualização de restrições complicadoras. Por fim, foram desenvolvidos métodos heurísticos para obter soluções sem garantia de otimalidade, mas com um baixo uso de recursos computacionais. Uma heurística de busca local em arborescências foi aplicada com objetivo de gerar e aperfeiçoar soluções viáveis em conjunto com a solução das relaxações lagrangianas. Também foram propostos três Algoritmos Genéticos de Chaves Aleatórias Viciadas, do inglês Biased Random-Key Genetic Algorithm (BRKGA), contendo variações no decodificador, no cálculo da função objetivo e no procedimento de geração das chaves aleatórias. Por fim, as instâncias utilizadas para os experimentos foram geradas utilizando simuladores de tráfego e de rede.

Esta dissertação está organizada em seis capítulos. O Capítulo (2) introduz notações e definições necessárias para o entendimento dos demais capítulos, incluindo a descrição dos modelos de Programação Linear Inteira (PLI) e PLIM, e as principais técnicas utilizadas neste trabalho são descritas de modo geral. No Capítulo 3, as principais publicações presentes na literatura para problemas de otimização em redes veiculares são apresentadas, com foco nos resultados obtidos. No Capítulo 4, são discutidas as metodologias utilizadas no desenvolvimento deste trabalho. A descrição dos experimentos e resultados computacionais é feita no Capítulo 5. Por fim, o Capítulo 6 traz as considerações finais do trabalho, seguido das referências.

Capítulo 2

Conceitos Preliminares e Formulações do Problema

Neste capítulo introduzimos conceitos fundamentais para o entendimento desta dissertação. Notações e definições básicas são apresentadas na Seção 2.1. Neste trabalho, empregamos conceitos básicos de Otimização Combinatória, os quais assume-se que são conhecidos. Caso o leitor julgue necessária uma revisão, recomendamos o livro texto de Nemhauser e Wolsey [30], o qual cobre tal tema com enfoque em PLI, uma das principais ferramentas utilizadas neste trabalho. Os conceitos básicos relacionados à teoria dos grafos são considerados conhecidos e caso o leitor julgue necessário uma revisão o conteúdo pode ser encontrado em algum livro texto sobre o tema, por exemplo, Diestel [14]. Os modelos matemáticos estão contidos nas Seções 2.2 e 2.3. A Seção 2.4 contém uma descrição do funcionamento e aplicação da relaxação lagrangiana, uma das principais abordagens por nós utilizada. Por fim, nas Seções 2.5 e 2.5.1 discutimos, de forma geral, meta-heurísticas e BRKGA.

2.1 Notações e Definições

Seja um grafo ponderado e direcionado $G = (V, A)$, sendo $V = \{1, \dots, n\}$ seu conjunto de vértices, $A = \{(u, v) : u \text{ e } v \in V, u \neq v\}$ seu conjunto de m arcos, onde o primeiro vértice do arco é a fonte e também predecessor do segundo vértice do par ordenado, que é conhecido como destino. Tratando-se de grafos não direcionados, podemos substituir a nomenclatura do conjunto de arcos A pelo conjunto de arestas E . Uma árvore T , obtida a partir de um grafo não orientado G , é um subgrafo de G conexo e que não contém ciclos. Para que T seja geradora em G o conjunto de vértices $V(T)$ deve ser igual a $V(G)$, ou seja, todos os vértices do grafo fazem parte da árvore. Uma arborescência, ou árvore enraizada, é um grafo direcionado no qual exatamente um vértice, digamos s , tem grau de entrada 0 e nenhum vértice tem grau de entrada maior que 1, de modo que todos os vértices do grafo são alcançáveis a partir da raiz s .

Segue uma definição formal do MS-MRP-QoS. Seja uma rede VANET representada como um grafo direcionado ponderado $G = (V, A)$. Cada arco de G contém três métricas de QoS associadas: *delay*, *jitter* e largura de banda. Definimos a entrada o MS-MRP-QoS

como uma tupla $(G(V, A), \lambda, \xi, \omega, s, D, \Delta_d, \Delta_j, \Delta_v, \Phi)$, onde:

- $G = (V, A)$ é um grafo ponderado orientado;
- $\lambda_{ij} : A \rightarrow \mathbb{N}$ é uma função que retorna o valor de *delay* para cada arco $(i, j) \in A$;
- $\xi_{ij} : A \rightarrow \mathbb{N}$ é uma função que retorna o valor de *jitter* para cada arco $(i, j) \in A$;
- $\omega_{ij} : A \rightarrow \mathbb{N}$ é uma função que retorna o valor de largura de banda para cada arco $(i, j) \in A$;
- $s \in V$ é definido como a raiz;
- D é o conjunto de vértices terminais, tal que $D \subseteq (V \setminus \{s\})$;
- Δ_d é uma constante que indica o limite de *delay* fim a fim permitido no caminho de s até cada um dos vértices de D ;
- Δ_j é uma constante que indica o limite de *jitter* permitido no caminho de s até cada um dos vértices de D ;
- Δ_v é uma constante que indica o limite da variação dos atrasos fim a fim entre todos os pares de caminhos de s até algum vértice de D ;
- Φ é uma constante que indica o mínimo de largura de banda necessário para que um arco qualquer possa fazer parte da solução;

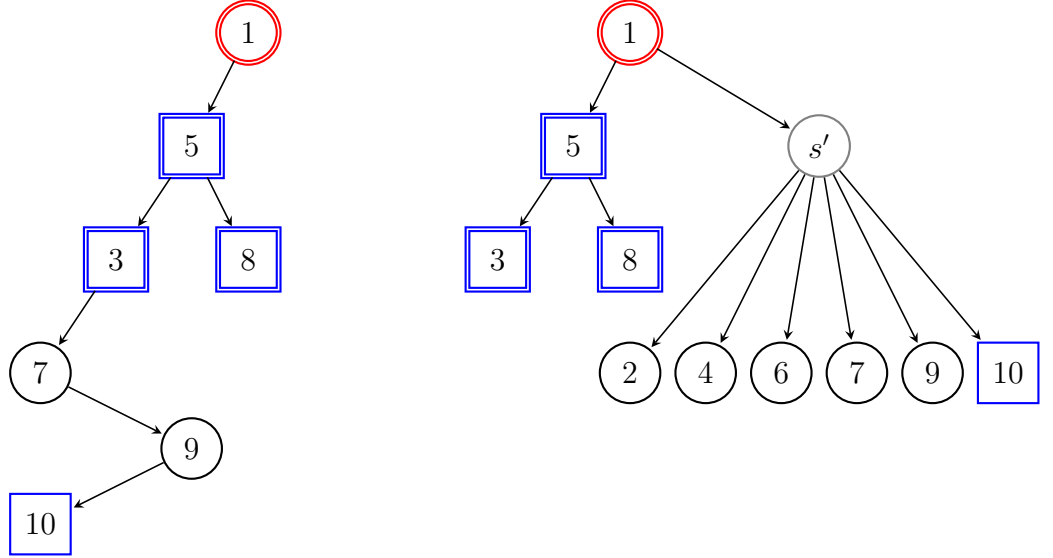
O conjunto $S \in V \setminus (D \cup \{s\})$ é composto por todos os vértices restantes de V . Assim, os vértices do conjunto S são chamados de opcionais ou de *Steiner*. Vale também ressaltar que, como as conexões para comunicação da rede VANET são bidirecionais, se o arco (i, j) pertence a A , então o arco (j, i) também está contido em A com os mesmos valores de QoS. Quando a largura de banda ω de um arco (i, j) está com o valor abaixo do limite demandado pela QoS (Φ), o mesmo não é adicionado ao conjunto A e, conseqüentemente, nunca será parte de uma solução viável do MS-MRP-QoS. Uma arborescência *multicast* T de G tem o vértice fonte s como sua raiz, alcança todos os vértices terminais em D e, possivelmente, alguns vértices de S , mas não necessariamente todos.

O grafo descrito acima permite uma modelagem correta do problema, mas é possível garantir que qualquer solução viável corresponda a uma arborescência geradora. Isso é conveniente pois formulações fortes de programação inteira para arborescências geradoras são conhecidas [28]. As seguintes mudanças foram realizadas no grafo para garantir a existência de uma solução que seja uma arborescência geradora:

1. Adiciona-se um vértice artificial s' e um novo arco artificial de s para s' cujo *delay* é igual a $\Delta_d + 1$ e *jitter* igual a $\Delta_j + 1$.
2. São adicionados arcos de s' para cada vértice $k \in D \cup S$ com *delay* e *jitter* iguais a 0.

O grafo resultante é $G' = (V', A')$, onde $V' = V \cup \{s'\}$ e $A' = A \cup \{(s, s')\} \cup \{(s', k) : k \in D \cup S\}$. Como foi dito anteriormente, soluções viáveis nesse grafo são dadas por arborescências geradoras na qual s é a raiz. Contudo, para representar adequadamente soluções factíveis, algumas restrições precisam ser acrescentadas. A arborescência deve conter o arco (s, s') e se o caminho de s até qualquer terminal k utilizar o nó s' como intermediário, então k não é atendido. O propósito dessas restrições é fazer com que haja uma subárvore onde s' é a raiz e os nós opcionais que não foram utilizados como encaminhadores ou nós terminais não atendidos sejam visitados pela arborescência a partir de s' .

Utilizando como exemplo o grafo G e a arborescência apresentada na Figura 1.3 do capítulo anterior, serão ilustradas duas possíveis soluções viáveis com base em G e G' . A Figura 2.1a descreve uma solução viável obtida a partir de G , enquanto a Figura 2.1b apresenta uma arborescência na qual o mesmo número de terminais é atendido mas a representação da solução é uma arborescência geradora que utiliza o nó e arcos artificiais.

(a) Arborescência obtida a partir de G (b) Arborescência obtida a partir de G'

Com base na adaptação para garantir que toda solução é representada por uma arborescência geradora, desenvolvemos uma formulação de PLI para o MS-MRP-QoS, baseada na formulação de multifluxo apresentada por Magnanti e Wolsey [28], a qual será denotada no restante deste documento por (DMFM-MRP), um mnemônico do inglês *Directed Multicommodity Flow Model* - MRP. A segunda modelagem desenvolvida utiliza programação inteira mista, ou seja, utiliza-se de variáveis reais em conjunto com as variáveis inteiras. Esse segundo modelo foi elaborado com base na remoção das variáveis de fluxo do DMFM-MRP e será denotada no restante do documento por (AB-MRP), um mnemônico do inglês *Arborescence Based* - MRP.

2.2 Modelo de PLI DMFM-MRP

Dada uma instância $I_{MS-MRP-QoS} = (G(V, A), \lambda, \xi, \omega, s, D, \Delta_d, \Delta_j, \Delta_v, \Phi)$ para o MS-MRP-QoS, considere as seguintes variáveis:

- f_{ij}^k : Uma variável binária indicando se o arco (i, j) pertence ao caminho da fonte s até o vértice k tal que $k \in D \cup S$, alternativamente, se o fluxo de s para k passa pelo arco (i, j) ;
- y_{ij} : Uma variável binária que recebe valor 1 caso o arco (i, j) pertença a arborescência ótima e 0 caso contrário;
- z_k : Uma variável binária que recebe valor 1 se o terminal k não é atendido e 0 caso contrário.

Assim, um modelo de programação inteira para o MS-MRP-QoS é:

$$(IP) \quad \min \sum_{k \in D} z_k \quad (2.1)$$

$$\text{s.a.:} \quad \sum_{(s,j) \in A'} f_{sj}^k - \sum_{(j,s) \in A'} f_{js}^k = 1 \quad \forall k \in D \quad (2.2)$$

$$\sum_{(i,j) \in A'} f_{ij}^k - \sum_{(j,i) \in A'} f_{ji}^k = 0 \quad \forall j \in V \setminus \{s, k\}, \forall k \in D \cup S \quad (2.3)$$

$$\sum_{(k,j) \in A'} f_{kj}^k - \sum_{(j,k) \in A'} f_{jk}^k = -1 \quad \forall k \in D \cup S \quad (2.4)$$

$$f_{ij}^k \leq y_{ij} \quad \forall (i, j) \in A', \forall k \in D \cup S \quad (2.5)$$

$$\sum_{(i,j) \in A'} y_{ij} = |V| \quad (2.6)$$

$$\sum_{(i,j) \in A'} \lambda_{ij} f_{ij}^k \leq \Delta_d + M_d z_k \quad \forall k \in D \quad (2.7)$$

$$\sum_{(i,j) \in A'} \xi_{ij} f_{ij}^k \leq \Delta_j + M_j z_k \quad \forall k \in D \quad (2.8)$$

$$\sum_{(i,j) \in A'} \lambda_{ij} (f_{ij}^k - f_{ij}^l) \leq \Delta_v + M_v^k z_k + M_v^l z_l \quad \forall k, l \in D, k \neq l \quad (2.9)$$

$$y_{ss'} = 1 \quad (2.10)$$

$$f_{ij}^k \in \{0, 1\} \quad \forall (i, j) \in A', \forall k \in D \cup S \quad (2.11)$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A' \quad (2.12)$$

$$z_i \in \{0, 1\} \quad \forall i \in D \quad (2.13)$$

O modelo DMFM-MRP apresenta como solução uma arborescência conectando a raiz a todos os nós terminais. A função objetivo (2.1) minimiza o número de terminais não atendidos. As restrições (2.2)-(2.4) impõem a conservação de fluxo em cada vértice, forçando ainda que apenas uma unidade de fluxo saia da fonte s para cada destino $k \in D$. As restrições (2.5) garantem que não haja fluxo passando por um arco (i, j) e dirigindo-se a um vértice $k \in D \cup S$, a menos que esse arco pertença à arborescência ótima. A restrição (2.6) força que a solução tenha $|V| = |D| + |S| + 1$ arcos, portanto, gera todos os vértices de G' . As restrições (2.7)-(2.8) limitam, respectivamente, o delay e o jitter no caminho para cada terminal atendido. As restrições (2.9) exigem que a diferença entre os *delays*

de quaisquer dois caminhos que vão de s até um terminal $k \in D$ não seja maior que Δ_v , caso ambos os terminais sejam atendidos. A restrição (2.10) garante que o arco (s, s') faça parte da solução final, evitando a ocorrência de ciclos com a característica $y_{ij} = y_{ji} = 1$. Finalmente, as restrições (2.11)-(2.13) definem os domínios das variáveis.

É necessário ainda uma descrição mais detalhada das restrições (2.7), (2.8) e (2.9). As duas primeiras restrições tornam-se inativas quando o terminal ao qual elas se referem não for atendido, enquanto a última restrição torna-se inativa quando pelo menos um dos terminais ao qual ela se refere não for atendido. Para isso, os parâmetros M_d , M_j , M_v^k e M_v^l devem ser escolhidos convenientemente de modo a manter as restrições válidas. Possíveis valores para esses parâmetros serão discutidos posteriormente.

2.3 Modelo de Programação Inteira Mista - AB-MRP

Dada uma instância $I_{PMA} = (G(V, A), \lambda, \xi, \omega, s, D, \Delta_d, \Delta_j, \Delta_v, \Phi)$ para o MS-MRP-QoS, considere as seguintes variáveis:

- y_{ij} : Uma variável binária que recebe valor 1 caso o arco (i, j) pertença à arborescência ótima e 0 caso contrário;
- z_k : Uma variável binária que recebe valor 1 se o terminal k não é atendido e 0 caso contrário;
- a_i : Uma variável real que representa o *delay* do caminho de s até i , caso i faça parte da arborescência ótima;
- t_i : Uma variável real que representa o *jitter* do caminho de s até i , caso i faça parte da arborescência ótima;

Assim, o modelo de programação linear inteira mista AB-MRP é proposto a seguir:

$$(\text{MIP}) \min \sum_{k \in D} z_k \quad (2.14)$$

$$\sum_{(i,j) \in A'} y_{ij} = 1 \quad \forall j \in D \cup S \quad (2.15)$$

$$a_j \geq a_i + \lambda_{ij} y_{ij} - M_d(1 - y_{ij}) \quad \forall (i, j) \in A' \quad (2.16)$$

$$a_j \leq a_i + \lambda_{ij} y_{ij} + M_d(1 - y_{ij}) \quad \forall (i, j) \in A' \quad (2.17)$$

$$t_j \geq T_i + \xi_{ij} y_{ij} - M_j(1 - y_{ij}) \quad \forall (i, j) \in A' \quad (2.18)$$

$$a_k \leq \Delta_d + M_d z_k \quad \forall k \in D \quad (2.19)$$

$$t_k \leq \Delta_j + M_j z_k \quad \forall k \in D \quad (2.20)$$

$$a_k - a_l \leq \Delta_v + M_v^k z_k + M_v^l z_l \quad \forall k, l \in D, k \neq l \quad (2.21)$$

$$y_{ij} \in \{0, 1\} \quad \forall (i, j) \in A' \quad (2.22)$$

$$z_k \in \{0, 1\} \quad \forall k \in D \quad (2.23)$$

$$a_i, t_i \geq 0 \quad \forall i \in V \quad (2.24)$$

No modelo AB-MRP uma solução representa uma arborescência geradora mínima onde os valores das métricas de QoS de cada caminho são calculadas nos nós. A função objetivo (2.14) minimiza o número de terminais não atendidos. As restrições (2.15) garantem que a arborescência é geradora. As restrições (2.16)-(2.17) calculam o custo de *delay* em cada vértice j , de modo que, se o arco (i, j) pertencer a arborescência ótima, o valor de a_j é igual a $a_i + \lambda_{ij}$. As restrições (2.18) aplicam o mesmo princípio das restrições (2.16) sobre as variáveis t_i . Para a variável a_k , o *delay* em um nó $k \in D \cup S$ será exatamente o *delay* acumulado no caminho até k . No caso da variável t_k , é suficiente que ela seja maior ou igual ao valor de *jitter* acumulado, pois ao contrário do *delay* não existe uma restrição limitando a variação de *jitter*. Cabe observar que os valores computados para as variáveis l_k e t_k dependem diretamente dos valores das mesmas variáveis no nó predecessor de k . Essa dependência impede a formação de ciclos na solução ótima. As restrições (2.19)-(2.21) limitam respectivamente *delay*, *jitter* e variação de *delay* utilizando a mesma abordagem das restrições (2.7)-(2.9) do modelo DMFM-MRP. Por fim, as restrições (2.22)-(2.24) definem os domínios das variáveis.

2.3.1 Possíveis valores para os parâmetros M

A seguir discutiremos sobre possíveis valores das constantes M_d, M_j, M_v^k e M_v^l . Foram propostos valores que podem ser aplicados à modelagem considerando o grafo G sem a adição dos nós e arcos artificiais. Também foram propostos valores que podem ser utilizados apenas para o grafo G' .

Inicialmente, suponha que os arcos do grafo $G = (V, A)$ tenham sido indexadas de 1 a $m = |A|$, ou seja, $A = \{e_1, e_2, \dots, e_m\}$. Suponha também que $\lambda_{e_1} \geq \lambda_{e_2} \geq \dots \geq \lambda_{e_m}$. Agora, para $n = |V|$, qualquer caminho simples no grafo tem no máximo $n - 1$ arcos e, portanto, $\Lambda = \sum_{i=1}^{n-1} \lambda_{e_i}$ é um limitante superior para o *delay* total de qualquer caminho no grafo. De forma análoga, pode-se definir um limitante superior para o *jitter* de qualquer caminho simples de G , o qual será denotado por Ξ . Também será usada a notação $SP_\lambda(k)$ para o comprimento do caminho mais curto da raiz s para o vértice k de G utilizando como métrica o *delay*.

Com as definições do parágrafo anterior, pode-se computar valores viáveis para as constantes M_d, M_j, M_v^k e M_v^l da seguinte forma:

$$M_d = \Lambda - \Delta_d \quad (2.25)$$

$$M_j = \Xi - \Delta_j \quad (2.26)$$

$$M_v^k = \Lambda - \mu, \text{ onde } \mu = \min\{SP_\lambda(l) + \Delta_v, \Delta_d\} \quad (2.27)$$

$$M_v^l = \Delta_d - \Delta_v - SP_\lambda(l) \quad (2.28)$$

Considerando as definições de Λ e Ξ e analisando as restrições (2.7) e (2.8), temos que se o terminal k for atendido ($z_k = 0$), essas duas restrições limitam o *delay* e o *jitter* conforme determinado pela QoS. Por outro lado, se k não for atendido ($z_k = 1$), o *delay* e o *jitter* acumulado no caminho de s a k ficam restritos de acordo com os seus limitantes superiores Λ e Ξ subtraídos do valor máximo de acumulado para que k seja atendido,

usando como métricas o *delay* e o *jitter*, respectivamente.

A restrição (2.9), que controla a variação de *delay*, requer uma análise mais aprofundada. O objetivo é encontrar um limitante superior para o lado direito da restrição nas quatro situações possíveis de atendimento do par de terminais (k, l) , explicadas a seguir considerando que as constantes M_v^k e M_v^l assumem os valores descritos nas Equações (2.27) e (2.28).

Caso 1 ($z_k = 0, z_l = 0$). Quando os dois terminais, k e l , são atendidos, limita a diferença de *delay* entre os pares em Δ_v e, portanto, é válida neste caso.

Caso 2 ($z_k = 0, z_l = 1$). Quando o terminal k é atendido mas l não, o lado esquerdo da desigualdade será no máximo $\Delta_d - SP_\lambda(l)$ (o terminal k é atendido com o limite máximo de *delay* permitido e o terminal l com o mínimo de *delay* requerido para ir de s a l). Neste caso, o lado direito da restrição será dado por

$$\Delta_v + M_v^l = \Delta_v + \Delta_d - \Delta_v - SP_\lambda(l) = \Delta_d - SP_\lambda(l),$$

garantindo, assim, a validade da desigualdade neste caso.

Caso 3 ($z_k = 1, z_l = 0$). Quando o terminal l é atendido mas k não, o lado esquerdo da desigualdade será no máximo $\Lambda - SP_\lambda(l)$ (o terminal k é alcançado por um caminho simples com o limite máximo de *delay* e o terminal l com o mínimo de *delay* requerido para ir de s a l). Neste caso, o lado direito da restrição será dado por

$$\Delta_v + M_v^k = \Delta_v + \Lambda - \mu \geq \Delta_v + \Lambda - SP_\lambda(l) - \Delta_v = \Lambda - SP_\lambda(l),$$

garantindo, assim, a validade da desigualdade neste caso.

Caso 4 ($z_k = 1, z_l = 1$). Quando o terminal l e k não são atendidos, o lado esquerdo da desigualdade será no máximo $\Lambda - SP_\lambda(l)$ (o terminal k é alcançado por um caminho simples com o limite máximo de *delay* e o terminal l com o mínimo de *delay* requerido para ir de s a l). Neste caso, o lado direito da restrição será dado por

$$\Delta_v + M_v^k + M_v^l = \Delta_v + \Lambda - \mu + \Delta_d - \Delta_v - SP_\lambda(l) = \Lambda - SP_\lambda(l) + (\Delta_d - \mu) \geq \Lambda - SP_\lambda(l),$$

garantindo, assim, a validade da desigualdade neste caso.

Considerando agora o grafo G' , é possível alterar as constantes associadas com as restrições (2.7) e (2.8), utilizando $M_d = M_j = 1$. Esses valores são válidos pois se o caminho da raiz s até o terminal k atender as métricas de *delay* e *jitter*, tanto M_d quanto M_j tornam-se irrelevantes. Além disso, se o terminal k não é atendido então o caminho para visitá-lo passará pelos arcos artificiais (s, s') e (s', k) , que, conforme a construção de G' explicada na Seção 2.1, possuem *delays* $\Delta + 1$ e 0 , e *jitters* $\Delta_j + 1$ e 0 , respectivamente. Portanto, em G' , $\Delta_d + 1$ e $\Delta_j + 1$ são limitantes superiores válidos para *delay* e *jitter*, respectivamente. O objetivo da utilização desses valores para os parâmetros visa desconsiderar todos os terminais que não são atendidos, de modo que, por mais que

exista um caminho para atingir k que não passe pelo nó artificial s' , esse caminho demanda mais que Δ_d ou Δ_j de recurso e portanto k continua como não atendido na arborescência gerada.

2.4 Relaxação Lagrangiana

A Relaxação Lagrangiana (RL) é um conhecido método de decomposição usado na resolução de problemas de otimização combinatória. A ideia principal da RL é remover restrições complicadas do modelo matemático e transferi-las para a função objetivo atribuindo a elas pesos (chamados multiplicadores de Lagrange) que irão penalizar uma solução que não satisfaz alguma das restrições dualizadas. É possível demonstrar que o custo de uma solução ótima da RL sempre será um limitante dual para o valor ótimo do problema original. Um limitante primal pode ser obtido a partir da constatação da viabilidade de uma solução retornada pelo modelo relaxado, bastando para isto computar o valor da função objetivo original para esta solução. Uma etapa importante da RL é determinar os valores para os multiplicadores lagrangianos que resultam no melhor limitante dual. Para isso, pode ser adotado o método de subgradiente que consiste em um processo iterativo por meio do qual os multiplicadores são atualizados iterativamente até convergirem para seus valores ótimos. Considerando um problema originalmente de minimização, este método pode ser visto como a maximização do limite inferior obtido pelo modelo relaxado baseado em escolhas adequadas de multiplicadores [4].

RL é bastante conveniente para problemas que quando são modelados matematicamente de modo que, não fosse por um subconjunto de restrições complicadoras, podem ser resolvidos de maneira eficiente. Por exemplo, considere o seguinte modelo de PLI:

$$\begin{aligned} \text{(IP)} \quad & z = \min cx \\ \text{s.a.} \quad & Ax \geq b, \\ & Dx \geq d, \\ & x \in \mathbb{Z}_+^n. \end{aligned}$$

Sendo $Dx \geq d$ o conjunto de restrições complicadoras do modelo, removê-las resulta em $z' = \min\{cx : x \in X\}$, onde $X = \{x \in \mathbb{Z}_+^n : Ax \geq b\}$, o qual é um problema mais fácil de ser resolvido, podendo ser chamado também de problema relaxado. Dois fatos podem ser observados. O primeiro é que z' é um limitante inferior (dual) de z , uma vez que, com uma restrição a menos delimitando a região viável de x , o valor obtido em z' será menor ou igual a z . O segundo é que a solução ótima em X não necessariamente satisfaz às restrições em $Dx \geq d$. Partindo dessas observações, a ideia é levar as restrições complicadoras para a função objetivo, penalizando-as com um vetor $u \in \mathbb{R}_+^m$. Feito isso, o problema resultante, chamado de Problema Primal Lagrangiano (PPL), pode ser escrito como:

$$\text{PPL}(u) \quad z(u) = \min cx + u(d - Dx)$$

$$\begin{aligned} \text{s.a.} \quad & x \in X, \\ & u \in \mathbb{R}_+^m. \end{aligned}$$

A proposição a seguir estabelece a relação entre $z(u)$ e z .

Proposição 2.4.1. *Seja $z(u) = \min\{cx + u(d - Dx) : x \in X\}$. Então, $z(u) \leq z$ para todo $u \geq 0$.*

A penalidade u_i associada à restrição $D_i x \geq d_i$ é chamada de multiplicador de Lagrange dessa restrição. Tem-se agora o seguinte problema: calcular o conjunto de multiplicadores que correspondem ao melhor, isto é maior, limitante dual $z(u)$. Para encontrar esses valores é necessário resolver o Problema Dual Lagrangiano (PDL), descrito como:

$$\text{PDL}(u) \quad w = \max\{z(u) : u \geq 0\}.$$

Para resolver o PDL existe o algoritmo de otimização por subgradiente, que se baseia no seguinte resultado.

Proposição 2.4.2. *Uma função $g : \mathbb{R}^n \rightarrow \mathbb{R}$ é côncava se e somente se, para todo $\bar{x} \in \mathbb{R}^n$, existe $s \in \mathbb{R}^n$ tal que $g(\bar{x}) + s(x - \bar{x}) \geq g(x)$ para todo $x \in \mathbb{R}^n$.*

Assim, estando em \bar{x} , é necessário escolher a direção que deve ser seguida para maximizar $g(x)$. Utilizando a Proposição 2.4.2, sabe-se que, se x é tal que $g(x) > g(\bar{x})$, então necessariamente $s(x - \bar{x}) > 0$. Ou seja, a partir de \bar{x} , movendo-se uma quantidade adequada na direção de s , g irá aumentar de valor. Logo, é preciso encontrar um vetor s que satisfaça à Proposição 2.4.2. Quando g é diferenciável em \bar{x} , pode-se fazer $s = \nabla g(\bar{x})$, ou seja, tomar o vetor s na direção do gradiente de g em \bar{x} . Porém, se este não for o caso, pela Proposição 2.4.2, sabemos que uma solução ótima x^* satisfaz $s(x^* - \bar{x}) > 0$. Assim, é possível sair de \bar{x} e dar um passo pequeno na direção s de modo a se aproximar mais de um ponto ótimo, ainda que não haja acréscimo no valor da função g . A questão que se coloca é como achar o vetor s cuja existência é garantida na Proposição 2.4.2. Antes porém, define-se o que é um subgradiente e um subdiferencial de uma função em um ponto do seu domínio.

Definição 2.4.1. Se $g : \mathbb{R}^n \rightarrow \mathbb{R}$ é uma função côncava, então s é um subgradiente de g em \bar{x} se e somente se $s(x - \bar{x}) \geq g(x) - g(\bar{x}), \forall x \in \mathbb{R}^n$. O subdiferencial ($\delta g(\bar{x})$) de g em \bar{x} é o conjunto de todos os subgradientes de g neste ponto.

Uma consequência imediata da proposição anterior é dada a seguir.

Proposição 2.4.3. *Se g é côncava e $0 \in \delta g(x^*)$ então $g(x^*) = \max\{g(x) : x \in \mathbb{R}^n\}$, ou seja, x^* é uma solução ótima.*

Portanto, em teoria, se o objetivo é maximizar uma função côncava g , basta começar de um ponto qualquer e iterativamente ir se deslocando, em pequenos passos, na direção de um subgradiente de g naquele ponto até que 0 pertença ao subdiferencial do ponto corrente, ou seja, o ponto atual é ótimo. Os resultados a seguir permitem aplicar a teoria acima à técnica de relaxação lagrangiana para PLI.

Proposição 2.4.4. $z(u) = \min\{cx + u(d - Dx)\}$ é côncava.

O próximo resultado mostra como calcular um subgradiente de $z(u) = \min\{cx + u(d - Dx) : x \in X\}$ no ponto u .

Proposição 2.4.5. *Seja $\bar{x} \in X$ tal que $z(u) = c\bar{x} + u(d - D\bar{x})$. Então, $(d - D\bar{x})$ é um subgradiente de $z(u)$ em u .*

A partir dos resultados e definições anteriores, estamos aptos a apresentar um procedimento para minimizar uma função côncava para a qual se conhece um subgradiente em todos os pontos do seu domínio. Para tal, um pseudocódigo em alto nível do método de subgradiente é apresentado no Algoritmo (1).

Algoritmo 1: Método de Subgradiente (Problema original de minimização)

Entrada: *limiar*, *maxIter*, *atualizaPi*

Saída: z_{LB} , z_{UB}

```

1   $m \leftarrow \rho \leftarrow 0$ ;
2   $z_{LB} \leftarrow 0$ ;
3   $z_{UB} \leftarrow$  custo de uma solução viável;
4   $\alpha^0 \leftarrow \theta^0 \leftarrow 0$ ;
5   $\pi^0 \leftarrow 2$ ;
6  enquanto  $(\frac{z_{UB} - z_{LB}}{z_{UB}}) \leq \textit{limiar}$  ou  $m < \textit{maxIter}$  faça
7       $x \leftarrow$  solução da  $RL(\alpha^m)$ ;
8       $z^m \leftarrow$  custo da função objetivo de  $x$ ;
9      se  $z^m > z_{LB}$  então
10          $z_{LB} \leftarrow z^m$ ;
11      $z_f^m \leftarrow$  função objetivo original da solução  $x$ ;
12     se  $x$  é viável e  $z_f^m < z_{UB}$  então
13          $z_{UB} \leftarrow z_f^m$ ;
14          $\rho \leftarrow 0$ ;
15      $\rho \leftarrow \rho + 1$ ;
16     se  $\rho = \textit{atualizaPi}$  então
17          $\pi \leftarrow \pi/2$ ;
18          $\rho \leftarrow 0$ ;
19      $\theta^m \leftarrow$  subgradiente( $x$ );
20      $n^m \leftarrow$  norma( $\theta^m$ );
21      $s^m \leftarrow \pi^m \frac{(z_{UB} - z^m)}{\|n^m\|^2}$ ;
22      $\alpha^{m+1} \leftarrow \max(0, \alpha^m + s^m \theta^m)$ ;
23      $m \leftarrow m + 1$ ;

```

O algoritmo recebe três parâmetros de entrada: *limiar*, *maxIter* e *atualizaPi*. Os dois primeiros são utilizados como condições de parada. O parâmetro *limiar* indica o valor máximo de *gap* para que a solução seja considerada ótima, encerrando assim a execução. O *maxIter* limita o máximo de iterações do método do subgradiente. O terceiro e último parâmetro, *atualizaPi*, serve de contador para atualizações do valor de π , ou seja, quando houver uma quantidade *atualizaPi* de iterações consecutivas sem melhora do limitante dual. As linhas (1)-(5) representam apenas o processo de inicialização de variáveis, onde

m é o contador de repetições do laço principal, ρ é contador de iterações consecutivas, sem que haja redução no valor de z_{UB} . As variáveis z_{LB} e z_{UB} são os limitantes inferior e superior, respectivamente. O símbolo α representa o vetor de multiplicadores de Lagrange e θ é o vetor de subgradientes. Primeiramente resolve-se o problema primal lagrangiano utilizando os valores de multiplicadores atuais e essa solução servirá para obtenção do vetor de subgradiente. Para um problema de maximização, se o valor obtido na resolução do modelo relaxado for menor que o valor do limitante superior atual, atualiza-se z_{UB} e caso a solução do problema relaxado seja viável para o problema original, o valor do limitante inferior, z_{LB} , é atualizado com o valor da função objetivo desconsiderando o custo dos multiplicadores. Após essas avaliações é calculado o vetor de subgradiente (θ^m) e a norma deste vetor (n^m) é utilizada para computar o tamanho do passo utilizado (s^m) na direção do subgradiente. Com isso, atualiza-se o valor dos multiplicadores lagrangianos para a próxima iteração do algoritmo. A linha (26) garante que o multiplicador seja não-negativo.

Neste trabalho desenvolvemos quatro relaxações lagrangianas baseadas no modelo DMFM-MRP. Um detalhe importante a considerar é a resolução das relaxações lagrangianas quando as restrições (2.7), (2.8) ou ambas não são dualizadas, devendo ser tratadas no PPL, ou seja, quando este se torna um problema de caminhos mais curtos com restrição de um ou mais recursos.

2.5 Meta-heurísticas

Uma meta-heurística é uma metodologia de solução para problemas de otimização combinatória que fornece uma estrutura e estratégias gerais para guiar o processo de solução de um método heurístico que se ajusta a um tipo de problema particular. Atualmente as meta-heurísticas se tornaram uma das mais importantes ferramentas para os profissionais da área de Pesquisa Operacional [20].

As meta-heurísticas podem ser classificadas em dois conjuntos: meta-heurísticas singulares, tais como Simulated Annealing, GRASP, VNS e Busca Tabú, que abordam uma única solução e sua estratégia de busca consiste em explorar a vizinhança da solução utilizada; e o segundo conjunto, chamado de meta-heurísticas populacionais, tais como Algoritmo Genético (AG), Colônia de Formigas e Enxame de Partículas, onde o foco está na manutenção de um conjunto de soluções sobre as quais o processo de busca atua, combinando e modificando atributos dessas soluções visando explorar o espaço de busca.

Por exemplo, nos algoritmos genéticos cada indivíduo (também chamado de cromossomo) da população representa uma solução para o problema que está sendo abordado. Cada solução tem sua qualidade determinada por uma função de aptidão e o procedimento de busca segue através de uma quantidade de gerações, nas quais os indivíduos contribuem para a formação da população na geração seguinte, com prioridade aos indivíduos com melhor aptidão. Dentre os operadores mais comuns do AG tradicional, destacam-se o *crossover* por meio do qual ocorre a obtenção de um novo indivíduo a partir da combinação de duas, ou mais, soluções da população (pais) e a mutação que visa acrescentar diversidade a busca, alterando indivíduos de modo que incorporem na solução que eles

representam características distintas dos seus pais.

2.5.1 Algoritmo Genético de Chaves Aleatórias Viciadas (BRKGA)

Os algoritmos genéticos com chaves aleatórias (Random-Key Genetic Algorithm (RKGA)) foram introduzidos por Bean [3], para tratar problemas clássicos de sequenciamento. Um motivo comum para a escolha de RKGA está no fato de que para alguns problemas, AGs tradicionais têm dificuldade em manter a viabilidade de soluções durante a operação de *crossover*. RKGAs solucionam esta dificuldade através de uma representação robusta que utiliza chaves aleatórias e um decodificador para cada tipo de problema. O BRKGA é uma meta-heurística que obtém uma solução viável para um problema a partir da decodificação de um cromossomo de números aleatórios (também chamados de chaves). Este processo de decodificação consiste em um método heurístico que, em termos computacionais, é desejável que apresente um custo relativamente baixo. As chaves são números reais gerados aleatoriamente no intervalo $[0, 1[$. O decodificador deve retornar uma solução viável independente da sequência de chaves apresentada pelo cromossomo e para isso, todo decodificador exige uma representação cromossômica apropriada.

De maneira mais formal, um RKGA evolui uma população de c vetores de chaves aplicando cruzamentos e mutações, onde os indivíduos mais fortes de uma população têm mais chance de se tornarem pais e com isso perpetuar características de sua solução. O algoritmo começa com uma população inicial de c vetores de n chaves aleatórias e produz uma série de populações. Na i -ésima geração, os c vetores da população são particionados em um subconjunto de vetores que correspondem às melhores soluções (este conjunto se chama elite) e um outro subconjunto com o restante da população (chamado de não-elite). Todos os vetores de elite são salvos, sem mudança alguma, para a população da $(i + 1)$ -ésima geração. Logo após, são inseridos c_m vetores de chaves aleatórias na população da $(i + 1)$ -ésima geração. Esses vetores são chamados de mutantes e tem o mesmo papel dos operadores de mutação nos AGs tradicionais, ou seja, evitar que a população convirja para um ótimo local não global. Para completar os c elementos da população, são gerados vetores combinando pares de soluções da população da k -ésima geração, ambos escolhidos aleatoriamente, com uma combinação uniforme. Sendo a e b os vetores escolhidos como pais e d o filho resultante, no esquema mais comum, $d[j]$, o j -ésimo componente do vetor filho, recebe a j -ésima chave de um dos pais, com probabilidade ρ_a de receber $a[j]$ e $\rho_b = 1 - \rho_a$ de receber $b[j]$.

O BRKGA, como apresentado por [17], vai um pouco além do RKGA no que se refere às operações de cruzamento e seleção de pais. Ambos os algoritmos selecionam pais aleatoriamente e com reposição, sendo assim um pai pode ter mais de um filho por geração. Ambos os algoritmos fazem a combinação dos pais a e b com o cruzamento para gerar o filho d . Enquanto no RKGA os pais podem ser quaisquer indivíduos da população, no BRKGA o pai a é do conjunto elite e o pai b é não elite. Por fim, $\rho_a > \frac{1}{2}$ no BRKGA, fazendo com que o filho d tenha maior probabilidade de herdar as chaves do pai elite, diferente do RKGA. Essas diferenças entre o RKGA e o BRKGA fazem com que os resultados do BRKGA, na prática, sejam superiores ao RKGA.

Neste trabalho a solução gerada pelo decodificador consiste em computar uma árvore

geradora mínima onde o custo de cada aresta é o valor de uma chave aleatória. Com base nessa abordagem desenvolvemos quatro decodificadores diferentes, variando desde as funções de aptidão de cada indivíduo, até uma nova proposta de viés associado com a etapa de geração das chaves aleatórias. Mais detalhes serão apresentados na Seção 4.

Capítulo 3

Revisão Bibliográfica

Neste Capítulo, descrevemos as abordagens e resultados obtidos por trabalhos disponíveis na literatura. Nosso foco são os trabalhos que realizaram estudos sobre algoritmos para o problema de roteamento em VANETs e novos protocolos otimizados.

Segundo Peterson [32], um protocolo fornece um serviço de comunicação que objetos (nós da rede) usam para troca de mensagens e informações. No caso da comunicação *multicast*, os protocolos são utilizados para enviar informações de uma fonte para um grande conjunto de destinos utilizando apenas uma operação de envio. Uma parte essencial dos protocolos de comunicação é o roteamento [31]. Do ponto de vista matemático, pode-se ver o roteamento como um algoritmo que manipula os usuários e enlaces da rede, de modo que os pacotes enviados por um dos nós pode seguir um caminho para o destino utilizando apenas enlaces selecionados e, caso existam restrições de QoS, deve-se garantir que todos os nós destinos recebem as informações respeitando os limites impostos pelas métricas utilizadas na rede. Assim, desenvolver procedimentos para construção de rotas otimizadas, significa otimizar também os protocolos de comunicação.

Para construção de uma rota é necessário primeiramente que o nó de origem (raiz) tenha conhecimento da topologia da rede como um todo. O processo de reconhecimento dos nós pode variar de acordo com o protocolo, entretanto o método mais utilizado é o de inundação (do inglês *flooding*), que consiste em um nó realizando um *broadcast* para todos os nós ao seu alcance. Os nós que recebem a mensagem, por sua vez, replicam para os demais da rede utilizando a mesma abordagem. Ao final, o nó que inicialmente enviou a mensagem consegue ter conhecimento total da topologia da rede. Perceba que esse processo não garante que um veículo, após o processo de reconhecimento da rede, se mantenha conectado.

Oliveira e Pardalos [31] afirmam que o MRP se assemelha com o problema clássico da Árvore Mínima de Steiner, do inglês Minimum Steiner Tree (MST), o qual é \mathcal{NP} -difícil e possui uma extensa literatura, por exemplo [16, 27, 45]. A MST pode ser utilizada tanto para formulação quanto para representação da solução em aplicações do MRP. As versões mais estudadas do MST, associadas ao MRP, são as que consideram atraso fim a fim [31]. Vale ressaltar ainda que a combinação de múltiplas restrições de QoS tornam o problema ainda mais difícil de ser resolvido.

Bitam e Mellouk [6] desenvolveram um algoritmo baseado em enxame de abelhas para resolver MRP-QoS em VANETs. Dado o nó origem e um conjunto de nós destino, o

objetivo é computar uma árvore que atenda todos os terminais na qual a soma dos custos das conexões seja mínimo e todas as métricas de QoS sejam respeitadas. As 4 métricas contidas em cada conexão são: custo de utilização, *delay*, *jitter* e largura de banda. O peso atribuído a cada métrica na função objetivo corresponde à sua importância de atendimento para qualidade de serviço. O algoritmo de enxame de abelhas foi comparado com outros da literatura em um cenário simulado contendo 20 nós (veículos) em uma área de 1200 metros quadrados durante um período de 120 segundos e, por se tratar de uma simulação, os veículos trafegam de forma aleatória no trecho de mapa fornecido para o simulador. A criação das instâncias foi feita com base na obtenção da topologia da rede no tempo de 85 segundos da simulação, ou seja, é computada uma fotografia que mostra o posicionamento dos nós e o valor das métricas para todas as conexões existentes na rede. Os resultados mostraram que o algoritmo de enxame de abelhas apresentou melhorias em comparação com demais trabalhos, incluindo convergência mais rápida para a melhor solução.

Os autores Sebastian *et al.* [39] apresentaram uma mudança no esquema de roteamento para fazer a disseminação de mensagens de segurança mais eficientemente em VANETs. O problema de roteamento *multicast* foi formulado como uma MST com restrições de atraso. O esquema de roteamento detecta os veículos que estão envolvidos e ameaçados em casos de acidentes e os coloca em um conjunto de destinatários das mensagens de alerta. Há também um recurso que visa aumentar a eficiência dos canais de comunicação da rede sem fio, visto que há uma seleção dos nós que receberão tais alertas de emergência, reduzindo assim o número de mensagens enviadas.

Fazio *et al.* [15] desenvolveram um novo protocolo para VANETs que tem em seu núcleo um algoritmo para construção de rotas otimizadas. Esse algoritmo leva em consideração três métricas: *delay* fim a fim, interferência co-canal e probabilidade de duração da conexão. A probabilidade de duração do enlace é calculada por um método proposto pelos próprios autores. O objetivo do trabalho é encontrar uma árvore geradora mínima considerando todas as restrições de QoS. A validação do protocolo desenvolvido foi feita com base na comparação de desempenho, utilizando um simulador de redes, com outros protocolos clássicos. O protocolo proposto apresentou resultados superiores na avaliação das métricas de largura de banda, taxa de entrega de pacotes e atraso fim a fim. Porém, os autores notaram uma pequena sobrecarga de processamento. Além da simulação do protocolo, houve a avaliação do algoritmo em relação ao seu desempenho em construir rotas de acordo com as restrições.

Ribeiro *et al.* [34] desenvolveram uma formulação de PLI e uma heurística *relax-and-fix* para resolver o MRP-QoS aplicado às VANETs. Dado um nó origem e um conjunto de nós destino, o objetivo é encontrar a árvore que minimiza a soma da quantidade de saltos e *delay* fim a fim no caminho, visando maximizar a seleção de *links* com maior estimativa de duração de conexão e obedecendo os limites máximos de *delay* acumulado, *jitter* acumulado, variação máxima entre *delays* totais para todos os pares de caminhos e o limite mínimo de largura de banda para cada conexão. As instâncias de *benchmark* foram geradas utilizando simuladores, buscando proximidade com configurações reais. O trabalho não faz comparação da heurística com a literatura, mas apresenta um conjunto de técnicas de pré-processamento e a diferença de desempenho relacionada com a utilização

dessas técnicas em conjunto com o modelo e o *relax-and-fix*.

É comum que autores abordem otimizações em redes VANETs propondo novas extensões de protocolos de comunicação com algoritmos otimizados. Souza [11] apresentou um protocolo de roteamento para VANETs que é baseado na meta-heurística colônia de formigas. Com base no protocolo MAODV (*Multicast Ad hoc On-demand Distance Vector routing protocol*) [37], o autor desenvolveu uma extensão que utiliza informações de mobilidade dos veículos para aumentar a estabilidade do roteamento *multicast*, essa extensão é chamada MAV-AODV. Para construção e manutenção de árvores *multicast* foi utilizado o algoritmo baseado em colônia de formigas. O método de avaliação do protocolo utilizou um ambiente de simulador de redes com uma quantidade variável de veículos (entre 25 e 100), por 150 segundos e em um cenário de 1600×1500 metros. Para análise do desempenho foram utilizados dois outros protocolos (MAODV e PUMA, *Protocol for Unified Multicasting through Announcement* [43]) e cinco métricas de QoS: *delay* fim a fim, variação do *delay* fim a fim, sobrecarga de roteamento, redundância e taxa de entrega dos pacotes. Em relação ao MAODV, o MAV-AODV obteve resultados superiores na maioria das métricas. Porém, ao comparar os resultados com o PUMA, o protocolo MAV-AODV obteve melhor desempenho apenas na métrica de redundância de pacotes.

Correia *et al.* [10] desenvolveram um protocolo de roteamento baseado no DYMO (*Dynamic MANET On-demand*) [18] que utiliza uma estratégia de otimização baseada no algoritmo de colônia de formigas. Por ser um protocolo reativo, o DYMO permite a integração do procedimento de otimização efetuando operações do algoritmo de colônia de formigas, tais como adição e evaporação de feromônios nas rotas disponíveis. A estratégia de otimização utiliza também a velocidade e posição dos veículos para auxiliar na escolha das rotas. Este protocolo foi validado em um software de simulação de rede, com um cenário de tamanho 1600×1500 metros, a quantidade de veículos dividida em 5 tamanhos, 25, 50, 75, 100 e 150, e comparado com dois outros protocolos, DYMO nativo e AODV (*Ad hoc On-demand Distance Vector routing protocol*) [37]. Para fins de comparação, foram consideradas 3 métricas de QoS: taxa média de entrega dos pacotes, *delay* fim a fim e sobrecarga de roteamento. A análise dos resultados confirmou uma melhora do novo protocolo em relação ao DYMO nativo apenas para a métrica de *delay* fim a fim, enquanto o mesmo protocolo obteve melhor desempenho para as métricas de taxa média de entrega e sobrecarga de roteamento em relação ao AODV.

Bitam *et al.* [7] propuseram um protocolo de roteamento híbrido que utiliza as vantagens dos protocolos de roteamento baseados em topologia em conjunto com as vantagens dos protocolos de roteamento baseados em informações geográficas. Este protocolo utiliza uma abordagem baseada em colônia de abelhas e aplica o conceito de roteamento *unicast*, ou seja, existe uma origem e um destino por vez. Além disso, o algoritmo possui um módulo de otimização, baseado em algoritmo genético, cuja finalidade é tratar do roteamento com base em informações geográficas. O resultado a ser retornado por este módulo consiste em uma rota entre o nó origem e o destino, levando em consideração o posicionamento dos nós, custo de utilização das conexões, *delay* fim a fim e a largura de banda mínima. Este protocolo foi validado utilizando simulador de redes e comparando os resultados com os protocolos AODV (baseado em topologia) e GPRS (*General Packet Radio Service*) que é baseado em informações geográficas. A movimentação dos veículos

da rede foi simulada com base em um modelo de tráfego em um trecho de 1000 metros quadrados da cidade de Biskra, na Argélia. Os experimentos trataram de 20, 35 e 50 veículos, analisando três métricas de QoS, *delay* fim a fim, taxa de pacotes entregues e sobrecarga na rede. O protocolo proposto apresentou bom desempenho, obtendo melhores resultados se tratando da métrica de *delay* fim a fim. Enquanto que para a métrica de taxa de pacotes entregues, os resultados foram competitivos com os resultados do AODV, apresentando uma diferença inferior a 1%. Por fim, para a métrica de sobrecarga da rede, o protocolo proposto apresentou vantagem sobre o AODV e perda em relação ao GPRS.

Saleet *et al.* [38] desenvolveram uma classe de protocolos para VANETs chamado IGRP (*Intersection-based Geographical Routing Protocol*), que apresentou resultados superiores em comparação com demais protocolos de roteamento aplicados em ambientes urbanos. A ideia principal do IGRP está na seleção de interseções de ruas pelas quais o pacote deve passar para atingir o seu destino. Esta seleção é feita de modo que existe uma garantia, com alta probabilidade, de conectividade da rede ao longo das interseções, enquanto satisfaz as métricas de QoS utilizadas. O IGRP faz uso de uma rotina baseada em algoritmo genético para seleção de interseções. O protocolo proposto foi avaliado utilizando uma ferramenta para simulação de redes que executou por 1000 segundos em um cenário contendo entre 150 e 600 nós. As comparações para análise de desempenho se deram entre o protocolo desenvolvido e demais trabalhos contidos na literatura, OLSR (*Optimized Link State Routing protocol*) [8], GPSR (*Greedy Perimeter Stateless Routing*) [24] e GPCR (*Greedy Perimeter Coordinator Routing*) [26]. O IGRP alcançou os melhores resultados se tratando das métricas de *delay* e quantidade de saltos, o que influencia diretamente na taxa de erro de entrega de mensagens, na qual o protocolo proposto também obteve vantagem em relação aos demais.

Capítulo 4

Metodologias

Neste capítulo são apresentadas as metodologias aplicadas para solução do MS-MRP-QoS e para análise estatística dos resultados.

4.1 Fixação de Variáveis nos Modelos Matemáticos

Um conjunto de pré-processamentos foram aplicados em ambos os modelos, DMFM-MRP e AB-MRP, para fixação de variáveis. Os procedimentos foram apresentados em Ribeiro et al. [34] e adaptados para o MS-MRP-QoS.

Dado um nó terminal k e um arco (i, j) é possível estabelecer que uma variável y_{ij} ou f_{ij}^k pode ser removida do modelo, preservando pelo menos uma solução ótima. Para verificar a necessidade de utilizar um arco (i, j) no caminho da raiz s até um terminal k , são computados limitantes inferiores para o *delay* e *jitter* necessários para construir tal caminho.

Seja r_{ij} um recurso associado com o arco (i, j) do grafo G . Um terminal k só pode ser classificado como atendido se, para um caminho P que vai da raiz s até k , a soma dos valores de r não excedem uma quantidade Δ_r . Seja LB_{si} um limitante inferior da quantidade de recurso r necessário para ir da raiz s até o nó i , e analogamente, seja LB_{jk} um limitante inferior da quantidade de recurso r necessário para ir de j até k . Logo, se

$$LB_{si} + r_{ij} + LB_{jk} > \Delta_r, \quad (4.1)$$

é possível afirmar que, com a utilização do arco (i, j) no caminho da raiz s até o terminal k , o terminal k não será atendido. Logo, a variável f_{ij}^k pode ser fixada em zero ou removida do modelo. Para o MS-MRP-QoS, existem dois recursos a serem considerados: *delay* e *jitter*. Ribeiro et al. [34] desenvolveram três diferentes variações de cálculo dos limitantes da Equação (4.1). Este trabalho aplica os dois procedimentos mais eficientes para eliminação de variáveis nas formulações, dado que o terceiro procedimento é dominado pelos demais.

Considere uma instância do MS-MRP-QoS definida por um grafo G e as funções λ e ξ que representam, respectivamente, os valores de *delay* e *jitter* de cada arco. As seguintes notações e definições são utilizadas:

- G_λ : grafo direcionado G com os custos dos arcos dados pela função λ e os recursos dados pela função ξ ;
- G_ξ : grafo direcionado G com os custos dos arcos dados pela função ξ e os recursos dados pela função λ ;
- G_r^T : o transposto do grafo G_r para $r \in \{\lambda, \xi\}$ (i.e., com as direções dos arcos invertidas);
- $SP(r, u, v)$: o caminho mínimo, do inglês Shortest Path (SP), de u para v no grafo G_r , onde $r \in \{\lambda, \xi\}$ e u e v correspondem à qualquer par de nós distintos em G_r ;
- $SPRC(r, u, v, \Delta_t)$: o caminho mínimo de u para v no grafo G_r restrito a usar até Δ_t unidades de recursos t , do inglês Shortest Path with Resource Constraints (SPRC), onde r e $t \in \{\lambda, \xi\}$, $r \neq t$ e u e v correspondem à qualquer par de nós distinto em G_r ; e
- $SPRC^T(r, u, v, \Delta_t)$: mesmo que acima, mas em G_r^T .

Os pré-processamentos seguem a mesma nomenclatura utilizada em [34]. Os procedimentos podem ser classificados em fraco, moderado e forte, dependendo da quantidade de vezes que os caminhos mínimos com e sem restrições de recursos são computados para obtenção dos limitantes da equação (4.1). Os procedimentos utilizados neste trabalho são os seguintes:

1. Remoção de Vértices Moderada, do inglês Moderated Vertex Elimination (MVE): Dado um nó $i \in S$ e recursos r e $t \in \{\lambda, \xi\}$ com $r \neq t$, se

$$SPRC(r, s, i, \Delta_t) + SP(r, i, k) > \Delta_r, \forall k \in D, \quad (4.2)$$

então o vértice i pode ser removido do grafo. Isso decorre da quantidade mínima de recurso r necessária para ir da raiz s até qualquer terminal k , passando pelo nó i , excede o limite permitido para que k seja atendido. A remoção pode ser aplicada no grafo ou fixando todas as variáveis y_{ij}, y_{ji}, f_{ij}^k e f_{ji}^k para todo $k \in D$, em 0.

2. Remoção de Arcos Moderada, do inglês Moderated Arc Elimination (MAE): Dado um arco $(i, j) \in A$ e recursos r e $t \in \{\lambda, \xi\}$ com $r \neq t$, se

$$SPRC(r, s, i, \Delta_t) + r_{ij} + SP(r, j, k) > \Delta_r, \forall k \in D \quad (4.3)$$

então o arco (i, j) pode ser removido do grafo. Isso decorre da quantidade mínima de recurso r para ir da raiz s até qualquer terminal k , utilizando o arco (i, j) , ultrapassar o limite permitido para que k seja atendido. A remoção pode ser aplicada no grafo ou fixando todas variáveis y_{ij} e f_{ij}^k para todo $k \in D$, em 0.

3. Remoção de Arcos Forte, do inglês Strong Arc Elimination (SAE): Dado um arco $(i, j) \in A$, um terminal $k \in D$ e os recursos r e $t \in \{\lambda, \xi\}$ com $r \neq t$, considerando a Equação (4.1). O valor do limitante LB_{si} pode ser computado da seguinte maneira:

$$LB_{si} = SPRC(r, s, i, \Delta_t - \min_{l \in D, l \neq i} \{SP(t, i, l)\}). \quad (4.4)$$

Esse valor é computado considerando que, em um fluxo enviado de s para um terminal k utilizando o arco (i, j) , a quantidade de recurso t consumido para ir de i até k é limitado inferiormente pelo valor de $\min\{SP(t, i, l) : l \in D, l \neq i\}$, ou seja, o caminho mínimo em função de t indo de i até qualquer terminal $l \neq i$. Assim, resta no máximo Δ_t subtraído do valor desse caminho mínimo. Consequentemente, o tamanho do caminho mínimo com restrição de recurso dado por essa fórmula é um limitante inferior da quantidade de recurso r gasto para ir da raiz s até o nó i . O valor de LB_{jk} pode ser obtido por:

$$LB_{jk} = SPRC^T(r, k, j, \Delta_t - SPRC(t, s, j, \Delta_r)). \quad (4.5)$$

O termo $\Delta_t - SPRC(t, s, j, \Delta_r)$ corresponde ao limitante superior de recurso t disponível para ir de j até k , considerando que utilizou-se o mínimo possível desse recurso no caminho de s até j . O motivo do computar esse caminho com base no grafo G^T está relacionado com a complexidade computacional e será discutida posteriormente. Finalmente, se

$$\begin{aligned} & (SPRC(r, s, i, \Delta_t - \min_{l \in D, l \neq i} \{SP(t, i, l)\})) + r_{ij} \\ & + SPRC^T(r, k, j, \Delta_t - SPRC(t, s, j, \Delta_r)) > \Delta_r, \end{aligned} \quad (4.6)$$

então o arco (i, j) pode ser removido do caminho que vai da raiz s até o terminal k . Para isso, a variável f_{ij}^k é fixada em zero.

Ribeiro et al. [34] apresentam a complexidade computacional dos testes propostos. Toda computação é feita no grafo $G = (V, A)$ ou seu transposto. Sendo $n = |V|$, $m = |A|$ e o número de nós terminais $d = |D|$, o tempo de execução dos testes depende diretamente do custo computacional de gerar os caminhos em cada caso. Sejam $\#SP$ e $\#SPRC$ as complexidades de computar, respectivamente, caminhos mínimos sem e com restrição de recurso em G ou G^T . $\#SP$ tem complexidade (n^2) com apenas um nó raiz e $O(n^3)$ para computar caminho mínimos entre todos os pares de vértices (ver, e.g., Cormen et al., [9]). Por outro lado, computar caminhos mínimos com restrição de recurso é \mathcal{NP} -difícil. Assim, a complexidade dos testes é dada por (a) MVE: $O(n \cdot \#SPRC + n^2 \cdot \#SP)$; (b) MAE: $O(n \cdot \#SPRC + n^2 \cdot \#SP)$; e (c) SAE: $O(n \cdot \#SPRC + n^2 \cdot \#SP) + O((d+n) \cdot \#SPRC)$, de modo que o primeiro e segundo termos do último somatório são as complexidades associadas às Equações (4.4) e (4.5), respectivamente.

Note que, com a utilização do algoritmo de Floyd-Warshall para computar o caminho mínimo entre todos os pares, os termos $n^2 O(\#SP)$ podem ser substituídos por $O(n^3)$. Ainda, veja que no segundo termo correspondente à complexidade do teste SAE, utiliza-se o do grafo transposto para evitar o cômputo de caminhos mínimos com restrição de recursos entre todos os pares de nós em $V \setminus \{s\}$. Isso ocorre pela possibilidade de restringir

somente os terminais do conjunto de nós utilizados como raiz.

Mesmo com o problema de caminho mínimo com restrição de recurso sendo \mathcal{NP} -difícil, existem algoritmos exatos rápidos o suficiente para resolver este problema na prática [21]. A implementação dos pré-processamentos utilizam o algoritmo de configuração de rótulos disponível na biblioteca Boost C++ [1]. Os tempos de execução dos pré-processamentos para a maioria das instâncias geradas neste trabalho foram irrelevantes se comparado com o tempo de execução do resolvidor de PLI. Mais detalhes sobre os pré-processamentos são descritos na Seção 5.1.

4.2 Relaxações Lagrangianas

Nesta seção, são apresentadas RLs obtidas a partir do modelo DMFM-MRP, proposto na Seção 2.2. Os multiplicadores de Lagrange estão representados por ϕ 's, de modo que $\phi^{2.5}$, $\phi^{2.7}$, $\phi^{2.8}$ e $\phi^{2.9}$, todas variáveis reais não negativas, estão associados com as restrições (2.5), (2.7), (2.8) e (2.9), respectivamente. Todas as relaxações mantêm, em seu PPL, problemas de caminho mínimo, do inglês SP ou caminho mínimo com restrições de recurso, do inglês SPRC. A nomenclatura utilizada para cada relaxação segue o padrão RL-X, onde "X" representa o subproblema mais difícil de ser resolvido do PPL.

4.2.1 Relaxação Lagrangiana - Caminho Mínimo

A RL que resulta da dualização das restrições (2.5) e (2.7)-(2.9) é bastante próxima do problema de computar SPs e, por isso, é denominada RL-SP. De fato, as restrições (2.2)-(2.4) modelam caminhos mínimos da raiz s para os demais nós do grafo, utilizando as variáveis f_{ij}^k para todo $k \in D \cup S$. Considerando as variáveis y_{ij} , o problema associado consiste na seleção do arco (s, s') juntamente com os $|V'| - 2$ arcos de menor coeficiente computados a partir dos multiplicadores lagrangianos. Como o PPL não contém restrições que se apliquem às variáveis z_k para todo $k \in D$, o problema associado com essas variáveis consiste na decisão de não atender o terminal k ($z_k = 1$) se o coeficiente gerado, considerando os multiplicadores lagrangianos, for menor que zero, caso contrário, $z_k = 0$. Assim, ao dualizar as restrições (2.5) e (2.7)-(2.9), o PPL resultante é:

$$\begin{aligned}
 (\text{RL-SP}) \quad \min \quad & \sum_{k \in D} z_k (1 - \phi_k^{2.7} M_d - \phi_k^{2.8} M_j - \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} M_v^k + \phi_{lk}^{2.9} M_v^l)) \\
 & + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5} + \xi_{ij} \phi_k^{2.8} + \lambda_{ij} (\phi_k^{2.7} + \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} - \phi_{lk}^{2.9}))) \\
 & + \sum_{q \in S} \sum_{(i,j) \in A'} f_{ij}^q (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) \\
 & - \sum_{k \in D} (\phi_k^{2.7} \Delta_d + \phi_k^{2.8} \Delta_j + \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} \Delta_v))
 \end{aligned} \tag{4.7}$$

s.a.:

(2.2) – (2.4), (2.6) e (2.10)

Foram utilizadas duas abordagens para resolução do RL-SP. A primeira estratégia de resolução do PPL utiliza resolvidor de PLI para obtenção de soluções a cada iteração do método de subgradiente. Na segunda abordagem, a resolução do PPL consiste em resolver os subproblemas associados ao conjunto de variáveis $(f, y$ e $z)$ utilizando algoritmos combinatórios. Um fato importante, que influencia na complexidade de resolução do problema de SP, está relacionado com a possibilidade de existirem coeficientes negativos associados às variáveis f_{ij}^k . Consequentemente, ao computar um SP, a solução retornada pode conter ciclos negativos, o que implica no valor da função objetivo do PPL ser igual a $-\infty$. Para evitar essas situações, foi explorada a resolução de uma versão simplificada de RL-SP na qual descarta-se um subconjunto de multiplicadores, mais especificamente $\phi^{2.9}$. Desconsiderar esses multiplicadores da função objetivo impede a existência de arcos, associados às variáveis f , cujos coeficientes sejam negativos. Assim, a nova função objetivo, denominada RL-SP', pode ser descrita do seguinte modo:

$$\begin{aligned}
 (\text{RL-SP}') \quad & \min \sum_{k \in D} z_k (1 - \phi_k^{2.7} M_d - \phi_k^{2.8} M_j) \\
 & + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5} + \xi_{ij} \phi_k^{2.8} + \lambda_{ij} \phi_k^{2.7}) \\
 & + \sum_{q \in S} \sum_{(i,j) \in A'} f_{ij}^q (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) \\
 & - \sum_{k \in D} (\phi_k^{2.7} \Delta_d + \phi_k^{2.8} \Delta_j)
 \end{aligned} \tag{4.8}$$

A respeito da complexidade da versão combinatória da relaxação RL-SP', considere $n = |V|$, $m = |A|$, $d = |D|$ e $\#SP$ como a complexidade de computar caminhos mínimos em G' . Ainda, considere $G^k = (V', A'^k)$ para todo $k \in D \cup S$, como subgrafo de G' induzido pelas variáveis f_{ij}^k . A diferença nos arcos do conjunto A'^k , para cada k , se deve à utilização do pré-processamento SAE. Assim, o problema combinatório associado às variáveis f tem complexidade $O((n-1) \cdot \#SP)$. Para as variáveis y , a complexidade é dada pela ordenação dos arcos de A' com os coeficientes lagrangianos, ou seja, $O(m \log m)$. Por fim, o problema computacionalmente mais simples, podendo ser resolvido em $O(d)$, consiste em verificar os coeficientes lagrangianos associados a cada variável z .

4.2.2 Relaxação Lagrangiana - Caminho Mínimo com Restrição de *delay*

A RL obtida a partir da dualização das restrições (2.5) e (2.7)-(2.9) resulta no problema de obter caminhos mínimos com restrição de *delay* e, por isso, será referenciada como RL-SPRC $_{\lambda}$. considerando as variáveis f_{ij}^k , as restrições (2.2)-(2.4) e (2.7) modelam caminhos mínimos com restrição de *delay* indo de s para os nós terminais do grafo, enquanto, para os nós opcionais, basta computar caminhos mínimos. O problema associado com as

variáveis y_{ij} se mantém como descrito na relaxação RL-SP. Para a variável z_k existe duas situações nas quais seu valor se torna igual a um. O primeiro caso ocorre se o coeficiente gerado pelos multiplicadores lagrangianos for menor que zero, enquanto o segundo caso está associado com os caminhos induzidos pelas variáveis f_{ij}^k , ou seja, se para visitar o terminal k o *delay* utilizado por um caminho excede Δ_d , então k torna-se não atendido e consequentemente $z_k = 1$. Ao dualizar as restrições (2.5) e (2.8)-(2.9), o PPL resultante é:

$$\begin{aligned}
 (\text{RL-SPRC}_\lambda) \quad & \min \sum_{k \in D} z_k (1 - \phi_k^{2.8} M_j - \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} M_v^k + \phi_{lk}^{2.9} M_v^l)) \\
 & + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5} + \xi_{ij} \phi_k^{2.8} + \lambda_{ij} (\sum_{\substack{l \in D \\ l \neq k}} \phi_{kl}^{2.9} - \phi_{lk}^{2.9})) \\
 & + \sum_{k \in S} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) \\
 & - \sum_{k \in D} (\phi_k^{2.8} \Delta_j + \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} \Delta_v)) \tag{4.9}
 \end{aligned}$$

s.a.

$$(2.2) - (2.4), (2.6) - (2.7) \text{ e } (2.10)$$

Para resolver o problema combinatório associado com as variáveis f_{ij}^k é necessário obter um conjunto de SPs e/ou SPRCs. Primeiramente, deve-se resolver dois subproblemas para cada terminal k . O primeiro considerando k como atendido e o segundo como não atendido. Esses dois subproblemas estão sujeitos aos valores utilizados pelos parâmetros M_d e M_j , conforme será melhor explicado na Seção 4.2.5.

Como descrito para a relaxação RL-SP, a dualização das restrições (2.9) possibilita a existência de coeficientes negativos associados às variáveis f_{ij}^k , podendo inviabilizar a utilização do algoritmo de SP. Seguindo a mesma simplificação adotada anteriormente, pode-se reescrever a função objetivo, referenciando-a como $\text{RL-SPRC}_\lambda'$, do seguinte modo:

$$\begin{aligned}
 (\text{RL-SPRC}'_\lambda) \quad & \min \sum_{k \in D} z_k (1 - \phi_k^{2.8} M_j) + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5} + \xi_{ij} \phi_k^{2.8}) \\
 & + \sum_{k \in S} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) - \sum_{k \in D} (\phi_k^{2.8} \Delta_j) \tag{4.10}
 \end{aligned}$$

4.2.3 Relaxação Lagrangiana - Caminho Mínimo com Restrição de *jitter*

Essa relaxação segue as mesmas propriedades da relaxação RL-SPRC_λ , modificando apenas o recurso de *delay* para *jitter*, por esse motivo é intitulada de RL-SPRC_ξ . Assim, removendo a dualização das restrições (2.7) e dualizando (2.8), temos o seguinte PPL.

$$\begin{aligned}
(\text{RL-SPRC}_\xi) \quad & \min \sum_{k \in D} z_k (1 - \phi_k^{2.7} M_d - \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} M_v^k + \phi_{lk}^{2.9} M_v^l)) \\
& + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\lambda_{ij} (\phi_k^{2.7} + \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} - \phi_{lk}^{2.9}))) \\
& + \sum_{k \in S} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) \\
& - \sum_{k \in D} (\phi_k^{2.7} \Delta_d + \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} \Delta_v)) \tag{4.11}
\end{aligned}$$

s.a.

(2.2) – (2.4), (2.6), (2.8) e (2.10)

E, por motivos análogos aos apresentados anteriormente, a função objetivo foi simplificada da seguinte forma.

$$\begin{aligned}
(\text{RL-SPRC}'_\xi) \quad & \min \sum_{k \in D} z_k (1 - \phi_k^{2.7} M_d) + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\lambda_{ij} \phi_k^{2.7}) \\
& + \sum_{k \in S} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) - \sum_{k \in D} (\phi_k^{2.7} \Delta_d) \tag{4.12}
\end{aligned}$$

4.2.4 Relaxação Lagrangiana - Caminho Mínimo com Restrição de Múltiplos Recursos

A RL obtida pela dualização das restrições (2.5) e (2.9) resulta no problema de computar caminhos mínimos com restrições de recursos, nesse caso *delay* e *jitter*, e por esse motivo será referenciada como RL-SPRC2. De fato, considerando as variáveis f_{ij}^k , as restrições (2.2)-(2.4) e (2.7)-(2.8) modelam SPRCs restritos em *delay* e *jitter* da raiz s até cada um dos nós terminais do grafo. Enquanto, para os nós opcionais, basta computar caminhos mínimos a partir de s . Em relação às variáveis y e z , os problemas combinatórios a serem resolvidos são os mesmos já descritos anteriormente. Assim, ao dualizar as restrições (2.5) e (2.9), o PPL resultante é:

$$\begin{aligned}
(\text{RL-SPRC2}) \quad & \min \sum_{k \in D} z_k (1 - \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} M_v^k + \phi_{lk}^{2.9} M_v^l)) \\
& + \sum_{k \in D} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5} + \lambda_{ij} \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} - \phi_{lk}^{2.9})) \\
& + \sum_{k \in S} \sum_{(i,j) \in A'} f_{ij}^k (\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij} (\phi_{ijk}^{2.5}) - \sum_{k \in D} \sum_{\substack{l \in D \\ l \neq k}} (\phi_{kl}^{2.9} \Delta_v) \tag{4.13}
\end{aligned}$$

s.a.:

$$(2.2) - (2.4), (2.6) - (2.8) \text{ e } (2.10)$$

E, por motivos já esclarecidos, mesmo com a utilização de dois recursos simultaneamente no computo do SPRC, a função objetivo foi simplificada como:

$$(\text{RL-SPRC2}') \quad \min \sum_{k \in D} z_k + \sum_{k \in D \cup S} \sum_{(i,j) \in A'} f_{ij}^k(\phi_{ijk}^{2.5}) - \sum_{k \in D \cup S} \sum_{(i,j) \in A'} y_{ij}(\phi_{ijk}^{2.5}) \quad (4.14)$$

A respeito da complexidade das versões combinatórias das relaxações RL-SPRC_λ , RL-SPRC_ξ e RL-SPRC2 , considere $n = |V|$, $m = |A|$, $d = |D|$, $\#SP$ como a complexidade de computar um SP e $\#SPRC$ a complexidade de computar um SPRC, ambos em G' . A análise do tempo de resolução dos problemas associados às variáveis y_{ij} e z_k são as mesmas descritas na Seção 4.2.1. A complexidade computacional dos problemas associados às variáveis f_{ij}^k é $O(d \cdot \#SPRC + n \cdot \#SP)$, se $M_d = \Lambda$ e $M_j = \Xi$, e $O(2 \cdot d \cdot \#SPRC + (n - d - 1) \cdot \#SP)$ quando $M_d = M_j = 1$. Tais diferenças serão melhor explicadas na Seção 4.2.5.

4.2.5 Decisões de Implementação

Em relação aos PPLs em que as variáveis f_{ij}^k modelam problemas de caminhos mínimos com restrições de recurso, é necessário destacar a diferença de subproblemas a serem resolvidos de acordo com os valores atribuídos aos parâmetros M_d e M_j . Iniciando pelo caso no qual $M_d = \Lambda$ e $M_j = \Xi$, para cada terminal $k \in D$ devem ser resolvidos dois subproblemas descritos a seguir, o primeiro quando k é atendido e o segundo para k não atendido.

Subproblema 1 ($z_k = 0$). Considerando que o terminal k é atendido, deve-se computar $\text{SPRC}(s, k, \Delta_r)$ com $r \in \{d, j\}$, ou seja, o caminho mínimo com restrição de recurso r de s até k . Considere $w_1(k)$ o custo desse caminho.

Subproblema 2 ($z_k = 1$). Considerando que o terminal k não é atendido, a quantidade de recurso disponível para o caminho modifica-se de Δ_r para M_r com $r \in \{d, j\}$. Esse acréscimo permite computar de maneira viável qualquer um dos seguintes caminhos, $\text{SPRC}(s, k, M_r)$ ou $\text{SP}(s, k)$. Considere o custo desse caminho como $w_2(k)$.

Quando $M_d = M_j = 1$, resolve-se o mesmo subproblema 1 descrito no caso em que $z_k = 0$, ou seja, o valor de $w_1(k)$ se mantém inalterado. A diferença consiste na etapa de calcular $w_2(k)$, quando k não é atendido. A quantidade de recurso r que pode ser utilizada é incrementada de Δ_r para $\Delta_r + 1$, tal que $r \in \{d, j\}$. Portanto, o valor de $w_2(k)$ é igual à $\text{SPRC}(s, k, \Delta_r + 1)$, inviabilizando a solução do problema de caminho mínimo.

Após computar os valores de $w_1(k)$ e $w_2(k)$ para um $k \in D$, o caminho que fará parte da solução do PPL é aquele que apresentar o menor valor associado à função objetivo, i.e., seleciona-se a solução de custo $w(k)$ tal que $w(k) = \min\{w_1(k), w_2(k) + 1\}$, onde o

valor 1 somado ao custo do caminho $w_2(k)$ representa a penalização na função objetivo pelo fato de k não ser atendido.

4.2.6 Ciclos Negativos

Uma observação referente à solução do problema de SPRC é que a presença de arcos com custos negativos não inviabilizam o algoritmo contanto que os consumos de recursos dos arcos sejam estritamente maiores que zero. Desse modo, para a relaxação RL-SPRC2 notou-se, dada a análise do conjunto de instâncias, a presença de subconjuntos de arcos cujo *jitter* é igual a zero, inviabilizando assim a obtenção de solução por algoritmos combinatórios. Mais detalhes serão apresentados na Seção 5.3.

Notam-se algumas diferenças ao considerar a solução do PPL utilizando resolvidor de PLI e a versão com algoritmos combinatórios. Primeiramente, os caminhos obtidos pelo resolvidor são simples, i.e., não repetem-se arcos. Assim, é possível obter soluções para o PPL independente da existência de ciclos negativos. A segunda diferença consiste no caminho de s para visitar cada nó opcional $q \in S$, de modo que o resolvidor de PLI considera os casos em que q é utilizado como intermediário no caminho para visitar cada terminal. Enquanto, na versão combinatória, computa-se o caminho mínimo de s até q , para todo $q \in S$.

4.3 Biased Random-Key Genetic Algorithm

Esta seção descreve as implementação de decodificadores do BRKGA para gerar uma solução viável para o MS-MRP-QoS a partir do vetor de chaves aleatórias recebido como parâmetro [42]. Foram desenvolvidas quatro diferentes versões do decodificador discutidas ao longo desta seção. O algoritmo 2 apresenta os procedimentos comuns a todos os decodificadores desenvolvidos.

Algoritmo 2: Decodificador BRKGA $O(m \log m + \#BL + \#f)$

Entrada: c, G

// c : vetor de chaves aleatórias com tamanho $|E|$

// G : grafo (V, E)

1 **para toda** aresta $e \in E$ **faça**

2 | peso(e) $\leftarrow -c[e]$;

3 $T \leftarrow$ AGM computada em G ;

4 $T \leftarrow T$ com as arestas direcionadas a partir da raiz em direção às folhas;

5 Aplicar busca local em T ;

6 **retorna** valor da função de aptidão computada em T ;

No Algoritmo 2, o primeiro passo consiste em atribuir a cada aresta do grafo G o valor negativo de uma chave aleatória de c , como apresentado na linha (2). Na linha (3) aplica-se o algoritmo de Kruskal (ver Cormen et al. [9]) para computar uma Árvore Geradora Mínima (AGM) em G . Neste trabalho utilizamos a implementação fornecida pela biblioteca da Boost para determinar a AGM. Como uma solução viável para o MS-MRP-QoS consiste em uma arborescência, a linha (4) direciona as arestas utilizadas em

T . O passo da linha (5) é opcional e consiste na aplicação de uma etapa de busca local em T , procedimento que será melhor detalhado na Seção 4.4. Por fim, a linha (6) retorna o valor de aptidão com base na função selecionada.

A seguir será explorada a complexidade computacional do decodificador. Assuma $n = |V|$, $m = |E|$ e $d = |D|$. O procedimento para atualizar o custo dos arcos tem complexidade $O(E)$. Para computar a AGM utiliza-se o algoritmo de Kruskal (ver, e.g., Cormen et al. [9]), que tem complexidade computacional $O(m \log m)$. O procedimento para direcionar as arestas de T a partir da raiz pode ser feito em tempo $O(n)$. A complexidade do decodificador depende também da utilização da busca local e da função de aptidão, cujas complexidades serão referenciadas por $\#BL$ e $\#F$, respectivamente. Inicialmente, é possível assumir a complexidade total do decodificador como $O(m \log m + \#BL + \#F)$, de modo que o termo $O(m \log m)$ está associado ao procedimento de maior custo computacional até a linha (4), ou seja, computar a AGM.

Uma vez determinada a arborescência T a partir das chaves aleatórias, cabe agora determinar a quantidade máxima de terminais que podem ser atendidos por essa arborescência. Primeiro verifica-se o conjunto de terminais que podem ser atendidos considerando apenas as restrições de *delay* e *jitter*. Desse conjunto é selecionado o subconjunto máximo de terminais que também atendem a restrição de variação de *delay*. O algoritmo 3 é responsável por essa seleção que, em linhas gerais, consiste na criação de um vetor ordenado dos *delays* de todos os terminais atendidos (considerando apenas as restrições de *delay* e *jitter*), para em seguida avaliar todos os subconjuntos ordenados maximais e selecionar o maior.

Algoritmo 3: Seleção Por Variação de *delay* $O(d \log d)$

Entrada: D'

```

1 Ordenar  $D'$  de maneira não-decrescente considerando o valor de delay;
2 maior  $\leftarrow 0$ ;
3 maiorsub  $\leftarrow \emptyset$ ;
4  $j \leftarrow 1$ ;
5 para  $i = 1, \dots, |D'|$  faça
6    $\alpha \leftarrow D'[i].\text{delay}$ ;
7   sub  $\leftarrow D'[i].k$  até  $D'[j].k$ ;
8   enquanto  $D'[j].\text{delay} \leq \alpha + \Delta_v$  and  $j \leq |D'|$  faça
9      $j \leftarrow j + 1$ ;
10    sub  $\leftarrow \text{sub} \cup \{D'[j].k\}$ ;
11  se  $|\text{sub}| > \text{maior}$  então
12    maior  $\leftarrow |\text{sub}|$ ;
13    maiorsub  $\leftarrow \text{sub}$ ;
14 retorna  $|\text{maior}_{\text{sub}}|$ ;
```

Com D' sendo um vetor que contém os pares $(delay, k)$, representando o $delay$ do caminho de s até o terminal k na arborescência T . Ordenar D' de maneira não-decrescente pelo valor de $delay$ possibilita a verificação organizada de todos os subconjuntos maximais, sempre utilizando dois ponteiros para marcar o início e o fim do subconjunto. O ponteiro i visita cada índice de D' , enquanto o segundo ponteiro, j , posiciona-se sempre a frente de i , indicando o último elemento desse subconjunto, ou seja, o terminal mais distante de i cujo $delay$ é menor ou igual à $delay(i) + \Delta_v$. Esse procedimento tem complexidade computacional dominada pela ordenação, que pode executar em $O(d' \log d')$, dado que o procedimento de verificação dos subconjuntos maximais é feito em tempo linear, $O(d)$.

4.3.1 Função de Aptidão F1

A função de aptidão F1 é a mais simples dentre as funções. Inicialmente computa-se o $delay$ e o $jitter$ no caminho de s para cada terminal em T . Assim, basta percorrer a lista de terminais e contabilizar aqueles que não são atendidos por pelo menos uma das duas métricas. Em seguida, monta-se o vetor D' com os terminais atendidos. Ao final de F1, são retornados os terminais não atendidos. A versão do BRKGA que aplica a função de aptidão F1 será referenciada como BRKGA-F1. A complexidade computacional do Algoritmo 4 é dominada pela função de *SeleçãoPorVariação*, dado que a complexidade de criação do vetor D' é $O(d)$.

Algoritmo 4: Função de Aptidão (F1) $O(d \log d)$

Entrada: T

```

1 não-atendidos  $\leftarrow 0$ ;
2  $D' \leftarrow \emptyset$ 
3 para  $k \in D$  faça
4    $d \leftarrow$  custo de  $delay$  do caminho de  $s$  até  $k$  em  $T$ ;
5    $j \leftarrow$  custo de  $jitter$  do caminho de  $s$  até  $k$  em  $T$ ;
6   se  $d > \Delta_d$  ou  $j > \Delta_j$  então
7     não-atendidos  $\leftarrow$  não-atendidos +1;
8   senão
9      $D' \leftarrow \{d, k\}$ ;
10 retorna não-atendidos + ( $|D|$  - SeleçãoPorVariação( $D'$ ));
```

4.3.2 Função de Aptidão F2

A função de aptidão F2, apresentada no Algoritmo 5, utiliza uma penalização associada aos valores que excede as métricas de QoS violadas quando um terminal não é atendido. Seja λ_k o $delay$ para ir de s a k na arborescência T , assumindo que k não é atendido. Como $\lambda_k > \Delta_d$, a penalização associada com a métrica de $delay$ é calculada como $\frac{d' - \Delta_d}{\Delta_d}$. O mesmo cálculo se aplica para as métricas de $jitter$ e variação de $delay$. Para priorizar o atendimento dos terminais na F.O., foi utilizado um coeficiente $|D|$ para cada terminal não atendido. A versão do BRKGA que utiliza a função de aptidão F2 será referenciada como BRKGA-F2.

Algoritmo 5: Função de Aptidão (F2) $O(d^2)$

Entrada: T

```

1 delaydif  $\leftarrow$  0; jitterdif  $\leftarrow$  0; vardif  $\leftarrow$  0;
2 não-atendidos  $\leftarrow$  0;
3 para  $k \in D$  faça
4    $d \leftarrow$  custo de delay do caminho de  $s$  até  $k$  em  $T$ ;
5    $j \leftarrow$  custo de jitter do caminho de  $s$  até  $k$  em  $T$ ;
6   se  $d > \Delta_d$  ou  $j > \Delta_j$  então
7     não-atendidos  $\leftarrow$  não-atendidos + 1;
8     se  $d > \Delta_d$  então
9       delaydif  $\leftarrow$  delaydif +  $\frac{d - \Delta_d}{\Delta_d}$ ;
10    se  $j > \Delta_j$  então
11      jitterdif  $\leftarrow$  jitterdif +  $\frac{j - \Delta_j}{\Delta_j}$ ;
12  senão
13     $D' \leftarrow \{d, k\}$ ;
14 para  $i = 1, 2, \dots, (|D| - 1)$  faça
15   para  $j = i + 1, \dots, |D|$  faça
16      $d_i \leftarrow$  custo de delay no caminho de  $s$  até  $D[i]$  em  $T$ ;
17      $d_j \leftarrow$  custo de delay no caminho de  $s$  até  $D[j]$  em  $T$ ;
18     se  $|d_i - d_j| > \Delta_v$  então
19       vardif  $\leftarrow$  vardif +  $\frac{|d_i - d_j| - \Delta_v}{\Delta_v}$ ;
20 não-atendidos  $\leftarrow$  não-atendidos + ( $|D|$  - SeleçãoPorVariação( $D'$ ));
21 retorna não-atendidos  $\times |D|$  + (delaydif + jitterdif + vardif);

```

O laço das linhas (3)-(13) consiste no processo de armazenar o *delay* e o *jitter* do caminho para cada terminal k na árvore T . Verifica-se a violação da restrição de alguma métrica e em caso afirmativo, computa-se a penalização associada ao terminal. O segundo laço, das linhas (14)-(19), calcula o valor de variação de *delay* para todos os pares de terminais, verificando quando a diferença excede Δ_v e computando a penalização. Por fim, a linha (21) retorna o valor da função de aptidão, ou seja, a penalização da quantidade de terminais não atendidos somada à penalização computada para cada métrica.

4.3.3 Função de Aptidão F3

A função de aptidão F3, é uma variação da função F2. Enquanto F2 utiliza a soma das penalizações de todas as métricas, a função de aptidão F3, disponível no algoritmo 6, considera apenas a menor penalização dentre as métricas excedidas. O objetivo dessa abordagem é destacar na função de aptidão apenas o terminal que está mais próximo de ser atendido. A versão do BRKGA que utiliza a função de aptidão F3 será denominada como BRKGA-F3

O primeiro laço, nas linhas (3)-(17), ao invés de realizar a soma das penalizações de cada terminal, armazena apenas o menor valor excedente dentre todos os terminais, em cada métrica. O mesmo se aplica ao laço das linhas (18)-(25) que verifica as restrições de variação de *delay*. O algoritmo retorna na linha (27), a penalização por terminais não

atendidos somada à menor penalização dentre as métricas.

Algoritmo 6: Função de Aptidão (F3) $O(d^2)$

Entrada: T

```

1  $delay_{min} \leftarrow \infty$ ;  $jitter_{min} \leftarrow \infty$ ;  $var_{min} \leftarrow \infty$ ;
2  $n\grave{a}o-atendidos \leftarrow 0$ ;
3 para  $k \in D$  faça
4    $d \leftarrow$  custo de delay do caminho de  $s$  até  $k$  em  $T$ ;
5    $j \leftarrow$  custo de jitter do caminho de  $s$  até  $k$  em  $T$ ;
6   se  $d > \Delta_d$  ou  $j > \Delta_j$  então
7      $n\grave{a}o-atendidos \leftarrow n\grave{a}o-atendidos + 1$ ;
8     se  $d > \Delta_d$  então
9        $dif_d \leftarrow \frac{d - \Delta_d}{\Delta_d}$ ;
10      se  $dif_d < delay_{min}$  então
11         $delay_{min} \leftarrow dif_d$ ;
12      se  $j > \Delta_j$  então
13         $dif_j \leftarrow \frac{j - \Delta_j}{\Delta_j}$ ;
14        se  $dif_j < delay_{min}$  então
15           $delay_{min} \leftarrow dif_j$ ;
16    senão
17       $D' \leftarrow \{d, k\}$ ;
18 para  $i = 1, 2, \dots, (|D| - 1)$  faça
19   para  $j = i + 1, \dots, |D|$  faça
20      $d_i \leftarrow$  custo de delay do caminho de  $s$  até  $D[i]$  em  $T$ ;
21      $d_j \leftarrow$  custo de delay do caminho de  $s$  até  $D[j]$  em  $T$ ;
22     se  $|d_i - d_j| > \Delta_v$  então
23        $dif \leftarrow \frac{|d_i - d_j| - \Delta_v}{\Delta_v}$ ;
24       se  $dif < var_{min}$  então
25          $var_{min} \leftarrow dif$ ;
26  $n\grave{a}o-atendidos \leftarrow n\grave{a}o-atendidos + SeleçãoPorVariação(D')$ ;
27 retorna  $n\grave{a}o-atendidos \times |D| + \min(delay_{min}, jitter_{min}, var_{min})$ ;

```

4.3.4 Novo procedimento para geração das chaves aleatórias

Por fim, foi desenvolvida uma variação do BRKGA que consiste na alteração da geração do vetor de chaves aleatórias. Como cada chave aleatória está relacionada a uma aresta de G , esse novo procedimento incorpora, ao valor de cada chave, a frequência que a aresta relacionada aparece nas soluções geradas pelas RLs. Pode-se obter essa frequência ao longo das iterações do método de subgradiente, contabilizando a quantidade de vezes que cada aresta esteve presente na solução computada pelo PPL, ou seja a frequência das variáveis y_{ij} . Mais formalmente, considere:

- $F(e)$ a frequência da aresta e entre as soluções do primal lagrangiano;
- F_{min} e F_{max} as frequências mínima e máxima entre as soluções do primal lagrangiano;
- $NF(e)$ a frequência normalizada da aresta e , calculada por: $NF(e) = (F(e) - F_{min}) / (F_{max} - F_{min})$;
- $K(e)$ é a chave aleatória da aresta e em um indivíduo do BRKGA;
- $rand(0, 1)$ é um número aleatório no intervalo $[0, 1)$.

Tradicionalmente, $K(e)$ é um valor aleatório no intervalo $[0, 1)$ com distribuição uniforme. A nova proposta é fazer $K(e) = \frac{rand(0,1)+NF(e)}{2}$ para todas as chaves que representam o indivíduo. Assim, quanto maior a frequência da aresta, maior a probabilidade que a chave aleatória se aproxime de 1. É possível observar que o novo procedimento de geração das chaves aleatórias pode ser combinado com quaisquer funções de aptidão F1, F2 ou F3. Sempre que uma variação do BRKGA utiliza esse procedimento, a referência “RL” estará contida na sua denominação.

4.4 Heurística de Busca Local

Esta seção descreve a heurística de busca local em arborescências, procedimento aplicado nas diferentes versões do BRKGA, bem como nas relaxações lagrangianas. Em linhas gerais, o procedimento de busca local pode ser descrito da seguinte maneira. Considere uma solução do MS-MRP-QoS representada por uma arborescência. A partir de um nó terminal k , atendido pela arborescência, o objetivo consiste em alterar os arcos adjacentes aos terminais diferentes de k , de modo que o *delay* do caminho da raiz s para cada $l \in D \setminus \{k\}$ seja um valor dentro do intervalo $delay(k) \pm \Delta_v$. Essa abordagem visa reduzir a quantidade de terminais que não são atendidos como consequência das restrições de variação de *delay*. A codificação dessa busca local é apresentada no Algoritmo 7.

Os dados de entrada do algoritmo são um grafo $G = (V, A)$ e uma arborescência T computada em G enraizada no nó s . Assume-se que a representação de T é dada por um vetor de predecessores com tamanho $|V|$, ou seja, cada posição desse vetor representa um nó do grafo e o valor contido nessa posição indica o predecessor do nó em T . As linhas (1) e (2) definem três vetores de tamanho $|V|$. L e J mantêm, respectivamente, o *delay* e o *jitter* do caminho de s para cada nó $i \in V \setminus \{s\}$. O vetor A armazena a informação sobre o atendimento ou não dos nós terminais.

Algoritmo 7: Heurística de Busca Local $O(d \cdot m)$

Entrada: G, T

```

1  L, J; // vetores inteiros (delay e jitter)
2  A; // vetor booleano iniciando com Verdadeiro em todas as posições
3  para todo  $k \in D \cup S$  faça
4       $k' \leftarrow k$ ;  $\text{delay}_k \leftarrow 0$ ;  $\text{jitter}_k \leftarrow 0$ ;
      // 0 predecessor da raiz  $s$  é ele mesmo
5      enquanto  $k' \neq s$  faça
6           $\text{delay}_k \leftarrow \text{delay}_k + \text{delay}(T[k'], k')$ ;
7           $\text{jitter}_k \leftarrow \text{jitter}_k + \text{jitter}(T[k'], k')$ ;
8           $k' \leftarrow T[k']$ ;
9       $L[k] \leftarrow \text{delay}_k$ ;  $J[k] \leftarrow \text{jitter}_k$ ;
10     se  $L[k] > \Delta_d$  ou  $J[k] > \Delta_j$  então  $A[k] \leftarrow \text{Falso}$ ;
11 Embaralha  $D$ ;
12  $f \leftarrow -1$ ;
13 para todo  $k \in D$  faça
14     se  $A[k]$  então
15          $f \leftarrow k$ ;
16     Encerra o laço;
17 se  $f = -1$  então retorna  $|D|$ ;
18 para todo  $k \in D \setminus \{f\}$  faça
19     se  $L[k] < (L[f] - \Delta_v)$  ou  $L[k] > (L[f] + \Delta_v)$  então
20          $m \leftarrow -1$ ;
21         para todo  $(i, k) \in \delta_k^+$  em  $T$  faça
22             se  $i \neq T[k]$  e  $k \neq T[i]$  então
23                  $d \leftarrow \text{delay}(i, k)$ ;  $j \leftarrow \text{jitter}(i, k)$ ;
24                 se  $L[i] + d \geq (L[f] - \Delta_v)$  e  $L[i] + d \leq (L[f] + \Delta_v)$  e  $L[i] + d \leq \Delta_d$ 
25                     e  $J[i] + j \leq \Delta_j$  então
26                      $m \leftarrow i$ ;
27                     Encerra o laço;
28             se  $m \neq -1$  então
29                  $\text{dif}_{\text{delay}} \leftarrow d - L[k]$ ;  $\text{dif}_{\text{jitter}} \leftarrow j - J[k]$ ;
30                  $T, L, J, A \leftarrow \text{AvaliaMudança}(T, L, J, A, f, k, m, \text{dif}_{\text{delay}}, \text{dif}_{\text{jitter}})$ ;
31 não-atendidos  $\leftarrow 0$ ;  $D' \leftarrow \emptyset$ ;
32 para todo  $k \in D$  faça
33     se  $A[k] = \text{Falso}$  então não-atendidos  $\leftarrow$  não-atendidos  $+ 1$ ;
34     senão  $D' \leftarrow D' \cup \{k\}$ ;
35 retorna não-atendidos  $+ (|D| - \text{SeleçãoPorVariação}(D'))$ ;

```

O primeiro laço, nas linhas (3)-(10), trata-se da inicialização dos valores de *delay* e *jitter* nos caminhos de T , juntamente da verificação de quais caminhos excedem pelo menos uma das métricas. Para que a ordem de visita dos nós não seja idêntica em todas as chamadas da busca local, é utilizada uma função, como apresentado na linha (11), para embaralhar o conjunto de terminais D . Logo em seguida, na linha (13), seleciona-se um terminal atendido f , cujo valor $L[f]$ será utilizado como o pivô para aproximação dos

outros terminais. Caso nenhum terminal, mesmo sem aplicar as restrições de variação de *delay*, seja atendido em T , o algoritmo encerra retornando o valor $|D|$, como mostrado na linha (17).

O procedimento de busca local tem início no laço da linha (18), de modo que visita-se todo terminal k diferente de f , caso o custo de *delay* no atual caminho para visitar k esteja fora do intervalo $L[f] \pm \Delta_v$. Nesse caso, deve-se verificar os nós incidentes em k no grafo G , ou seja, todo arco $(i, k) \in \delta_k^+(G)$, como descrito na linha (21). A linha (22) garante que ao mudar o predecessor de k , as propriedades da arborescência se manterão. Assim, ao detectar uma alteração viável, caso ela exista, o algoritmo passa para a etapa de análise dos valores de *delay*, *jitter* e variação para o novo caminho. O procedimento total do laço de busca utiliza uma decisão baseada em *First Improving*, ou seja, a busca em uma vizinhança encerra-se na primeira alteração encontrada que melhore a solução. Na linha (27) verifica-se a existência de uma alteração viável para o predecessor de k e, caso não exista, a iteração do laço passa para o próximo terminal. Se existir movimento viável, chama-se a função *AvaliaMudanca*, descrita no Algoritmo 8. Por fim, as linhas (30)-(34) contam a quantidade terminais não atendidos.

O Algoritmo 8 propaga as alterações de *delay* e *jitter* para todo nó $j \in T$, cujo caminho utilizado para visitar j passa pelo nó k . Avalia-se a troca do predecessor de k como vantajosa se a alteração não fizer com que dois ou mais terminais deixem de ser atendidos por consequência. As linhas (2)-(4) fazem as inicializações, criando também a variável que contabiliza o número de terminais não atendidos. Por fim, adiciona-se k ao conjunto C para efetuar um procedimento de propagação das alterações de valores para os nós descendentes de k .

Algoritmo 8: Avaliação de Mudança de Predecessor $O(m)$

Entrada: $T, L, J, A, f, k, m, dif_{delay}, dif_{jitter}$

```

1  $T[k] \leftarrow m, L[k] \leftarrow d, J[k] \leftarrow j$ ;
2 perdidos  $\leftarrow 0$ ;
3  $A[k] \leftarrow \text{Verdadeiro}$ ;
4  $C \leftarrow \{k\}$ ;
5 enquanto  $C \neq \emptyset$  faça
6    $k' \leftarrow$  último elemento de  $C$ ;
7    $C \setminus \{k'\}$ ;
8   para todo  $i \in \delta_{k'}^-(T)$  faça
9      $L[i] \leftarrow L[i] + dif_{delay}, J[i] \leftarrow J[i] + dif_{jitter}$ ;
10    se  $L[i] > \Delta_d$  ou  $J[i] > \Delta_j$  então
11      se  $i \in D$  e  $A[i] = \text{Verdadeiro}$  então
12         $A[i] \leftarrow \text{Falso}$ ;
13        perdidos  $\leftarrow$  perdidos + 1;
14      senão  $A[i] \leftarrow \text{Verdadeiro}$ 
15    se  $perdidos \geq 2$  ou  $(i = f \text{ e } A[f] = \text{Falso})$  então
16      Desfaz todas as alterações nos vetores de entrada;
17      retorna  $L, J, T, A$ ;
18    senão  $C \leftarrow C \cup \{i\}$ ;
19 retorna  $L, J, T, A$ ;

```

O laço principal do Algoritmo 8 inicia na linha (5), que se repete enquanto ainda houverem nós a serem atualizados. O primeiro passo consiste na seleção de k' , o último nó adicionado a C . Após a escolha, como mostrado na linha (8), visita-se todo i tal que $(k', i) \in T'$, computa-se o novo valor de *delay* e *jitter* para o nó i e a linha (10) verifica se esses valores excedem alguma das métricas. A linha (15) verifica as duas condições de parada onde o retorno do algoritmo deve ser Falso. A primeira condição é se a alteração do predecessor de k fez com que dois ou mais terminais não fossem mais atendidos; e a segunda condição é se a alteração influencia no valor de *delay* (*jitter*) do terminal f a ponto de torná-lo não atendido. Caso nenhuma dessas condições torne-se verdadeira, o procedimento prossegue adicionando os novos nós visitados à C .

Um exemplo, passo a passo, do funcionamento da heurística de busca local é ilustrado na Figura 4.1. Em 4.1a apresenta-se o grafo da instância, sendo o s , com círculo duplo, a raiz, os nós representados por retângulos são terminais e os círculos representam nós opcionais. Por simplicidade, assuma que os valores contidos nos arcos representam apenas o *delay*. Assim, considerando $\Delta_d = 20$ e $\Delta_v = 10$, foi gerada a arborescência T , apresentada na Figura 4.1b, que atende apenas o terminal b , representado por um retângulo duplo. Assim, com a aplicação da busca local apresentada, fixa-se o nó b como pivô para aproximação do valor de *delay* dos demais terminais, dado que b é o único atendido. Na Figura 4.1c é apresentada a verificação da troca de predecessor no nó c , representada pelos arcos tracejados. Com isso, o valor de $L[c]$, que inicialmente é 40, pode ser reduzido para 15 com a alteração do arco que o atende de (b, c) para (f, c) .

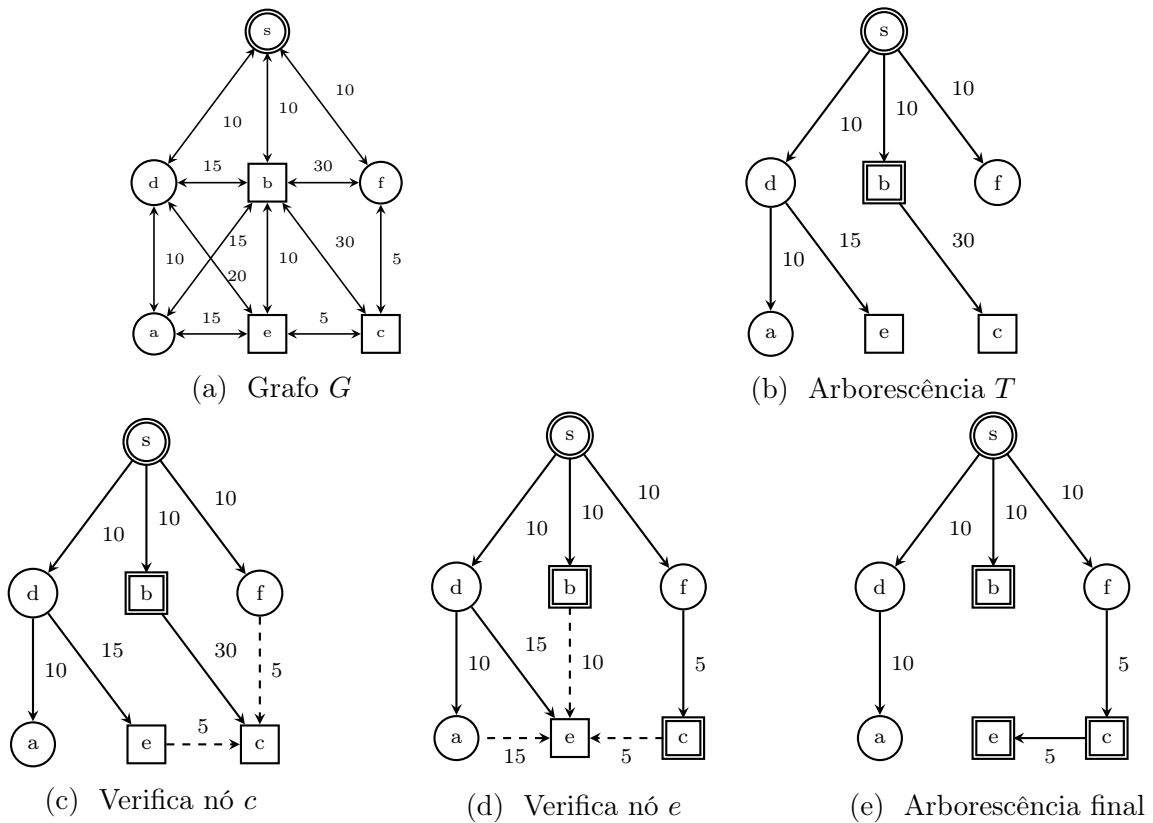


Figura 4.1: Exemplo de aplicação da busca local

Com a alteração do predecessor de c validada, repete-se o processo de verificação e

busca para o terminal e , como apresentado na Figura 4.1d. Para este caso, existem três possibilidades, mas apenas duas são viáveis. Alterar o predecessor de e para b ou c faz com que o valor $L[e]$ seja reduzido de 25 para 20, tornando-o atendido em *delay* e variação. Por fim, assumindo a seleção do arco (c, e) , temos como resultado a Figura 4.1e que apresenta a arborescência final do procedimento de busca local, atendendo os três terminais.

4.5 Análise Estatística de Resultados

Neste trabalho, utilizaram-se testes estatísticos não paramétricos para análise de múltiplas instâncias. O objetivo da utilização desses testes é fornecer um embasamento científico mais rígido às comparações feitas, validando se a diferença entre duas ou mais estratégias é estatisticamente significativa ou não. O trabalho de Demšar [12] sugere uma metodologia para fazer tais comparações no contexto de aprendizado de máquina, no qual utilizam-se diversos testes, de modo a identificar existência de diferença estatisticamente significativa (DES) entre vários classificadores em múltiplos conjuntos de dados. Nesta seção, serão brevemente discutidos os testes e a forma de interpretação de seus resultados.

Para todos os testes foi considerado um nível de confiança de 95% com a hipótese nula de que não existe DES entre as amostras. Todos os testes se baseiam em *ranks*, de modo que a abordagem que detém *rank* k é aquela contendo o k -ésimo melhor resultado para uma instância.

O primeiro teste descrito é o de Iman-Davenport, utilizado para identificar existência de DES entre múltiplas abordagens. Inicialmente precisa-se computar o valor de *rank* médio R_i para cada abordagem. O cálculo é feito da seguinte forma: seja r_k^i o *rank* da k -ésima abordagem para a i -ésima instância (com *rank* médio sendo utilizado em caso de empates). O valor R_i para a i -ésima metodologia é dado por $R_i = \frac{1}{N} \cdot \sum_k r_k^i, \forall k \in \{1, \dots, k\}$, tal que k é o número de abordagens e N o número de instâncias.

O valor do teste de Iman-Davenport é calculado de acordo com as equações (4.15) e (4.16).

$$x_F^2 = \frac{12 \cdot N}{k \cdot (k+1)} \left[\sum_i R_i^2 - \frac{k \cdot (k+1)^2}{4} \right] \quad (4.15)$$

$$F_F = \frac{(N-1) \cdot x_F^2}{N \cdot (k-1) - x_F^2} \quad (4.16)$$

O valor do teste calculado para F_F segue a distribuição F , a qual tem como parâmetros os graus de liberdade, $k-1$ e $(k-1) \cdot (N-1)$. Logo, o valor crítico pode ser obtido seguindo tal distribuição. Ao final, se o valor F_F encontrado pelo teste for maior que o valor crítico, pode-se rejeitar a hipótese nula.

Caso a hipótese nula seja rejeitada, inicia-se a aplicação de um teste *post-hoc* para encontrar quais das heurísticas de fato diferem entre si. Para tal, utiliza-se o teste de Nemenyi, que calcula a diferença crítica e identifica grupos de abordagens que são estatisticamente equivalentes entre si. Os resultados de duas abordagens distintas são ditos diferentes, segundo o teste de Nemenyi, se seus *ranks* médios se diferem por pelo menos a diferença crítica (DC) obtida segundo a equação (4.17).

$$DC = q_\alpha \cdot \sqrt{\frac{l \cdot (k + 1)}{6 \cdot N}} \quad (4.17)$$

Na equação (4.17), o α representa o nível de confiança, e o valor crítico q_α é definido segundo a estatística de *Studentized range* dividida por $\sqrt{2}$. Os valores de tal estatística podem ser encontrados no artigo de Deñsar [12]. Ainda, o resultado do teste de Nemenyi pode ser representado graficamente, como ilustrado na Figura 4.2. O eixo horizontal equivale aos valores de *rank* médio, em ordem crescente da esquerda para a direita. Cada abordagem é representada por uma linha vertical com seu respectivo nome e valor de *rank* médio logo abaixo. Uma linha horizontal de comprimento equivalente ao valor da DC é posicionada acima do eixo horizontal principal. Abordagens nas quais o valor de *rank* médio não diferem mais do que a DC são conectadas por um segmento horizontal em negrito, de comprimento sempre inferior ao valor da DC.

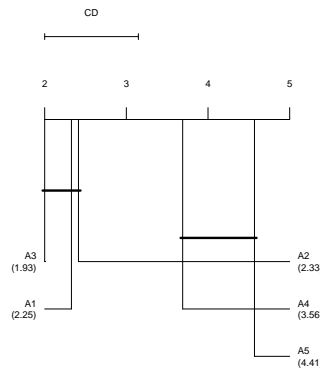


Figura 4.2: Exemplo de representação gráfica do teste Nemenyi.

Observa-se que as abordagens fictícias A3, A1 e A2 formam o grupo de menor valor de *rank* médio, e consequentemente, são estatisticamente equivalentes entre si de acordo com o teste de Nemenyi. Portanto, podemos definir como melhor algoritmo aquele que apresenta o maior número de vitórias, i.e., o número de vezes que cada abordagem foi melhor que as demais.

Outro teste utilizado neste trabalho foi o de Wilcoxon, aplicado quando deseja-se verificar se existe DES entre os resultados de dois métodos, sendo mais poderoso que o teste de Iman-Davenport para esse cenário. O teste de Wilcoxon computa os *rank*s das diferenças entre os resultados dos dois métodos ignorando o sinal e posteriormente compara os *rank*s das diferenças positivas e negativas.

Para computar o teste de Wilcoxon, considere $d_i, i \in \{1, \dots, N\}$, a diferença entre os resultados das duas abordagens na i -ésima instância, r_{d_i} o *rank* do valor absoluto da i -ésima diferença (*rank* médio é atribuído no caso de empate). Assuma R^+ a soma dos *rank*s para as instâncias nas quais a abordagem fictícia A foi melhor do que B ($d_i > 0$) e R^- a soma dos *rank*s para o caso oposto ($d_i < 0$). Os *rank*s para $d_i = 0$ são separados

igualmente entre as somas, como mostrado na Equação (4.18).

$$R^+ = \sum_{d_i > 0} r_{d_i} + \frac{1}{2} \sum_{d_i = 0} \quad R^- = \sum_{d_i < 0} r_{d_i} + \frac{1}{2} \sum_{d_i = 0} \quad (4.18)$$

Seja $T = \min\{R^+, R^-\}$. Segundo Demšar [12], nos casos onde $N \leq 25$, o valor crítico para o teste de Wilcoxon pode ser consultado na maioria dos livros de estatística, como por exemplo, o de Devore [13]. Uma vez que tal valor é calculado, rejeita-se a hipótese nula se o valor de T for menor ou igual ao valor crítico. No caso em que $N > 25$, computa-se a estatística z como apresentado na Equação (4.19). Segundo Demšar, considerando um nível de confiança de 95%, a hipótese nula pode ser rejeitada quando $z < -1.96$.

$$z = \frac{T - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}} \quad (4.19)$$

4.6 Geração de Instâncias

Nos trabalhos presentes na literatura que envolvem o problema de roteamento aplicado a redes veiculares, não há um consenso entre os autores em relação às instâncias utilizadas em experimentos. Cada autor criou as próprias instâncias para validação do algoritmo ou protocolo proposto, como pode-se observar nos trabalhos de [6, 15, 34]. A geração de instâncias descrita nesta seção é fortemente baseada no trabalho de Ribeiro et al. [34], que descreve uma maneira coerente de criar instâncias para avaliação de algoritmos de roteamento em VANET's utilizando modelos baseados em simulação de tráfego. Nessa simulação é possível visualizar os movimentos dos veículos que são gerados, além de poder exportar os dados de mobilidade, ou seja, o posicionamento, velocidade e direção de um veículo em determinado instante. Os dados exportados do modelo de mobilidade podem ser introduzidos em um simulador específico para redes de comunicação, como NS-2 [22], NS-3 [35] e OMNet [44].

Para a simulação de tráfego e mobilidade veicular tratada neste trabalho, foi escolhida a biblioteca SUMO (*Simulation of Urban MObility*), que é uma ferramenta amplamente utilizada tanto no meio acadêmico quanto empresarial. O SUMO é acoplado a uma ferramenta externa de simulação de redes, neste caso NS-2, para fornecer dados realistas de comunicação dos veículos. O NS-2 é um simulador de eventos discretos direcionados à pesquisa de rede, inicialmente desenvolvido pela Universidade de Berkeley e altamente utilizado pela comunidade científica. O NS-2 fornece um apoio substancial para a simulação do funcionamento de transmissões, roteamento e protocolos de redes, com e sem fio. É importante utilizar o modelo de mobilidade gerado pelos simuladores de veículos para que o posicionamento dos nós esteja o mais próximo possível da realidade.

O primeiro passo na geração de instâncias para o MS-MRP-QoS é a criação de estradas e rodovias. Neste trabalho foram adotados como base os mapas disponíveis no projeto *Open Street Maps* (OSM) [19], que é um projeto de mapeamento colaborativo para criar um mapa livre e editável do mundo e está em desenvolvimento desde 2004. A Figura 4.3 apresenta uma parte do mapa da cidade de Washington D. C., medindo aproximadamente 250.000 metros quadrados.



Figura 4.3: Parte do mapa da cidade de Washington no *Open Street Maps*

O passo seguinte é tratar e selecionar as informações desejadas com o OSM, ou seja, o recorte do mapa que se deseja utilizar. Os dados são convertidos e importados para o SUMO gerando assim o modelo de mobilidade, inserindo veículos no mapa, bem como semáforos e outros componentes inerentes ao tráfego. Após essas inserções, inicia-se o processo de simulação do tráfego veicular, exportando os dados de movimentação dos veículos para utilização na etapa posterior. A Figura 4.4 apresenta uma parte ampliada do mapa de Washington, ressaltando dados de tráfego e veículos.

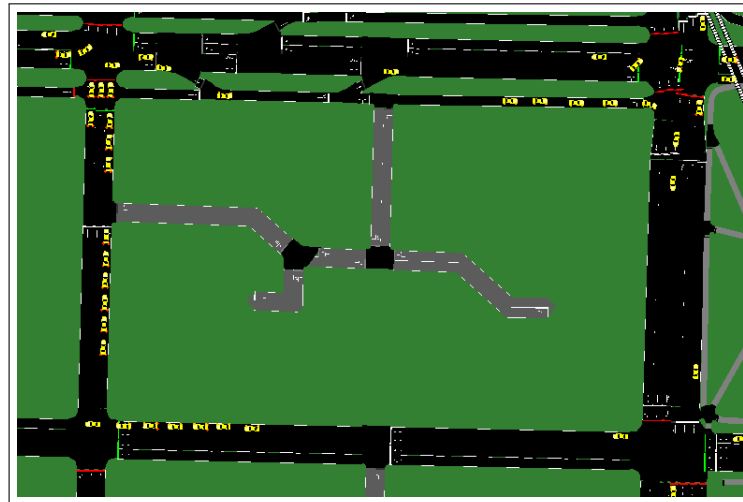


Figura 4.4: Parte do mapa ampliado da cidade de Washington no SUMO

Após a exportação dos dados vindos do SUMO, será feita a utilização do simulador de rede NS-2. O processo de simulação de rede é então iniciado e, durante o mesmo, é possível fotografar um momento qualquer e com isso obter a topologia da rede como um todo, representando-a como um grafo. O valor das métricas em cada *link* é medido com base na troca de mensagem entre os pares de veículos. Considera-se o cálculo médio em um intervalo de 15 segundos. Assim é possível ter uma instância próxima da realidade,

dentro das limitações do processo de simulação.

A Figura 4.5 ilustra o grafo da instância final gerada a partir de um intervalo de tempo da simulação. Os nós não seguem o posicionamento geográfico real, apenas exibem as conexões existentes naquele intervalo entre os 10 veículos da rede. O nó azul representa a origem, os nós vermelhos representam os destinos, ambos decididos aleatoriamente. Os nós pretos são nós de Steiner, que podem ser utilizados ou não no processo de construção da árvore *multicast*.

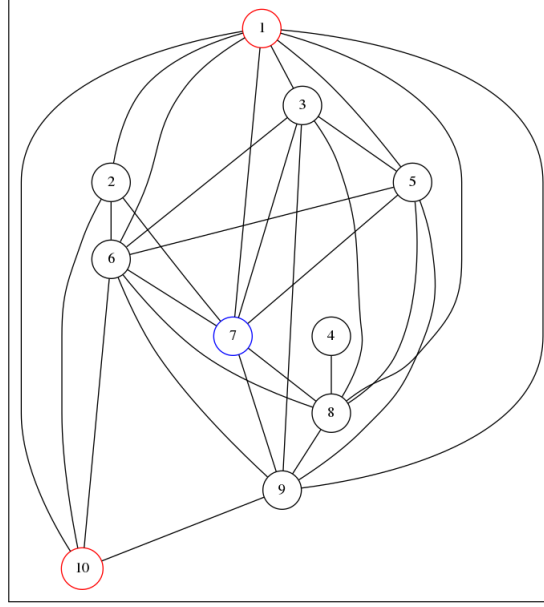


Figura 4.5: Fotografia dos enlaces da rede.

Outros parâmetros da instância são os valores máximos de *delay* e *jitter* permitidos no caminho da raiz até cada um dos terminais, o valor máximo da variação de *delay* entre o nó origem e qualquer par de destinos e o valor mínimo de largura de banda dos arcos. Foram gerados diferentes valores de parâmetros visando estressar as metodologias na etapa de avaliação dos algoritmos. Para tal, foram geradas duas árvores de caminhos mínimos do nó origem para todos os nós destino. A primeira é calculada utilizando como custo o *delay* dos arcos e a segunda o *jitter*. Essas árvores são referenciadas como T^d e T^j , respectivamente.

Seja λ^* o maior valor de *delay* entre os caminhos da árvore T^d . A partir da árvore T^j , é possível inverter os custos de *jitter* dos arcos pelos de *delay* e assim obter $\tilde{\lambda}$, que é o maior valor de *delay* nos caminhos da árvore de *jitter*, ou seja, T^j . Seguindo o mesmo procedimento, obtém-se os valores ξ^* e $\tilde{\xi}$, sendo respectivamente o maior valor de *jitter* da árvore T^j e da árvore T^d .

Para a geração das instâncias, os parâmetros foram configurados como: $\Delta_d = \tilde{\lambda}$ e $\Delta_j = \tilde{\xi}$. O valor do mínimo de largura de banda necessária para o link foi fixado em 200, utilizando um parâmetro aplicado diretamente ao simulador de rede. Por fim, assumindo $\lambda^1, \dots, \lambda^{|D|}$ como os valores de *delay* dos caminhos mínimos para cada terminal na árvore T^d , obtém-se o valor de Δ_v igual ao desvio padrão da diferença absoluta entre todos os pares de *delays* em $\lambda^1, \dots, \lambda^{|D|}$.

A Figura 4.6, retirada de [34], apresenta um fluxograma simplificado do processo de geração das instâncias, começando pelo processo de extração do mapa, passando pela simulação de tráfego usando o SUMO, juntamente com a criação do modelo de mobilidade e terminando no processo de simulação no NS-2, onde a instância final é gerada. O conjunto de testes desenvolvido neste trabalho é formado por 40 instâncias criadas a partir de um trecho real do mapa da cidade de Washington. Para 30 instâncias o número de nós varia de 10 a 100, o número de terminais de 4 a 52 e o número de arcos de 18 a 4416. Para 10 outras instâncias, o número de nós varia de 125 a 350, o número de terminais de 55 a 170 e o número de arcos de 2448 a 24446.

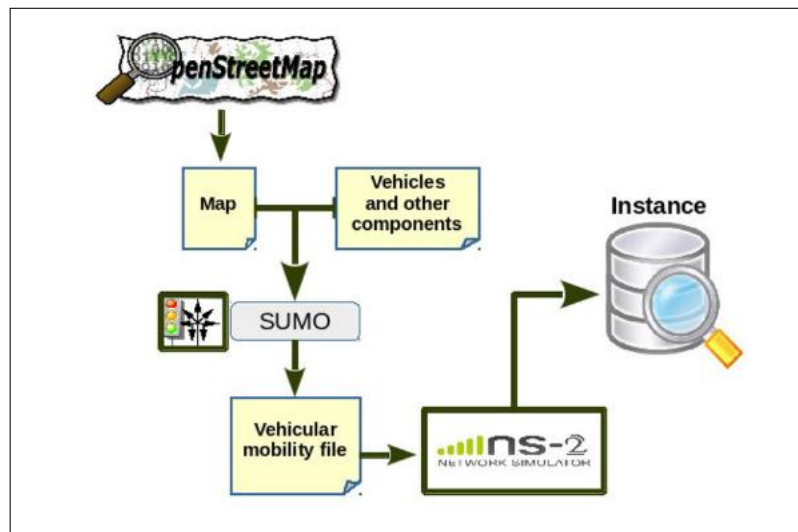


Figura 4.6: Fluxograma do processo de geração das instâncias.

Capítulo 5

Resultados Computacionais

Neste capítulo são descritos os experimentos realizados para avaliar a eficácia das diversas abordagens propostas nesta dissertação. Este capítulo está organizado da seguinte maneira: a Seção 5.1 contém as informações sobre os pré-processamentos, seguida da Seção 5.2, que apresenta os resultados dos modelos de PLI e PLIM desenvolvidos.

As estratégias apresentadas foram implementadas em C++, os programas compilados com GCC versão 7.5.0, usando o nível de otimização -O3. Os experimentos foram realizados em um processador Intel(R) Xeon(R) CPU E5-2630 v4 com 10 núcleos de 2.20 GHz cada e 64GB de memória RAM. A execução ocorreu em um sistema operacional Linux Ubuntu 18.04 de 64 bits. Os experimentos foram realizados sob um conjunto de 40 instâncias geradas e disponibilizadas em <https://github.com/cvaraujo/QoS-MRP-Instances>.

5.1 Pré-processamentos

Esta seção apresenta os resultados relacionados as fixações de variáveis com a aplicação dos pré-processamentos apresentados na Seção 4.1. Para computar problemas de SP e SPRC utilizaram-se as implementações fornecidas pela biblioteca Boost C++¹. Os resultados encontram-se na Tabela 5.1. A primeira coluna contém a identificação da instância, as três colunas seguintes apresentam, respectivamente, o número de nós do grafo ($|V|$), número de nós terminais ($|D|$) e quantidade de arcos ($|A|$). A coluna “MVE” informa a quantidade de nós que podem ser removidos da instância pelo processamento em questão. A mesma lógica se aplica às colunas “MAE” e “SAE”, que indicam a quantidade de arcos removidos e a quantidade de variáveis $f_{ij}^k = 0$ para todo $k \in D \cup S$ e $(i, j) \in A$, respectivamente. A coluna “Rem. SAE (%)” apresenta, em porcentagens, os valores da coluna “SAE”. As colunas “Rem. $|V|$ ”, “Rem. $|A|$ ” e “Rem. $|D|$ ” correspondem, respectivamente, aos nós, arcos e terminais removidos do grafo após aplicação dos três pré-processamentos para a instância. Por fim, a coluna “Tempo” apresenta o tempo somado dos três procedimentos.

¹<https://www.boost.org/>

Instância	V	D	A	MVE	MAE	SAE	Rem. SAE (%)	Rem. V	Rem. A	Rem. D	Tempo
washington-50-10-6	10	6	46	1	11	156	33,9	3	20	0	0
washington-50-20-11	20	11	184	0	33	1508	41,0	2	49	0	0
washington-50-30-15	30	15	420	2	82	4925	39,1	5	128	0	0
washington-50-40-23	40	23	830	1	184	17015	51,3	4	230	0	1
washington-50-50-28	50	28	1248	6	321	31422	50,4	10	527	0	2
washington-50-60-35	60	35	1812	9	412	56068	51,6	13	823	0	4
washington-50-70-37	70	37	2516	2	707	86755	49,3	6	803	0	21
washington-50-80-39	80	39	3592	12	1051	126159	43,9	15	1883	0	32
washington-50-90-51	90	51	4218	14	1049	198888	52,4	19	1775	0	34
washington-50-100-45	100	45	4416	31	1025	180369	40,8	33	2297	0	44
washington-75-10-4	10	4	32	0	7	89	27,8	0	7	0	0
washington-75-20-12	20	12	126	1	12	923	36,6	2	24	0	0
washington-75-30-16	30	16	318	0	39	4190	43,9	1	44	0	0
washington-75-40-21	40	21	538	2	40	9892	46,0	6	135	3	1
washington-75-50-30	50	30	876	0	212	22354	51,0	4	259	0	3
washington-75-60-25	60	25	1256	3	384	27572	36,6	11	539	0	3
washington-75-70-42	70	42	1700	2	339	63943	53,7	5	470	0	17
washington-75-80-48	80	48	2504	3	490	106488	53,2	11	825	0	30
washington-75-90-47	90	47	3034	2	942	132330	48,5	8	1182	0	74
washington-75-100-52	100	52	3792	4	1179	184042	48,5	6	1455	0	86
washington-100-10-6	10	6	18	4	0	30	16,7	5	4	1	0
washington-100-20-10	20	10	98	2	5	651	33,2	3	10	0	0
washington-100-30-12	30	12	206	8	19	1466	23,7	9	46	1	0
washington-100-40-18	40	21	422	1	29	5515	32,7	2	49	0	1
washington-100-50-27	50	27	596	3	94	10405	34,9	8	154	1	2
washington-100-60-34	60	34	854	2	99	26131	51,0	9	305	5	5
washington-100-70-39	70	39	1204	3	276	37123	44,0	14	471	2	7
washington-100-80-32	80	32	1764	3	442	48638	34,5	8	587	0	10
washington-100-90-43	90	43	2302	6	713	84985	41,0	15	1021	0	19
washington-100-100-43	100	43	2816	0	950	108877	38,7	10	1086	1	47
washington-200-125-55	125	55	2448	31	580	108619	35,5	35	1362	0	34
washington-200-150-61	150	61	4468	21	1000	240997	36,0	26	1694	0	239
washington-200-175-74	175	74	6048	13	2099	381693	36,1	22	2912	0	584
washington-200-200-114	200	114	7290	19	1743	756030	51,9	20	2690	0	1173
washington-200-225-135	225	135	8928	0	1330	1140034	56,8	7	1479	2	1960
washington-200-250-109	250	109	12806	7	4142	1325105	41,4	10	4444	0	1511
washington-200-275-146	275	146	14390	12	5005	1936238	48,9	19	5574	1	2325
washington-200-300-136	300	136	17418	11	6309	2185292	41,8	21	6992	0	3537
washington-200-325-170	325	170	18520	34	6206	2977336	49,5	42	7453	0	3373
washington-200-350-150	350	150	24436	5	8537	3497422	40,9	12	8864	0	5844

Tabela 5.1: Variáveis removidas pelos pré-processamentos.

Diante dos resultados apresentados na Tabela 5.1, percebe-se uma redução considerável, especialmente das variáveis f_{ij}^k , chegando a fixar até 56,8% delas. O número total de nós removidos em cada instância considera o efeito da aplicação dos três pré-processamentos. Para o procedimento MVE, a remoção de um nó juntamente com todos os arcos adjacentes pode ocasionar a desconexão de outros nós do grafo. A mesma ideia aplica-se para remoção de arcos utilizando MAE. A remoção de nós terminais é a mais relevante, dado que influencia diretamente no limitante dual do MS-MRP-QoS. Das 40 instâncias, em apenas 9 foi possível pré-fixar um subconjunto de terminais como não atendidos.

O tempo de execução necessário para computar os procedimentos MVE e MAE é muito baixo, sendo menor que um segundo para as 30 instâncias que contêm até 100 nós. Para as 10 instâncias contendo o maior número de nós, o tempo de execução não excedeu 10 segundos. O pré-processamento que demanda mais tempo é o SAE, executando, em alguns poucos casos, um tempo próximo a 90 minutos. Para 33 instâncias o tempo total do pré-processamento não excedeu 10 minutos. Para as outras 7 instâncias, esse tempo excedeu 20 minutos. Ainda assim, vale ressaltar que basta computar uma vez os pré-processamentos para cada instância, permitindo o armazenamento, por exemplo, em arquivos de texto contendo os índices das variáveis que foram fixadas por cada procedimento. Esses arquivos

estão disponíveis juntamente das instâncias no endereço <https://github.com/cvaraujo/QoS-MRP-Instances>.

5.2 Modelos DMFM-MRP e AB-MRP

Para implementação dos modelos DMFM-MRP e AB-MRP, foi utilizado o resolvidor Gurobi versão 9.0.3 mantendo a configuração padrão de seus parâmetros. O tempo máximo para resolução de cada instância, em ambos os modelos, foi pré-definido com o valor de uma hora (3600 segundos) e não houve limitação de consumo de memória.

Os modelos nesta seção referenciam-se pelo valor das constantes M_d e M_j utilizados nas restrições de *delay* e *jitter* (2.7 e 2.8). O modelo DMFM-MRP apresenta resultados de duas variações, a primeira utilizando o valor dos parâmetros $M_d = \Lambda$ e $M_j = \Xi$, onde Λ e Ξ são calculados como descrito na Seção 2.3.1, e portanto será referenciada como DMFM-MRP $_{\Lambda, \Xi}$. A segunda variação utiliza $M_d = M_j = 1$ e, conseqüentemente, será denominada DMFM-MRP $_{1,1}$. A formulação AB-MRP segue o mesmo padrão de diferenciação pelo valor dos parâmetros M_d e M_j , mas não foi possível obter resultados confiáveis na versão AB-MRP $_{\Lambda, \Xi}$ pelo fato do valor dos parâmetros ocasionar instabilidades numéricas associados às variáveis reais do modelo. Portanto, são reportados apenas os resultados da versão AB-MRP $_{1,1}$.

As Tabelas 5.2, 5.3, 5.4 e 5.5 contém os resultados dos modelos DMFM-MRP $_{\Lambda, \Xi}$, DMFM-MRP $_{1,1}$, DMFM-MRP $_{1,1}$ sem utilização dos pré-processamentos e AB-MRP $_{1,1}$, respectivamente. As primeiras quatro colunas contém o nome da instância, a quantidade de nós ($|V|$), o número de nós terminais ($|D|$) e a quantidade de arcos ($|A|$). As colunas “LI” e “LS” apresentam os valores de Limitante Inferior (LI) e Limitante Superior (LS) para cada instância. Caso uma determinada célula da tabela contenha um hífen (–) significa que o resolvidor não retornou valor ao final da execução. A coluna “gap%” apresenta o gap relativo percentual entre o LI e o LS². A coluna seguinte, “gap % LS”, representa o gap percentual do LS computado pelo modelo em relação ao número total de terminais em $|D|$ ³. A coluna “Nós GRB” contém a quantidade de nós explorados da árvore de enumeração (árvore de *branch-and-bound*) do Gurobi. A última coluna dispõe do tempo de execução em segundos. Os valores “TLE” representam Tempo Limite Excedido. As linhas ressaltadas em cor cinza escura (■) representam a instâncias nas quais o modelo retornou solução ótima, enquanto as linhas colocadas em cinza claro (■) são instâncias onde obteve-se solução viável, mas o gap para o limitante dual não foi fechado.

² $\frac{LS-LI}{LS} \times 100$
³ $100 - \frac{|D|-LS}{|D|} * 100$

Instância	V	D	A	LI	LS	gap (%)	gap (%) LS	Nós GRB	Tempo
washington-50-10-6	10	6	46	1	1	0	17	0	0
washington-50-20-11	20	11	184	4	4	0	36	1	0
washington-50-30-15	30	15	420	3	3	0	20	1	2
washington-50-40-23	40	23	830	6	6	0	26	2440	110
washington-50-50-28	50	28	1248	13	13	0	46	442	21
washington-50-60-35	60	35	1812	15	15	0	43	3766	85
washington-50-70-37	70	37	2516	3	-	-	-	1	TLE
washington-50-80-39	80	39	3592	7	28	75	72	1	TLE
washington-50-90-51	90	51	4218	13	37	65	73	1	TLE
washington-50-100-45	100	45	4416	8	31	74	69	1	TLE
washington-75-10-4	10	4	32	3	3	0	75	1	0
washington-75-20-12	20	12	126	4	4	0	33	761	1
washington-75-30-16	30	16	318	5	5	0	31	5244	40
washington-75-40-21	40	21	538	5	5	0	24	1	5
washington-75-50-30	50	30	876	9	9	0	30	440178	2259
washington-75-60-25	60	25	1256	11	11	0	44	152	31
washington-75-70-42	70	42	1700	6	12	50	29	1	TLE
washington-75-80-48	80	48	2504	2	-	-	-	1	TLE
washington-75-90-47	90	47	3034	2	-	-	-	1	TLE
washington-75-100-52	100	52	3792	4	-	-	-	1	TLE
washington-100-10-6	10	6	18	2	2	0	33	0	0
washington-100-20-10	20	10	98	2	2	0	20	1	0
washington-100-30-12	30	12	206	2	2	0	17	5407	9
washington-100-40-18	40	21	422	4	4	0	19	38322	354
washington-100-50-27	50	27	596	3	9	67	33	59737	TLE
washington-100-60-34	60	34	854	10	10	0	29	1	17
washington-100-70-39	70	39	1204	17	17	0	44	27009	1636
washington-100-80-32	80	32	1764	5	5	0	16	1	78
washington-100-90-43	90	43	2302	11	11	0	26	4686	196
washington-100-100-43	100	43	2816	2	-	-	-	1	TLE
washington-200-125-55	125	55	2448	21	29	28	53	1	TLE
washington-200-150-61	150	61	4468	2	61	97	100	1	TLE
washington-200-175-74	175	74	6048	11	-	-	-	1	TLE
washington-200-200-114	200	114	7290	8	114	93	100	1	TLE
washington-200-225-135	225	135	8928	2	-	-	-	0	TLE
washington-200-250-109	250	109	12806	3	-	-	-	0	TLE
washington-200-275-146	275	146	14390	4	-	-	-	0	TLE
washington-200-300-136	300	136	17418	4	-	-	-	0	TLE
washington-200-325-170	325	170	18520	-	-	-	-	-	TLE
washington-200-350-150	350	150	24436	-	-	-	-	-	TLE

Tabela 5.2: Resultados do modelo DMFM-MRP $_{\Lambda, \Xi}$.

Considerando os dados constantes na Tabela 5.2, percebe-se que o modelo DMFM-MRP $_{\Lambda, \Xi}$ resolveu de maneira ótima 20 instâncias, em maioria contendo entre 10 e 60 nós, com exceção da instância “washington-100-50-27”, e 3 instâncias contendo entre 70 e 90 nós. A média de tempo, para as 20 instâncias nas quais obteve resultado ótimo, foi de 242 segundos. Para outras 8 instâncias o modelo obteve soluções viáveis, mas não foi possível fechar o gap de otimalidade dessas 8 instâncias, em apenas uma o número de nós da árvore de enumeração do Gurobi foi maior que 1, ou seja, o modelo tende a consumir todo o tempo disponível na raiz da árvore de enumeração. Para as 12 últimas instâncias, o modelo não foi capaz de retornar uma solução viável. Para as duas maiores instâncias do conjunto, “washington-200-325-170” e “washington-200-350-150”, o modelo não retornou nem mesmo valores de LI, destacando a dificuldade até mesmo para etapa de resolução da relaxação linear na raiz da árvore de enumeração.

Nos resultados do DMFM-MRP $_{\Lambda, \Xi}$, notou-se a dificuldade na obtenção de soluções viáveis de acordo com o aumento de nós (consequentemente de arcos) da instância. Foi pos-

sível observar a alta quantidade de variáveis e restrições carregadas em memória, mesmo com a utilização dos pré-processamentos, fazendo com que o resolvidor tenha dificuldade em computar até mesmo o primeiro nó da árvore de enumeração.

Instância	V	D	A	LI	LS	gap	gap LS	Nós GRB	Tempo
washington-50-10-6	10	6	46	1	1	0	17	0	0
washington-50-20-11	20	11	184	4	4	0	36	1	0
washington-50-30-15	30	15	420	3	3	0	20	1	0
washington-50-40-23	40	23	830	6	6	0	26	1	3
washington-50-50-28	50	28	1248	13	13	0	46	1	3
washington-50-60-35	60	35	1812	15	15	0	43	1	4
washington-50-70-37	70	37	2516	16	16	0	43	14	41
washington-50-80-39	80	39	3592	26	26	0	67	1	25
washington-50-90-51	90	51	4218	35	35	0	69	839	285
washington-50-100-45	100	45	4416	28	28	0	62	2433	71
washington-75-10-4	10	4	32	3	3	0	75	0	0
washington-75-20-12	20	12	126	4	4	0	33	1	0
washington-75-30-16	30	16	318	5	5	0	31	249	6
washington-75-40-21	40	21	538	5	5	0	24	1	1
washington-75-50-30	50	30	876	9	9	0	30	67252	579
washington-75-60-25	60	25	1256	11	11	0	44	1	4
washington-75-70-42	70	42	1700	12	12	0	29	1	19
washington-75-80-48	80	48	2504	9	9	0	19	1	42
washington-75-90-47	90	47	3034	19	19	0	40	1	89
washington-75-100-52	100	52	3792	14	-	-	-	1	TLE
washington-100-10-6	10	6	18	2	2	0	33	0	0
washington-100-20-10	20	10	98	2	2	0	20	1	0
washington-100-30-12	30	12	206	2	2	0	17	2770	16
washington-100-40-18	40	21	422	4	4	0	19	1249	22
washington-100-50-27	50	27	596	5	9	44	33	127796	TLE
washington-100-60-34	60	34	854	10	10	0	29	1	3
washington-100-70-39	70	39	1204	17	17	0	44	8770	151
washington-100-80-32	80	32	1764	5	5	0	16	1	11
washington-100-90-43	90	43	2302	11	11	0	26	1	107
washington-100-100-43	100	43	2816	4	4	0	9	1	146
washington-200-125-55	125	55	2448	29	29	0	53	3733	338
washington-200-150-61	150	61	4468	10	61	84	100	1	TLE
washington-200-175-74	175	74	6048	20	74	73	100	1	TLE
washington-200-200-114	200	114	7290	32	114	72	100	1	TLE
washington-200-225-135	225	135	8928	2	-	-	-	0	TLE
washington-200-250-109	250	109	12806	6	-	-	-	0	TLE
washington-200-275-146	275	146	14390	11	-	-	-	0	TLE
washington-200-300-136	300	136	17418	-	-	-	-	-	TLE
washington-200-325-170	325	170	18520	-	-	-	-	-	TLE
washington-200-350-150	350	150	24436	-	-	-	-	-	TLE

Tabela 5.3: Resultados do modelo DMFM-MRP_{1,1}.

Para os dados presentes na Tabela 5.3, que contém os resultados do modelo DMFM-MRP_{1,1}, é possível observar um domínio quase total sobre os resultados da versão DMFM-MRP _{Λ, Ξ} , apresentando um LI com valor menor apenas para a instância “washington-200-300-136”. Para 29 instâncias o modelo DMFM-MRP_{1,1} retornou soluções ótimas, apresentando um aumento de 9 instâncias em relação ao modelo DMFM-MRP _{Λ, Ξ} . Em outras 4 instâncias o valor obtido não foi ótimo, mas o modelo retornou solução viável para o problema.

O tempo médio de execução do DMFM-MRP_{1,1}, para o conjunto de instâncias em que

foi retornada solução ótima, foi de 68 segundos, sendo o maior tempo de execução dentre esses casos inferior à 600 segundos. Ainda considerando esse conjunto, percebe-se que para 22 soluções, das 29 ótimas, o número de nós da árvore de enumeração foi inferior a 250. Mais especificamente, para 21 dessas instâncias o total de nós da árvore de enumeração foi 1 ou 0. Essas informações indicam que o modelo DMFM-MRP mostrou-se bastante apertado em que se refere a qualidade da solução relaxada para instâncias contendo até 125 nós. Ainda, o fato de serem necessários poucos nós na árvore de enumeração pode ser justificado possivelmente pela rápida melhora na qualidade dos limitantes.

Considerando as duas variações de DMFM-MRP, vale destacar que os valores de $M_d = \Lambda$ e $M_j = \Xi$ permitem que um nó terminal não atendido seja visitado na arborescência geradora por meio de múltiplas possibilidades de caminho. A modificação do valor das constantes M_d e M_j para 1 reduziu consideravelmente essas possibilidades para visitar um terminal k não atendido, forçando que esse terminal seja visitado a partir do nó artificial, pelos arcos (s, s') e (s', k) . Por fim, observa-se que a redução de valor dos parâmetros M_d e M_j possibilitou a obtenção de melhores resultados, mas ainda não possibilitou a geração de soluções viáveis para instâncias contendo 225 nós ou mais.

A Tabela 5.4 contém os resultados retornados pelo modelo DMFM-MRP_{1,1} sem a utilização dos pré-processamentos. O objetivo da apresentação desses resultados é demonstrar o impacto das fixações computadas pelos pré-processamentos na obtenção de limitantes.

Diante dos resultados apresentados na Tabela 5.4, nota-se uma redução considerável na obtenção de resultados ótimos e viáveis. Todas as análises apresentadas neste parágrafo comparam valores obtidos a partir do modelo DMFM-MRP_{1,1} com pré-processamentos. Primeiramente, o número de instâncias nas quais anteriormente era obtida solução ótima, dado o tempo de execução máximo, foi reduzido de 29 para 15, dentre as quais o número de nós do grafo não excede 60. Em 9 outras instâncias o modelo retornou solução viável, mas o *gap* não foi fechado. Em 16 testes não foram retornados limitantes superiores, enquanto com a aplicação dos pré-processamentos esse número era de apenas 7. Por fim, o tempo médio, considerando apenas instâncias resolvidas de maneira ótima, foi de 243 segundos, sendo aproximadamente três vezes maior que o tempo médio com utilização dos pré-processamentos.

Considerando os dados constantes na Tabela 5.5, percebe-se que o modelo AB-MRP_{1,1}, obteve soluções ótimas para 28 instâncias e viáveis para as 12 restantes. O tempo médio para obtenção de soluções ótimas foi de 159 segundos, e dentre elas o maior tempo de execução foi de 2175 segundos. Em apenas 8 instâncias, das 40 testadas, o número de nós da árvore de enumeração foi inferior a 250. Ainda, o valor médio de nós da árvore de enumeração retornado pelo resolvidor foi 249224. AB-MRP_{1,1} retornou soluções ótimas para todas as instâncias com até 100 nós, com exceção das instâncias “washington-75-90-47” e “washington-75-100-52”.

Considerando esses resultados, nota-se que, em contraste com o DMFM-MRP_{1,1}, o modelo AB-MRP_{1,1} apresentou limitantes inferiores e superiores para todas as instâncias do conjunto. Também houve um aumento da quantidade de nós da árvore de enumeração e o tempo médio de execução. O bom desempenho do modelo pode ser explicado pela redução considerável no número de variáveis e restrições. Observa-se que o modelo DMFM-MRP necessitou de menos tempo de execução e nós da árvore de enumeração para

Instância	V	D	A	LI	LS	gap	gap LS	Nós GRB	Tempo
washington-50-10-6	10	6	46	1	1	0	17	0	0
washington-50-20-11	20	11	184	4	4	0	36	1	4
washington-50-30-15	30	15	420	3	3	0	20	1	8
washington-50-40-23	40	23	830	6	6	0	26	1	15
washington-50-50-28	50	28	1248	13	13	0	46	1	1113
washington-50-60-35	60	35	1812	15	15	0	43	1	40
washington-50-70-37	70	37	2516	4	30	87	81	34	TLE
washington-50-80-39	80	39	3592	0	-	-	-	0	TLE
washington-50-90-51	90	51	4218	18	35	49	69	2567	TLE
washington-50-100-45	100	45	4416	14	31	55	69	1569	TLE
washington-75-10-4	10	4	32	3	3	0	75	1	0
washington-75-20-12	20	12	126	4	4	0	33	2643	9
washington-75-30-16	30	16	318	5	5	0	31	32517	152
washington-75-40-21	40	21	538	5	5	0	24	1	128
washington-75-50-30	50	30	876	5	9	44	30	16772	TLE
washington-75-60-25	60	25	1256	11	11	0	44	2831	195
washington-75-70-42	70	42	1700	4	41	90	98	1	TLE
washington-75-80-48	80	48	2504	0	-	-	-	0	TLE
washington-75-90-47	90	47	3034	0	-	-	-	0	TLE
washington-75-100-52	100	52	3792	0	-	-	-	0	TLE
washington-100-10-6	10	6	18	2	2	0	33	0	0
washington-100-20-10	20	10	98	2	2	0	20	3826	10
washington-100-30-12	30	12	206	2	2	0	17	29143	117
washington-100-40-18	40	21	422	4	4	0	19	77939	1848
washington-100-50-27	50	27	596	2	10	80	37	2557	TLE
washington-100-60-34	60	34	854	6	10	40	29	361	TLE
washington-100-70-39	70	39	1204	9	17	47	44	177	TLE
washington-100-80-32	80	32	1764	2	30	93	94	1	TLE
washington-100-90-43	90	43	2302	0	-	-	-	0	TLE
washington-100-100-43	100	43	2816	1	-	-	-	0	TLE
washington-200-125-55	125	55	2448	-	-	-	-	-	TLE
washington-200-150-61	150	61	4468	-	-	-	-	-	TLE
washington-200-175-74	175	74	6048	-	-	-	-	-	TLE
washington-200-200-114	200	114	7290	-	-	-	-	-	TLE
washington-200-225-135	225	135	8928	-	-	-	-	-	TLE
washington-200-250-109	250	109	12806	-	-	-	-	-	TLE
washington-200-275-146	275	146	14390	-	-	-	-	-	TLE
washington-200-300-136	300	136	17418	-	-	-	-	-	TLE
washington-200-325-170	325	170	18520	-	-	-	-	-	TLE
washington-200-350-150	350	150	24436	-	-	-	-	-	TLE

Tabela 5.4: Resultados do modelo DMFM-MRP_{1,1} sem a utilização de pré-processamentos.

certificar soluções ótimas. Em contrapartida, o modelo AB-MRP, com menos variáveis e restrições, permite que o resolvidor de PLI consiga explorar mais nós ao longo da árvore de enumeração, possibilitando melhores limitantes e obtenção de mais soluções viáveis.

Instância	V	D	A	LI	LS	gap	gap LS	Nós GRB	Tempo
washington-50-10-6	10	6	46	1	1	0	17	0	0
washington-50-20-11	20	11	184	4	4	0	36	557	0
washington-50-30-15	30	15	420	3	3	0	20	1	0
washington-50-40-23	40	23	830	6	6	0	26	17096	4
washington-50-50-28	50	28	1248	13	13	0	46	65	1
washington-50-60-35	60	35	1812	15	15	0	43	5637	3
washington-50-70-37	70	37	2516	16	16	0	43	16422	26
washington-50-80-39	80	39	3592	26	26	0	67	287148	584
washington-50-90-51	90	51	4218	35	35	0	69	205467	355
washington-50-100-45	100	45	4416	28	28	0	62	190420	262
washington-75-10-4	10	4	32	3	3	0	75	1	0
washington-75-20-12	20	12	126	4	4	0	33	229	0
washington-75-30-16	30	16	318	5	5	0	31	6430	3
washington-75-40-21	40	21	538	5	5	0	24	2628	3
washington-75-50-30	50	30	876	9	9	0	30	3269750	2175
washington-75-60-25	60	25	1256	11	11	0	44	13702	2
washington-75-70-42	70	42	1700	12	12	0	29	59552	136
washington-75-80-48	80	48	2504	9	9	0	19	34481	148
washington-75-90-47	90	47	3034	10	19	47	40	1190250	TLE
washington-75-100-52	100	52	3792	13	20	35	38	754860	TLE
washington-100-10-6	10	6	18	2	2	0	33	0	0
washington-100-20-10	20	10	98	2	2	0	20	1	0
washington-100-30-12	30	12	206	2	2	0	17	51	0
washington-100-40-18	40	21	422	4	4	0	19	2509	5
washington-100-50-27	50	27	596	9	9	0	33	173952	121
washington-100-60-34	60	34	854	10	10	0	29	59255	94
washington-100-70-39	70	39	1204	17	17	0	44	67860	183
washington-100-80-32	80	32	1764	5	5	0	16	51718	23
washington-100-90-43	90	43	2302	11	11	0	26	36600	160
washington-100-100-43	100	43	2816	4	4	0	9	55409	157
washington-200-125-55	125	55	2448	27	29	7	53	1446730	TLE
washington-200-150-61	150	61	4468	2	36	94	59	981311	TLE
washington-200-175-74	175	74	6048	7	37	81	50	426631	TLE
washington-200-200-114	200	114	7290	8	105	92	92	121458	TLE
washington-200-225-135	225	135	8928	8	15	47	11	49815	TLE
washington-200-250-109	250	109	12806	11	19	42	17	197778	TLE
washington-200-275-146	275	146	14390	11	71	85	49	47332	TLE
washington-200-300-136	300	136	17418	2	61	97	45	92629	TLE
washington-200-325-170	325	170	18520	16	77	79	45	46563	TLE
washington-200-350-150	350	150	24436	4	45	91	30	56654	TLE

Tabela 5.5: Resultados modelo AB-MRP_{1,1}.

Por fim, a Tabela 5.6 apresenta o número de instâncias resolvidas por cada modelo, organizada de modo que a primeira linha contém a quantidade de soluções ótimas, a segunda linha contém a quantidade de soluções viáveis quando não foi possível obter o resultado ótimo e, por fim, a terceira linha apresenta a soma de soluções ótimas e viáveis para cada modelagem. Os resultados da versão DMFM-MRP sem pré-processamento foram desconsiderados pelo fato de seu desempenho ser inferior, em todos os casos, aos resultados retornados pela versão com a utilização dos pré-processamentos.

	DMFM-MRP $_{\Lambda, \Xi}$	DMFM-MRP $_{1,1}$	AB-MRP $_{1,1}$
Ótimo	20	29	28
Viável	8	4	12
Total	28	33	40

Tabela 5.6: Contagem de soluções dos modelos.

5.3 Relaxações Lagrangianas

Para a implementação das relaxações lagrangianas RL-SP, RL-SPRC $_{\lambda}$, RL-SPRC $_{\xi}$ e RL-SPRC2 descritas, respectivamente, nas Seções 4.2.1, 4.2.2, 4.2.3 e 4.2.4, são consideradas as seguintes variações:

- Utilização do resolvedor de PLI na resolução do PPL;
- Utilização de algoritmos combinatórios na resolução do PPL;
- Inicialização de todos os multiplicadores lagrangianos com valor 0;
- Inicialização de todos os multiplicadores lagrangianos com valor das variáveis na solução ótima do problema dual resolvido pelo método de barreira;
- Remoção dos multiplicadores de Lagrange associados às restrições de variação de *delay*.

Nas implementações com resolvedor de PLI utilizou-se o Gurobi versão 9.0.3, mantendo a configuração padrão de seus parâmetros. O tempo máximo para resolução de cada instância utilizando o resolvedor PLI foi de 3600 segundos, enquanto para as versões combinatórias o limite foi de 1800 segundos. Foram utilizadas todas as fixações de variáveis de acordo com os três pré-processamentos apresentados e o valor das constantes M_d e M_j foi igual a 1 para todas as relaxações. Por fim, os valores de parâmetros utilizados no método de subgradiente foram configurados empiricamente conforme descrito a seguir:

- LI inicial: igual a 0, assumindo que nenhum terminal foi perdido;
- LS inicial: resultado da aplicação da heurística de busca local com os multiplicadores iniciais;
- O gap mínimo para encerrar a execução do método de subgradiente é dado por $(LS - LI) < 1$. Dado que, por exemplo, se o resultado do PPL for 0.1, significa que não atende-se pelo menos 1 terminal;
- Outra condição de parada utilizada foi o limite máximo de iterações, fixado em 2000;
- O tamanho do passo π é iniciado com valor 2.0. A cada 100 iterações do método de subgradiente em que não há melhora do LI, atualiza-se $\pi = \frac{\pi}{2}$.

Além dos valores descritos, foram testadas diferentes conjuntos de parâmetros. O tamanho do passo inicial foi igual a 1 e 1.5, com atualização a cada 50, 150, 200 e 300 iterações sem melhora do LS. O número máximo de iterações variou entre 500, 1000 e 3000.

Se uma RL utiliza como multiplicadores iniciais os valores do **Método de Barreiras**, haverá uma referência “-MB”. Para a versão da RL que desconsidera os multiplicadores associados com as restrições de variação de *delay*, a referência é “-SV”. Por fim, diferenciam-se versões com algoritmos **Combinatórios** das versões com **Resolvedor** utilizando, respectivamente, RL^C ou RL^R . Todas as RLs aplicam a heurística de busca local, descrita na Seção 4.4, na arborescência geradora mínima induzida pelos valores das variáveis y .

As Tabelas 5.7, 5.8, 5.9 e 5.10 contêm os resultados das relaxações RL-SP, RL-SPRC $_{\lambda}$, RL-SPRC $_{\xi}$ e RL-SPRC2, respectivamente. As primeiras quatro colunas contêm o nome da instância, quantidade de nós ($|V|$), número de nós terminais ($|D|$) e quantidade de arcos ($|A|$). As colunas subsequentes se dividem em variações da mesma RL, organizadas de modo que para cada variação estão dispostos os valores de limitante inferior (“LI”), limitante superior (“LS”) e o tempo de execução em segundos para cada instância. O valor TLE em uma linha representa que o algoritmo encerrou ao atingir o tempo limite de execução e os valores em **negrito** simbolizam que o limitante é igual ao melhor obtido dentre as variações presentes na tabela.

5.3.1 RL-SP

A Tabela 5.7 contêm os resultados associados com a relaxação lagrangiana RL-SP. Utiliza-se $RL-SP^R$ pois podem haver arcos com coeficientes negativos, consequentemente a versão $RL-SP^C$ não pôde ser aplicada. O mesmo se aplica à versão RL-SP-MB. Por fim, a última coluna apresenta os resultados da versão $RL-SP-SV^C$, cujos resultados dominaram, para todas as instâncias, a mesma versão utilizando resolvedor de PLI.

Diante dos resultados apresentados na Tabela 5.7, percebe-se a dificuldade de melhora na qualidade dos LIs iniciais para as versões $RL-SP^R$ e $RL-SP-MB^R$. Para a versão $RL-SP-SV$, foi observada uma melhora nos LIs para 10 instâncias. O tempo computacional, como era esperado, foi menor na versão combinatória, sendo em média 472 segundos, enquanto as versões com resolvedor tiveram em média 991 segundos, desconsiderando os TLEs e o tempo de resolução do problema de barreiras. Para quatro instâncias do conjunto o tempo não foi suficiente resolver o método de barreira, consequentemente não foram reportados limitantes. Para os limitantes superiores, segundo o teste estatístico de Iman-Davenport, não existe diferença estatisticamente significativa entre as amostras. Considerando a contagem de vitórias, a versão $RL-SP-SV^C$ apresentou desempenho superior, obtendo LSs melhores ou iguais às demais variações para 31 instâncias.

Instância	V	D	E	RL-SP ^R			RL-SP-MB ^R			RL-SP-SV ^C		
				LI	LS	Tempo	LI	LS	Tempo	LI	LS	Tempo
washington-50-10-6	10	6	23	0	1	19	0	1	20	0	1	9
washington-50-20-11	20	11	92	0	4	67	0	4	67	1	4	26
washington-50-30-15	30	15	210	0	3	180	0	3	181	0	3	57
washington-50-40-23	40	23	415	0	6	439	0	6	451	0	6	139
washington-50-50-28	50	28	624	0	13	657	0	13	655	0	13	215
washington-50-60-35	60	35	906	0	16	1036	0	16	1059	0	16	379
washington-50-70-37	70	37	1258	0	16	2083	0	16	2085	0	16	707
washington-50-80-39	80	39	1796	0	28	2560	0	29	1381	0	28	777
washington-50-90-51	90	51	2109	0	36	TLE	0	35	TLE	0	35	1564
washington-50-100-45	100	45	2208	0	30	TLE	0	29	TLE	0	30	1177
washington-75-10-4	10	4	16	0	3	10	0	3	10	0	3	8
washington-75-20-12	20	12	63	0	4	56	0	4	56	0	4	23
washington-75-30-16	30	16	159	0	5	166	0	5	168	0	5	55
washington-75-40-21	40	21	269	0	8	285	0	8	292	3	8	96
washington-75-50-30	50	30	438	0	10	555	0	11	562	0	11	213
washington-75-60-25	60	25	628	0	11	837	0	11	847	0	11	243
washington-75-70-42	70	42	850	0	15	1433	0	16	1492	0	17	606
washington-75-80-48	80	48	1252	0	14	2438	0	15	2464	0	15	1076
washington-75-90-47	90	47	1517	0	19	2958	0	19	2958	0	19	1197
washington-75-100-52	100	52	1896	0	29	TLE	0	29	TLE	0	29	1663
washington-100-10-6	10	6	9	0	2	13	0	2	14	1	2	8
washington-100-20-10	20	10	49	0	2	53	0	2	54	0	2	22
washington-100-30-12	30	12	103	0	2	107	0	2	108	1	2	36
washington-100-40-18	40	21	211	0	5	280	0	5	290	0	5	87
washington-100-50-27	50	27	298	0	12	447	0	10	445	1	10	157
washington-100-60-34	60	34	427	0	12	572	0	11	564	5	13	229
washington-100-70-39	70	39	602	0	18	1003	0	20	1000	2	19	397
washington-100-80-32	80	32	882	0	11	1664	0	12	1683	0	13	492
washington-100-90-43	90	43	1151	0	17	2100	0	17	2175	0	17	806
washington-100-100-43	100	43	1408	0	13	3107	0	13	3184	1	11	1099
washington-200-125-55	125	55	1224	0	35	2629	0	34	2580	0	35	1075
washington-200-150-61	150	61	2234	0	36	TLE	0	39	TLE	0	36	TLE
washington-200-175-74	175	74	3024	0	43	TLE	0	43	TLE	0	44	TLE
washington-200-200-114	200	114	3645	0	79	TLE	0	74	TLE	0	75	TLE
washington-200-225-135	225	135	4464	0	72	TLE	0	57	TLE	2	58	TLE
washington-200-250-109	250	109	6403	0	59	TLE	0	64	TLE	0	57	TLE
washington-200-275-146	275	146	7195	0	97	TLE	-	-	TLE	1	96	TLE
washington-200-300-136	300	136	8709	0	94	TLE	-	-	TLE	0	89	TLE
washington-200-325-170	325	170	9260	0	113	TLE	-	-	TLE	0	115	TLE
washington-200-350-150	350	150	12218	0	82	TLE	-	-	TLE	0	68	TLE

Tabela 5.7: Resultados da Relaxação RL-SP e suas variações.

5.3.2 RL-SPRC_λ

A Tabela 5.8 apresenta os resultados das variações da relaxação RL-SPRC_λ. A primeira coluna, RL-SPRC_λ^C, contém os valores da versão combinatória. Omitiram-se os resultados da versão com resolvidor de PLI pelo fato de serem inferiores à versão combinatória para todas as instâncias. O mesmo se aplica às relaxações RL-SPRC_λ-MB e RL-SPRC_λ-SV, cujos resultados são apresentados, respectivamente, na segunda e terceira coluna.

De acordo com os resultados dispostos na Tabela 5.8, pode-se destacar melhora na qualidade dos limitantes inferiores em relação às variações da relaxação lagrangiana RL-SP. Pode-se atribuir tal melhora à modificação do problema de caminhos resolvido no PPL, e como consequência o acréscimo da complexidade computacional. Nota-se uma diferença de desempenho causada pelos coeficientes negativos, dado que o tempo médio

Instância	V	D	E	RL-SPRC $_{\lambda}^C$			RL-SPRC $_{\lambda}$ -MB C			RL-SPRC $_{\lambda}$ -SV C		
				LI	LS	Tempo	LI	LS	Tempo	LI	LS	Tempo
washington-50-10-6	10	6	23	1	1	0	0	1	15	0	1	9
washington-50-20-11	20	11	92	2	4	34	0	4	79	2	4	30
washington-50-30-15	30	15	210	2	3	74	0	3	201	2	3	70
washington-50-40-23	40	23	415	4	6	492	0	6	440	0	6	162
washington-50-50-28	50	28	624	4	13	292	0	13	669	4	13	249
washington-50-60-35	60	35	906	0	15	443	0	17	1104	2	15	422
washington-50-70-37	70	37	1258	4	16	820	0	16	2088	2	16	798
washington-50-80-39	80	39	1796	14	28	1705	0	28	2503	12	28	857
washington-50-90-51	90	51	2109	10	35	TLE	0	36	TLE	7	35	1779
washington-50-100-45	100	45	2208	3	30	TLE	0	31	TLE	6	29	1316
washington-75-10-4	10	4	16	2	3	10	0	3	15	0	3	9
washington-75-20-12	20	12	63	1	4	36	0	4	67	1	4	29
washington-75-30-16	30	16	159	0	5	255	0	5	197	0	5	66
washington-75-40-21	40	21	269	3	6	211	0	7	334	3	7	108
washington-75-50-30	50	30	438	3	11	397	0	11	676	3	10	250
washington-75-60-25	60	25	628	1	11	306	0	11	848	1	11	273
washington-75-70-42	70	42	850	1	14	800	0	17	1753	1	14	698
washington-75-80-48	80	48	1252	0	14	1305	0	15	2895	1	15	1206
washington-75-90-47	90	47	1517	0	19	1461	0	19	3158	1	19	1321
washington-75-100-52	100	52	1896	4	27	TLE	0	29	TLE	5	25	TLE
washington-100-10-6	10	6	9	1	2	9	0	2	13	1	2	8
washington-100-20-10	20	10	49	0	2	36	0	2	65	0	2	26
washington-100-30-12	30	12	103	1	2	74	1	2	141	1	2	47
washington-100-40-18	40	21	211	1	5	269	0	5	455	1	5	108
washington-100-50-27	50	27	298	1	23	0	1	12	1192	1	10	212
washington-100-60-34	60	34	427	6	10	280	0	11	631	5	11	255
washington-100-70-39	70	39	602	8	17	664	0	20	1364	9	18	450
washington-100-80-32	80	32	882	2	9	1564	0	13	2009	2	9	568
washington-100-90-43	90	43	1151	1	14	1176	0	18	2499	2	14	907
washington-100-100-43	100	43	1408	1	9	1455	1	13	3500	1	11	1230
washington-200-125-55	125	55	1224	8	33	TLE	0	34	3114	8	33	1206
washington-200-150-61	150	61	2234	3	40	TLE	0	40	TLE	5	35	TLE
washington-200-175-74	175	74	3024	10	70	TLE	0	44	TLE	11	42	TLE
washington-200-200-114	200	114	3645	7	105	TLE	0	82	TLE	9	78	TLE
washington-200-225-135	225	135	4464	2	127	TLE	0	63	TLE	2	52	TLE
washington-200-250-109	250	109	6403	6	91	TLE	6	69	TLE	6	47	TLE
washington-200-275-146	275	146	7195	4	145	TLE	-	-	TLE	4	95	TLE
washington-200-300-136	300	136	8709	5	127	TLE	-	-	TLE	5	90	TLE
washington-200-325-170	325	170	9260	8	154	TLE	-	-	TLE	8	115	TLE
washington-200-350-150	350	150	12218	2	134	TLE	-	-	TLE	2	72	TLE

Tabela 5.8: Resultados da relaxação RL-SPRC $_{\lambda}$ e suas variações.

da versão RL-SPRC $_{\lambda}^C$, contendo arcos de custo negativo, foi 969 segundos enquanto, para RL-SPRC $_{\lambda}$ -SV C , a média de tempo ficou em 807 segundos. De modo geral, dobrou-se o tempo de execução médio em relação à relaxação RL-SP.

Considerando os valores de LI, segundo o teste de Iman-Davenport, existe DES entre as amostras. Mais especificamente, os limitantes inferiores obtidos pela versão RL-SPRC $_{\lambda}$ -MB C foram, para todas as instâncias, menores ou iguais às demais versões. Ao analisar os resultados retornados pelas versões RL-SPRC $_{\lambda}^C$ e RL-SPRC $_{\lambda}$ -SV C , não foi detectada DES, podendo assim assumir ambas como estatisticamente equivalentes. Portanto, pode-se selecionar uma versão como melhor utilizando a contagem do número de instâncias onde cada uma obteve o maior LI. Como a relaxação RL-SPRC $_{\lambda}^C$ obteve melhores limitantes em 7 instâncias e RL-SPRC $_{\lambda}$ -SV C foi superior em 10 casos, podemos considera esta última mais eficiente para este conjunto de teste.

Considerando os LSs, a utilização do teste estatístico de Iman-Davenport indicou DES entre as amostras. Com base na utilização do teste Nemenyi, detectou-se novamente a versão $\text{RL-SPRC}_\lambda\text{-MB}^C$ como a que detém os piores *rank*s médios e as versões RL-SPRC_λ^C e $\text{RL-SPRC}_\lambda\text{-SV}^C$ como estatisticamente equivalentes, mas a contagem de vitórias indica como melhor a relaxação $\text{RL-SPRC}_\lambda\text{-SV}^C$, que obteve LSs melhores ou iguais às demais versões para 35 das 40 instâncias. Percebe-se que a diferença na qualidade dos LSs vai se tornando mais evidente de acordo com o aumento do número de nós da instância. Pode-se justificar essa diferença pelo fato do algoritmo de SPRC da Boost executar por mais tempo quando existem arcos de custo negativo. Consequentemente, a relaxação realizará menos iterações do método de subgradiente para o mesmo tempo limite de execução.

5.3.3 RL-SPRC_ξ

A Tabela 5.9 apresenta os resultados das variações da relaxação RL-SPRC_ξ . A primeira, RL-SPRC_ξ^R com resolvidor PLI, a versão combinatória desta relaxação é inviável por características específicas dos valores de *jitter* das instâncias. Como dito anteriormente, para que o seja possível computar o SPRC entre um par de nós do grafo, é essencial que a quantidade de recurso consumido em cada arco seja estritamente maior que zero, caso contrário pode haver casos, considerando arcos com custo negativo e sem consumo de recurso, em que o algoritmo detecta ciclos negativos. A mesma justificativa se aplica a $\text{RL-SPRC}_\xi\text{-MB}^R$. A última coluna apresenta os resultados obtidos pela $\text{RL-SPRC}_\xi\text{-SV}^C$, já que a mesma obteve limitantes melhores ou iguais à versão com resolvidor PLI para todas as instâncias.

Na Tabela 5.9, vale ressaltar que o tempo computacional é semelhante ao da relaxação RL-SPRC_λ , dado que o PPL de ambos consiste em computar SPRCs considerando apenas um recurso. As médias de tempo foram, respectivamente, 1902, 1857 e 815, segundos para as relaxações RL-SPRC_ξ^R , $\text{RL-SPRC}_\xi\text{-MB}^R$ e $\text{RL-SPRC}_\xi\text{-SV}^C$. Como esperado, o tempo médio de execução da versão combinatória foi semelhante a relaxação $\text{RL-SPRC}_\lambda\text{-SV}^C$. Entretanto, comparando os LIs, a utilização do *jitter* como recurso se mostrou mais eficiente que o *delay*. Tal dado pode ser comprovado diante da comparação dos limitantes, de modo que a relaxação RL-SPRC_ξ obteve LI maior que RL-SPRC_λ para 21 instâncias e para outras 16 o resultado foi igual.

Considerando os LIs apenas das versões de RL-SPRC_ξ , ao aplicar o teste de Iman-Davenport, o resultado obtido indicou a existência de DES entre as amostras. Assim, ao utilizar o teste de Nemenyi agrupou-se as versões RL-SPRC_ξ^R e $\text{RL-SPRC}_\xi\text{-SV}^C$ com menor *rank* médio, o que significa que os resultados das melhores versões não apresentam DES. Considerando a contagem de vitórias de cada variação da relaxação RL-SPRC_ξ , pode-se assumir a relaxação $\text{RL-SPRC}_\xi\text{-SV}^C$ como melhor, dado que seus valores de LI foram melhores que a RL-SPRC_ξ^R em 6 casos, e para outras 32 instâncias o valor foi o mesmo.

Para os LSs, o teste estatístico de Iman-Davenport indicou DES entre as amostras. Com isso, ao aplicar o teste de Nemenyi, o resultado indica que a relaxação $\text{RL-SPRC}_\lambda\text{-SV}^C$ apresentou melhor qualidade de limitantes, retornando resultados melhores que as demais versões para 14 instâncias, enquanto que para 21 outras o LS foi igual. Essa

Instância	V	D	E	RL-SPRC $^R_\xi$			RL-SPRC $^R_\xi$ -MB R			RL-SPRC $^R_\xi$ -SV C		
				LI	LS	Tempo	LI	LS	Tempo	LI	LS	Tempo
washington-50-10-6	10	6	23	0	1	16	1	1	0	0	1	10
washington-50-20-11	20	11	92	2	4	74	0	4	75	2	4	31
washington-50-30-15	30	15	210	2	3	221	2	3	211	2	3	71
washington-50-40-23	40	23	415	0	6	479	0	6	471	0	6	169
washington-50-50-28	50	28	624	4	13	702	0	13	689	5	13	244
washington-50-60-35	60	35	906	2	15	997	0	15	999	2	15	433
washington-50-70-37	70	37	1258	3	16	2288	0	16	2293	4	16	793
washington-50-80-39	80	39	1796	19	29	2503	0	27	2494	20	28	868
washington-50-90-51	90	51	2109	14	36	TLE	0	35	TLE	14	35	1751
washington-50-100-45	100	45	2208	5	31	TLE	0	31	TLE	6	29	1310
washington-75-10-4	10	4	16	0	3	15	0	3	16	0	3	9
washington-75-20-12	20	12	63	2	4	67	1	4	70	1	4	28
washington-75-30-16	30	16	159	0	5	231	0	5	197	0	5	68
washington-75-40-21	40	21	269	4	7	312	4	7	325	4	7	109
washington-75-50-30	50	30	438	3	9	733	3	9	683	3	9	257
washington-75-60-25	60	25	628	2	11	902	2	11	897	2	11	275
washington-75-70-42	70	42	850	5	15	1575	3	15	1570	5	14	683
washington-75-80-48	80	48	1252	0	17	2731	0	11	2626	0	10	1218
washington-75-90-47	90	47	1517	1	19	3198	0	19	3072	2	19	1314
washington-75-100-52	100	52	1896	5	29	TLE	0	29	TLE	6	28	TLE
washington-100-10-6	10	6	9	1	1	0	0	1	12	1	2	8
washington-100-20-10	20	10	49	0	2	66	0	2	64	0	2	27
washington-100-30-12	30	12	103	1	2	164	1	2	153	1	2	50
washington-100-40-18	40	21	211	1	6	567	1	6	541	1	5	111
washington-100-50-27	50	27	298	1	10	2019	1	10	1001	1	10	214
washington-100-60-34	60	34	427	6	11	600	5	10	620	6	10	264
washington-100-70-39	70	39	602	9	18	1298	0	19	1327	9	17	450
washington-100-80-32	80	32	882	3	15	1968	0	11	1959	3	9	559
washington-100-90-43	90	43	1151	2	19	2379	0	14	2314	2	13	899
washington-100-100-43	100	43	1408	1	17	TLE	1	6	3451	1	7	1216
washington-200-125-55	125	55	1224	9	35	3054	0	31	2888	9	32	1196
washington-200-150-61	150	61	2234	6	38	TLE	0	39	TLE	6	36	TLE
washington-200-175-74	175	74	3024	18	42	TLE	0	43	TLE	18	42	TLE
washington-200-200-114	200	114	3645	21	70	TLE	0	76	TLE	21	73	TLE
washington-200-225-135	225	135	4464	2	57	TLE	2	53	TLE	2	46	TLE
washington-200-250-109	250	109	6403	6	60	TLE	6	60	TLE	6	54	TLE
washington-200-275-146	275	146	7195	9	98	TLE	-	-	TLE	9	94	TLE
washington-200-300-136	300	136	8709	10	88	TLE	-	-	TLE	10	88	TLE
washington-200-325-170	325	170	9260	13	117	TLE	-	-	TLE	13	113	TLE
washington-200-350-150	350	150	12218	3	78	TLE	-	-	TLE	3	66	TLE

Tabela 5.9: Resultados da relaxação RL-SPRC $_\xi$ e suas variações.

diferença na qualidade dos limitantes pode ser atribuída ao fato da versão combinatória executar mais iterações do método de subgradiente e mais aplicações do procedimento de busca local.

5.3.4 RL-SPRC2

A Tabela 5.10 contém os resultados mais relevantes associados à relaxação RL-SPRC2. A primeira coluna, RL-SPRC2 C contém os dados da versão combinatória. Foram omitidos os resultados da versão com resolvidor PLI pelo fato de serem inferiores a versão combinatória para todas as instâncias. A mesma justificativa aplica-se a terceira coluna, que apresenta os resultados obtidos pela versão RL-SPRC2-SV C . Para a versão RL-SPRC2-MB, optou-se por apresentar apenas os resultados obtidos pela versão com resolvidor.

Apesar dos valores serem próximos da versão combinatória, a utilização do resolvidor manteve uma qualidade mais alta na obtenção de limitantes para um maior número de instâncias.

Instância	V	D	E	RL-SPRC2 ^C			RL-SPRC2-MB ^R			RL-SPRC2-SV ^C		
				LI	LS	Tempo	LI	LS	Tempo	LI	LS	Tempo
washington-50-10-6	10	6	23	1	1	0	1	1	0	0	1	9
washington-50-20-11	20	11	92	3	4	34	2	4	82	2	4	32
washington-50-30-15	30	15	210	2	3	75	2	3	213	2	3	70
washington-50-40-23	40	23	415	4	6	786	0	6	457	0	6	158
washington-50-50-28	50	28	624	6	13	345	6	13	678	6	13	240
washington-50-60-35	60	35	906	2	15	435	2	15	1067	2	15	429
washington-50-70-37	70	37	1258	7	16	823	4	16	2290	4	16	790
washington-50-80-39	80	39	1796	23	27	1336	22	27	2649	22	27	860
washington-50-90-51	90	51	2109	19	35	1790	0	36	TLE	17	35	1708
washington-50-100-45	100	45	2208	16	29	TLE	13	29	TLE	13	29	1305
washington-75-10-4	10	4	16	2	3	9	0	3	16	0	3	9
washington-75-20-12	20	12	63	2	4	34	2	4	70	1	4	29
washington-75-30-16	30	16	159	0	5	265	0	5	215	0	5	67
washington-75-40-21	40	21	269	4	6	209	4	6	351	4	6	112
washington-75-50-30	50	30	438	3	9	375	3	10	777	3	10	249
washington-75-60-25	60	25	628	2	11	302	2	11	882	2	11	276
washington-75-70-42	70	42	850	6	13	856	5	14	1692	5	13	677
washington-75-80-48	80	48	1252	2	10	1292	2	10	2863	2	12	1186
washington-75-90-47	90	47	1517	3	19	1448	2	19	3252	2	19	1299
washington-75-100-52	100	52	1896	10	27	TLE	9	28	TLE	9	27	TLE
washington-100-10-6	10	6	9	2	1	0	1	1	0	1	2	8
washington-100-20-10	20	10	49	0	2	37	0	2	68	0	2	26
washington-100-30-12	30	12	103	1	2	90	2	2	0	1	2	48
washington-100-40-18	40	21	211	1	5	298	1	5	443	1	5	113
washington-100-50-27	50	27	298	1	10	TLE	1	9	814	1	10	230
washington-100-60-34	60	34	427	7	10	270	6	10	642	6	11	252
washington-100-70-39	70	39	602	9	17	663	9	17	1213	9	17	455
washington-100-80-32	80	32	882	3	7	TLE	3	8	1997	3	8	555
washington-100-90-43	90	43	1151	2	12	1216	2	12	2490	2	12	904
washington-100-100-43	100	43	1408	1	7	1513	1	9	TLE	1	6	1207
washington-200-125-55	125	55	1224	9	32	TLE	9	33	3051	9	32	1158
washington-200-150-61	150	61	2234	7	39	TLE	6	38	TLE	7	32	TLE
washington-200-175-74	175	74	3024	18	43	TLE	18	40	TLE	18	40	TLE
washington-200-200-114	200	114	3645	21	71	TLE	23	72	TLE	21	70	TLE
washington-200-225-135	225	135	4464	2	66	TLE	2	56	TLE	2	49	TLE
washington-200-250-109	250	109	6403	6	72	TLE	6	74	TLE	6	45	TLE
washington-200-275-146	275	146	7195	11	103	TLE	-	-	TLE	11	90	TLE
washington-200-300-136	300	136	8709	10	100	TLE	-	-	TLE	10	83	TLE
washington-200-325-170	325	170	9260	13	131	TLE	-	-	TLE	13	114	TLE
washington-200-350-150	350	150	12218	4	95	TLE	-	-	TLE	4	71	TLE

Tabela 5.10: Resultados da relaxação RL-SPRC2 e suas variações.

Diante dos resultados presentes na Tabela 5.10, destaca-se o domínio da relaxação RL-SPRC2 sobre as versões RL-SP, RL-SPRC _{λ} e RL-SPRC _{ξ} considerando o LI. Esse salto considerável de qualidade deve-se ao fato do PPL restringir mais rigidamente os caminhos, considerando *delay* e *jitter* simultaneamente. O tempo médio de execução das variações de RL-SPRC2 foram 950, 2066 e 802 segundos, para as versões RL-SPRC2^C, RL-SPRC2-MB^R e RL-SPRC2-SV^C, respectivamente. Diante dessas informações, nota-se um acréscimo moderado no tempo de execução das versões combinatórias ao considerar as relaxações apresentadas anteriormente. Ainda sim, pode-se assumir como baixo um aumento de aproximadamente 100 segundo na média, considerando a obtenção de LIs

melhores ou iguais para todas as instâncias do conjunto.

Considerando os LIs das diferentes versões da relaxação RL-SPRC2, as implementações RL-SPRC2^C e RL-SPRC2-SV^C obtiveram melhores resultados para o maior número de instâncias, dado que em apenas 2 casos o LI retornado pela relaxação RL-SPRC2-MB^R foi melhor que os demais.

Pelo fato da diferença entre o *rank* médio das relaxações RL-SPRC2^C e RL-SPRC2-SV^C ser bastante próximo ao valor da diferença crítica, é interessante aplicar o teste de Wilcoxon nos resultados. Diante desse resultado, foi detectada DES entre as duas abordagens e a contagem de vitórias reafirma a relaxação RL-SPRC2^C com os melhores ou iguais às demais versões para 38, das 40 instâncias do conjunto.

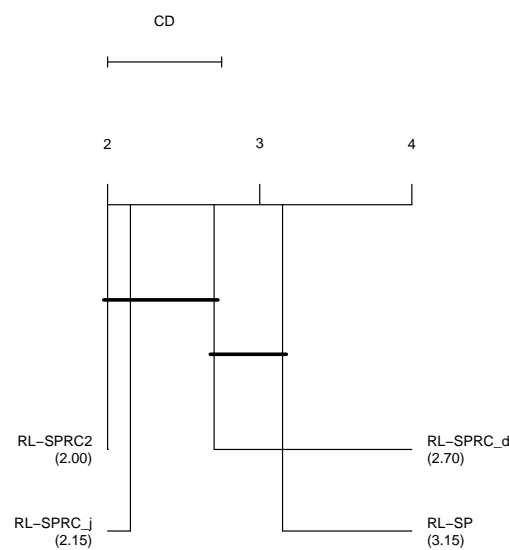
Para os LSs, a relaxação RL-SPRC2-SV^C apresentou melhor desempenho para um maior número de instâncias do conjunto, obtendo resultados melhores ou iguais às demais variações em 35, das 40 instâncias. Essa melhora está associada à demanda menor de tempo para computar os SPRCs para cada terminal do grafo, dado que não existe possibilidade de arcos com coeficiente negativo e quanto mais rápido os subproblemas são resolvidos, mais iterações do método de subgradiente podem ser executadas.

5.3.5 Análise geral

Por fim, considerando os resultados de todas as relaxações lagrangianas apresentadas, é possível sumarizar as seguintes análises:

- A implementação das RLs utilizando algoritmos combinatórios, de modo geral, apresentam limitantes inferiores e superiores de melhor qualidade, principalmente pelo fato de executarem mais iterações do método de subgradiente.
- A presença de multiplicadores que possam gerar custos negativos nos arcos, quando não inviabilizam, aumentam o tempo de execução das versões combinatórias.
- Inicializar o conjunto de multiplicadores de Lagrange com o valor da solução do método de barreiras forneceu os melhores LIs para 3 instâncias da relaxação RL-SPRC2. Nas demais instâncias o valor de LI obtido foi menor ou igual às versões com os multiplicadores inicializados em zero.
- Para todas as instâncias, os limitantes inferiores obtidos pela relaxação RL-SPRC2 foram maiores ou iguais aos limitantes das outras relaxações. Não houve acréscimo significativo no tempo médio de execução da relaxação RL-SPRC2, ainda que a complexidade do PPL seja assintoticamente maior. Diante dessas vantagens, é possível considerá-la como a melhor RL no quesito de limitantes inferiores.
- Considerando os limitantes superiores, independente da RL, mostrou-se mais vantajoso resolver a versão da relaxação desconsiderando os multiplicadores de variação de *delay*. Tais resultados se devem ao fato da redução de tempo de computação do problema de SPRC em grafos cujos arcos não tem custos negativos, permitindo mais iterações do método de subgradiente.

Para avaliar a qualidade dos LSs das relaxações RL-SP, RL-SPRC $_{\lambda}$, RL-SPRC $_{\xi}$ e RL-SPRC2, considera-se o melhor valor retornado para cada instância, dentre todas as variações da mesma relaxação. O resultado do teste de Iman-Davenport rejeitou a hipótese nula, ou seja, indicou a existência de DES entre as amostras. Prosseguindo assim para a aplicação do teste de Nemenyi, agrupa-se as relaxações RL-SPRC $_{\lambda}$, RL-SPRC $_{\xi}$ e RL-SPRC2 com menor *rank* médio, o que indica que eles são estatisticamente equivalentes. Esses dados estão representados na Figura 5.1, que também contém o valor computado pelos testes de Iman-Davenport (ID), o Valor Crítico (VC) e a Diferença Crítica (DC).



(a) ID = 7.81, VC = 2.68, DC = 0.74

Figura 5.1: Testes de Iman-Davenport e Nemenyi para as diferentes RLs

Por serem estatisticamente equivalentes, no procedimento para selecionar qual dentre as três relaxações é considerada melhor, pode-se analisar o número de vitórias para cada relaxação, como apresentado na Tabela 5.11. A primeira linha e coluna referenciam as relaxações, cada posição da tabela representa o número de vitórias da abordagem referente a linha sobre a abordagem na coluna.

Relaxações	RL-SP	RL-SPRC $_{\lambda}$	RL-SPRC $_{\xi}$	RL-SPRC2
RL-SP	-	04	00	02
RL-SPRC $_{\lambda}$	15	-	04	01
RL-SPRC $_{\xi}$	22	14	-	05
RL-SPRC2	21	18	09	-

Tabela 5.11: Contagem de vitórias entre as RLs.

Pela contagem de vitórias, observa-se que a relaxação RL-SPRC2 teve desempenho superior as outras duas em uma quantidade maior de instâncias. Enquanto a relaxação

RL-SP obteve os piores *rank*s médios e o menor número de vitórias em comparação com demais relaxações. É possível inferir que a qualidade dos LSs obtidos com a aplicação da heurística de busca local está fortemente atrelada com a qualidade dos caminhos gerados pela solução do PPL, pois o pior *rank* médio está associado com a relaxação mais fraca, que por sua vez resulta em um PPL cuja solução é menos interessante do ponto de vista heurístico. A relaxação contendo maior número de vitórias e o melhor *rank* médio, para esse conjunto de instâncias, se trata da RL-SPRC2, na qual o PPL tem a maior complexidade computacional dentre as relaxações propostas.

5.4 BRKGA

Na implementação dos decodificadores para o BRKGA, foi utilizada a biblioteca Boost C++ na versão 1.73 para computar a AGM. O tempo máximo para resolução de cada instância foi pré-definido com valor de 1800 segundos (30 minutos). Os valores de parâmetros utilizados no BRKGA foram empiricamente validados e consistem em:

- **População:** são geradas três populações independentes contendo 500 indivíduos cada uma. O conjunto de elite é formado por 10% da população. A cada iteração, 5% dos indivíduos da população são substituídos por mutantes;
- **Crossover:** um novo indivíduo é gerado a partir do cruzamento de dois indivíduos da geração atual, um do conjunto elite e um do conjunto não elite. Sendo 70% de probabilidade do filho herdar o alelo do pai do conjunto de elite.
- **Gerações:** o número máximo de gerações foi limitada em 1000, efetuando a troca dos 2 melhores indivíduos a cada 100 gerações.

As versões do BRKGA nesta seção são diferenciadas principalmente pela função de aptidão utilizada (F1, F2 ou F3), mas são consideradas características adicionais ao decodificador como a adição do procedimento de Busca Local (BL) ou a alteração no procedimento de geração das chaves aleatórias com base na frequência dos arcos da relaxação lagrangiana (RL).

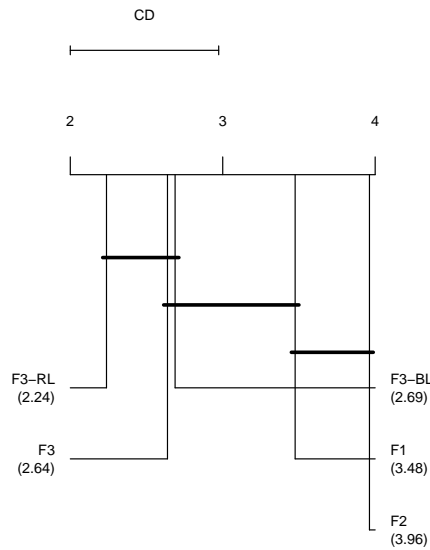
A Tabela 5.12 contém os resultados de 5 variações do BRKGA, as duas primeiras com função de aptidão F1 (“F1”) e a função de aptidão F2 (“F2”). A função de aptidão F3 é utilizada nas 3 outras variações, respectivamente (“F3”), (“F3-BL”) e (“F3-RL”). As primeiras 4 colunas da tabela apresentam, respectivamente, o nome da instância, quantidade de nós do grafo (“|V|”), número de nós terminais (“|D|”) e a quantidade de arcos (“|A|”). Para cada algoritmo são dispostas duas colunas: “LS” contendo o número de terminais atendidos na melhor solução e a coluna “Tempo” que apresenta o tempo de execução em segundos. O valor TLE representa que a execução atingiu o tempo limite de execução. Valores em **negrito** representam os melhores LSs dentre as versões do BRKGA para cada instância. A linha “**Melhor**” indica o número de instâncias nas quais o algoritmo obteve a melhor solução e que nenhuma outra variação atingiu o mesmo valor. A linha “**Igual**” contém o número de instâncias cujo limitante foi igual ao melhor obtido entre as demais versões. A linha “**Pior**” contabiliza o total de casos onde a solução retornada pelo decodificador foi inferior ao melhor limitante para a instância.

Instância	V	D	E	F1		F2		F3		F3-BL		F3-RL	
				LS	Tempo	LS	Tempo	LS	Tempo	LS	Tempo	LS	Tempo
washington-50-10-6	10	6	23	1	105	1	108	1	115	1	138	1	191
washington-50-20-11	20	11	92	4	328	4	339	4	354	4	399	4	587
washington-50-30-15	30	15	210	3	697	3	829	3	829	3	1.002	3	1347
washington-50-40-23	40	23	415	6	1.453	7	TLE	6	1.688	6	1.651	6	1683
washington-50-50-28	50	28	624	13	TLE	13	TLE	13	TLE	13	TLE	13	TLE
washington-50-60-35	60	35	906	17	TLE	19	TLE	17	TLE	18	TLE	17	TLE
washington-50-70-37	70	37	1258	16	TLE	18	TLE	16	TLE	17	TLE	16	TLE
washington-50-80-39	80	39	1796	29	TLE	29	TLE	29	TLE	29	TLE	29	TLE
washington-50-90-51	90	51	2109	36	TLE	36	TLE	36	TLE	36	TLE	36	TLE
washington-50-100-45	100	45	2208	35	TLE	35	TLE	34	TLE	34	TLE	34	TLE
washington-75-10-4	10	4	16	3	144	3	147	3	100	3	148	3	152
washington-75-20-12	20	12	63	5	413	4	388	4	338	4	513	4	468
washington-75-30-16	30	16	159	5	935	5	794	5	742	5	897	5	740
washington-75-40-21	40	21	269	5	1.127	8	1.172	6	1.228	6	1.351	5	1183
washington-75-50-30	50	30	438	11	TLE	11	1.707	11	TLE	11	TLE	11	TLE
washington-75-60-25	60	25	628	11	TLE	12	TLE	11	TLE	11	TLE	11	TLE
washington-75-70-42	70	42	850	23	TLE	16	TLE	13	TLE	13	TLE	13	TLE
washington-75-80-48	80	48	1252	29	TLE	26	TLE	13	TLE	11	TLE	13	TLE
washington-75-90-47	90	47	1517	19	TLE	19	TLE	19	TLE	19	TLE	19	TLE
washington-75-100-52	100	52	1896	28	TLE	33	TLE	27	TLE	28	TLE	26	TLE
washington-100-10-6	10	6	9	2	94	2	95	2	104	2	89	2	105
washington-100-20-10	20	10	49	2	272	2	359	2	379	2	285	2	334
washington-100-30-12	30	12	103	2	401	2	569	2	573	2	444	2	433
washington-100-40-18	40	21	211	4	833	4	TLE	4	1.004	4	1.337	4	1278
washington-100-50-27	50	27	298	9	1.147	9	TLE	9	1.666	9	1.485	9	TLE
washington-100-60-34	60	34	427	10	1.620	11	TLE	10	TLE	10	TLE	10	TLE
washington-100-70-39	70	39	602	24	TLE	24	TLE	24	TLE	24	TLE	24	TLE
washington-100-80-32	80	32	882	18	TLE	7	TLE	6	TLE	6	TLE	6	TLE
washington-100-90-43	90	43	1151	32	TLE	20	TLE	11	TLE	13	TLE	11	TLE
washington-100-100-43	100	43	1408	14	TLE	13	TLE	12	TLE	12	TLE	12	TLE
washington-200-125-55	125	55	1224	42	TLE	42	TLE	32	TLE	33	TLE	33	TLE
washington-200-150-61	150	61	2234	41	TLE	44	TLE	37	TLE	37	TLE	35	TLE
washington-200-175-74	175	74	3024	39	TLE	42	TLE	40	TLE	38	TLE	39	TLE
washington-200-200-114	200	114	3645	105	TLE	105	TLE	88	TLE	89	TLE	75	TLE
washington-200-225-135	225	135	4464	79	TLE	87	TLE	65	TLE	73	TLE	62	TLE
washington-200-250-109	250	109	6403	76	TLE	83	TLE	61	TLE	61	TLE	55	TLE
washington-200-275-146	275	146	7195	109	TLE	116	TLE	104	TLE	103	TLE	101	TLE
washington-200-300-136	300	136	8709	116	TLE	116	TLE	102	TLE	100	TLE	99	TLE
washington-200-325-170	325	170	9260	151	TLE	154	TLE	135	TLE	130	TLE	128	TLE
washington-200-350-150	350	150	12218	124	TLE	127	TLE	98	TLE	97	TLE	96	TLE
Melhor				0		0		1		2		9	
Igual				22		17		27		24		28	
Pior				18		23		12		14		3	

Tabela 5.12: Resultados das versões do BRKGA.

Considerando os resultados da Tabela 5.12, nota-se um consumo de tempo relativamente alto dos decodificadores independente de função de aptidão, dado que quase todas as instâncias contendo 50 ou mais nós, não foi possível executar o BRKGA por 1000 gerações antes de atingir o tempo limite de 30 minutos. A quantidade de TLEs de cada algoritmo foi bem semelhante, sendo BRKGA-F1 a versão com menos TLEs, totalizando 26. Em contrapartida, a versão BRKGA-F2 tem o maior número de instâncias onde o tempo limite foi alcançado, totalizando 29 casos. O tempo médio para as instâncias em que os BRKGAs executaram por 1000 gerações, se manteve entre 10 e 12 minutos, para todas as versões.

A aplicação do teste de Iman-Davenport indicou a existência de DES entre as amostras. Ao aplicar o teste de Nemenyi o resultado agrupou as versões F3, F3-BL e F3-RL com os menores valores de *rank* médio, o que indica equivalência estatística. Esses dados estão representados na Figura 5.2, que também contém o valor computado pelos testes de Iman-Davenport (ID), o Valor Crítico (VC) e a Diferença Crítica (DC).



(a) $ID = 9.52$, $VC = 2.42$, $DC = 0.97$

Figura 5.2: Testes de Iman-Davenport e Nemenyi para os decodificadores das versões do BRKGA.

Entretanto, ao notar a diferença de *rank* médio entre os BRKGAs F3-RL e F3, optou-se pela aplicação do teste estatístico de Wilcoxon para essas duas variações, motivado pelo fato desse teste estatístico apresentar resultados mais confiáveis tratando-se apenas de duas amostras. O resultado calculado pelo teste de Wilcoxon foi -5.47 , o que significa que existe DES entre as abordagens. Por fim, além da diferença estatística, a contagem de vitórias da versão F3-RL foi maior, possibilitando afirmar que essa versão pode ser considerada melhor em comparação com as demais variações do BRKGA.

Considerando a capacidade de melhora das soluções, é possível observar que a versão BRKGA-F2, que utiliza a abordagem de soma das penalizações, foi a função de aptidão que apresentou os piores resultados dentre as 5 abordagens comparadas. Dentre as versões do BRKGA que obtiveram os melhores resultados, mais especificamente as variações da função F3, nenhuma dominou por completo as demais abordagens, ou seja, não houve um algoritmo cujos resultados foram melhores em todas as 40 instâncias.

A utilização da busca local não apresentou ganhos significativos, obtendo apenas 6 soluções melhores que a versão BRKGA-F3 sem utilizar busca local, enquanto gerou resultados inferiores para outras 7 instâncias. Já a versão do decodificador F3-RL apresentou LSs com a maior qualidade, gerando as melhores soluções para 37 instâncias. Para as outras 3 instâncias em que o melhor valor foi obtido exclusivamente por outra versão, a diferença entre resultados foi menor ou igual a 2 terminais. Tais resultados reafirmam a melhora de qualidade das soluções com base na utilização da informação de frequência dos arcos selecionados pelo PPL na relaxação RL-SPRC2^C.

5.5 Avaliação Geral dos Limitantes Inferiores

Esta seção apresenta a comparação de todos LIs obtidos pelas abordagens propostas neste trabalho. Para tal, serão considerados os melhores LIs obtidos pelos modelos DMFM-MRP e AB-MRP, juntamente dos valores retornados por suas respectivas relaxações lineares e os melhores LIs das RLs.

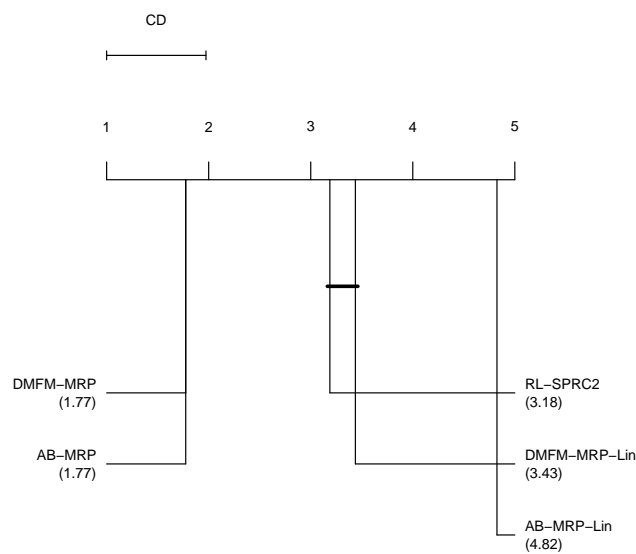
A Tabela 5.13 contém os LIs das melhores metodologias utilizadas nesta dissertação. As primeiras quatro colunas contém, respectivamente, o nome da instância, o número de nós do grafo (“|V|”), a quantidade de nós terminais (“|D|”) e o número de arcos (“|A|”). As colunas subsequentes dividem-se entre as abordagens, indicando o valor de LI obtido (“LI”) e o tempo de execução em segundos (“Tempo”). As colunas “DMFM-MRP” e “AB-MRP” apresentam, respectivamente, os melhores limitantes retornados pelo resolvidor PLI. As colunas “DMFM-MRP_{lin}” e “AB-MRP_{lin}” contém as respectivas relaxações lineares dos modelos. A última coluna apresenta os melhores resultados da relaxação RL-SPRC2, dado que esta RL detém os melhores LIs dentre todas as variações.

Instância	V	D	A	DMFM-MRP		AB-MRP		DMFM-MRP _{lin}		AB-MRP _{lin}		RL-SPRC2	
				LI	Tempo	LI	Tempo	LI	Tempo	LI	Tempo	LI	Tempo
washington-50-10-6	10	6	46	1	0,0	1	0,0	1	0,0	0	0,0	1	0
washington-50-20-11	20	11	184	4	0,0	4	0,0	3	0,1	0	0,0	3	34
washington-50-30-15	30	15	420	3	0,0	3	0,0	2	0,1	0	0,0	2	75
washington-50-40-23	40	23	830	6	3,0	6	4,0	1	0,6	0	0,0	4	786
washington-50-50-28	50	28	1248	13	3,0	13	1,0	7	1,0	0	0,0	6	345
washington-50-60-35	60	35	1812	15	4,0	15	3,0	3	2,7	0	0,0	2	435
washington-50-70-37	70	37	2516	16	41,0	16	26,0	4	4,9	0	0,0	7	823
washington-50-80-39	80	39	3592	26	25,0	26	584,0	23	5,1	0	0,0	23	1.336
washington-50-90-51	90	51	4218	35	285,0	35	355,0	18	13,4	0	0,0	19	1.790
washington-50-100-45	100	45	4416	28	71,0	28	262,0	12	11,8	0	0,0	16	TLE
washington-75-10-4	10	4	32	3	0,0	3	0,0	1	0,0	0	0,0	2	9
washington-75-20-12	20	12	126	4	0,0	4	0,0	2	0,1	0	0,0	2	34
washington-75-30-16	30	16	318	5	6,0	5	3,0	1	0,3	0	0,0	0	265
washington-75-40-21	40	21	538	5	1,0	5	3,0	5	0,5	0	0,0	4	209
washington-75-50-30	50	30	876	9	579,0	9	2175,0	3	1,0	0	0,0	3	375
washington-75-60-25	60	25	1256	11	4,0	11	2,0	2	0,7	0	0,0	2	302
washington-75-70-42	70	42	1700	12	19,0	12	136,0	6	5,3	0	0,0	6	856
washington-75-80-48	80	48	2504	9	42,0	9	148,0	2	11,1	0	0,0	2	1.292
washington-75-90-47	90	47	3034	19	89,0	10	TLE	3	10,7	0	0,0	3	1.448
washington-75-100-52	100	52	3792	14	TLE	13	TLE	9	24,4	0	0,0	10	TLE
washington-100-10-6	10	6	18	2	0,0	2	0,0	2	0,0	0	0,0	2	0
washington-100-20-10	20	10	98	2	0,0	2	0,0	1	0,0	0	0,0	0	37
washington-100-30-12	30	12	206	2	16,0	2	0,0	1	0,1	1	0,0	2	90
washington-100-40-18	40	21	422	4	22,0	4	5,0	1	0,5	0	0,0	1	298
washington-100-50-27	50	27	596	5	TLE	9	121,0	1	2,7	0	0,0	1	0
washington-100-60-34	60	34	854	10	3,0	10	94,0	7	1,2	0	0,0	7	270
washington-100-70-39	70	39	1204	17	151,0	17	183,0	9	2,6	0	0,0	9	663
washington-100-80-32	80	32	1764	5	11,0	5	23,0	4	3,8	0	0,0	3	TLE
washington-100-90-43	90	43	2302	11	107,0	11	160,0	3	8,6	0	0,0	2	1.216
washington-100-100-43	100	43	2816	4	146,0	4	157,0	2	17,3	0	0,0	1	1.513
washington-200-125-55	125	55	2448	29	338,0	27	TLE	10	13,4	0	0,0	9	TLE
washington-200-150-61	150	61	4468	10	TLE	2	TLE	7	187,4	0	0,1	7	TLE
washington-200-175-74	175	74	6048	20	TLE	7	TLE	19	2225,4	0	0,1	18	TLE
washington-200-200-114	200	114	7290	32	TLE	8	TLE	24	730,3	0	0,2	23	TLE
washington-200-225-135	225	135	8928	2	TLE	8	TLE	0	TLE	0	0,2	2	TLE
washington-200-250-109	250	109	12806	6	TLE	11	TLE	0	TLE	0	0,2	6	TLE
washington-200-275-146	275	146	14390	11	TLE	11	TLE	0	TLE	0	0,2	11	TLE
washington-200-300-136	300	136	17418	-	TLE	2	TLE	0	TLE	1	0,3	10	TLE
washington-200-325-170	325	170	18520	-	TLE	16	TLE	0	TLE	0	0,4	13	TLE
washington-200-350-150	350	150	24436	-	TLE	4	TLE	0	TLE	0	0,4	4	TLE

Tabela 5.13: Comparação dos melhores LIs entre as metodologias.

A aplicação do teste de Iman-Davenport nos valores apresentados na Tabela 5.13, indicou a presença de DES entre as amostras. Assim, aplicando o teste de Nemenyi

agruparam-se, com menor *rank* médio, os resultados dos modelos DMFM-MRP e AB-MRP, o que indica equivalência estatística. Esses dados estão representados na Figura 5.3, que também contém o valor computado pelos testes de Iman-Davenport (ID), o Valor Crítico (VC) e a Diferença Crítica (DC). Até mesmo a contagem de vitórias nos modelos foi igual: cada um contabilizou 6 vitórias, enquanto os LIs foram iguais para as outras 28 instâncias. Ainda assim, os resultados dos modelos não dominaram as demais metodologias para todas as instâncias, ou seja, alguma outra metodologia apresentou pelo menos um LI melhor que ambos os modelos.



(a) ID = 74.32, VC = 2.42, DC = 0.97

Figura 5.3: Testes de Iman-Davenport e Nemenyi para os Limitantes Inferiores.

Analisando as relaxações lineares e a RL, tornou-se evidente a falta de qualidade dos limitantes obtidos pela relaxação linear do modelo AB-MRP. O valor da relaxação do AB-MRP para 38 instâncias foi igual a 0 e nas outras duas instâncias, o resultado foi 1. A relaxação linear do modelo DMFM-MRP apresentou melhor qualidade, sendo estatisticamente equivalente à relaxação RL-SPRC2. Entretanto, ao considerar a contagem de vitórias entre essas duas abordagens, pode-se assumir que RL-SPRC2 foi melhor. Dado que RL-SPRC2 contabilizou vitória em 13 instâncias e DMFM-MRP_{lin} em apenas 11. Com base nos resultados e análises apresentados, notou-se o excelente desempenho dos modelos matemáticos, de modo que eles detêm os melhores LIs para 39 instâncias.

5.6 Avaliação Geral dos Limitantes Superiores

Esta seção apresenta a comparação de todos os LSs obtidos pelas diferentes abordagens propostas nesse trabalho. Para tal, serão considerados os melhores LSs dos modelos DMFM-MRP, AB-MRP, a melhor versão do BRKGA e da RL.

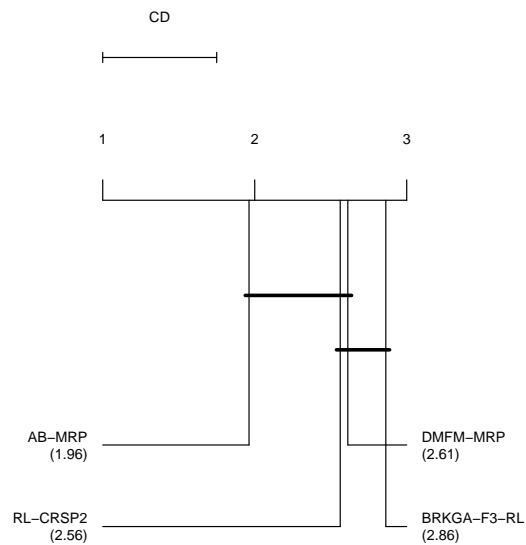
A Tabela 5.14 apresenta os LSs das melhores metodologias utilizadas nesta dissertação. As primeiras quatro colunas contém, respectivamente, o nome da instância, o número de nós do grafo (“|V|”), a quantidade de nós terminais (“|D|”) e o número de arcos (“|A|”). As colunas subsequentes dividem-se entre as abordagens, indicando o valor de LS (“LS”) obtido e o tempo de execução em segundos (“Tempo”). As colunas “DMFM-MRP” e “AB-MRP” contém, respectivamente, os melhores LS retornados pelo resolvidor PLI para cada modelo. A coluna “RL-SPRC2” contém os resultados associados com a RL que apresentou os melhores resultados. Por fim, a coluna “BRKGA-F3-RL” dispõe os LSs retornados pela melhor versão do BRKGA.

Instância	V	D	A	DMFM-MRP		AB-MRP		RL-SPRC2		BRKGA-F3-RL	
				LS	Tempo	LS	Tempo	LS	Tempo	LS	Tempo
washington-50-10-6	11	6	46	1	0	1	0	1	9	1	191
washington-50-20-11	21	11	184	4	0	4	0	4	32	4	587
washington-50-30-15	31	15	420	3	0	3	0	3	70	3	1.347
washington-50-40-23	41	23	830	6	3	6	4	6	158	6	1.683
washington-50-50-28	51	28	1248	13	3	13	1	13	240	13	TLE
washington-50-60-35	61	35	1812	15	4	15	3	15	429	17	TLE
washington-50-70-37	71	37	2516	16	41	16	26	16	790	16	TLE
washington-50-80-39	81	39	3592	26	25	26	584	27	860	29	TLE
washington-50-90-51	91	51	4218	35	285	35	355	35	1.708	36	TLE
washington-50-100-45	101	45	4416	28	71	28	262	29	1.305	34	TLE
washington-75-10-4	11	4	32	3	0	3	0	3	9	3	152
washington-75-20-12	21	12	126	4	0	4	0	4	29	4	468
washington-75-30-16	31	16	318	5	6	5	3	5	67	5	740
washington-75-40-21	41	21	538	5	1	5	3	6	112	5	1.183
washington-75-50-30	51	30	876	9	579	9	2.175	9	249	11	TLE
washington-75-60-25	61	25	1256	11	4	11	2	11	276	11	TLE
washington-75-70-42	71	42	1700	12	19	12	136	13	677	13	TLE
washington-75-80-48	81	48	2504	9	42	9	148	10	1.186	13	TLE
washington-75-90-47	91	47	3034	19	89	19	TLE	19	1.299	19	TLE
washington-75-100-52	101	52	3792	52	TLE	20	TLE	27	TLE	26	TLE
washington-100-10-6	11	6	18	2	0	2	0	1	8	2	105
washington-100-20-10	21	10	98	2	0	2	0	2	26	2	334
washington-100-30-12	31	12	206	2	16	2	0	2	48	2	433
washington-100-40-18	41	21	422	4	22	4	5	5	113	4	1.278
washington-100-50-27	51	27	596	9	TLE	9	121	10	230	9	TLE
washington-100-60-34	61	34	854	10	3	10	94	10	252	10	TLE
washington-100-70-39	71	39	1204	17	151	17	183	17	455	24	TLE
washington-100-80-32	81	32	1764	5	11	5	23	7	555	6	TLE
washington-100-90-43	91	43	2302	11	107	11	160	12	904	11	TLE
washington-100-100-43	101	43	2816	4	146	4	157	6	1.207	12	TLE
washington-200-125-55	126	55	2448	29	338	29	TLE	32	1.158	33	TLE
washington-200-150-61	151	61	4468	61	TLE	36	TLE	32	TLE	35	TLE
washington-200-175-74	176	74	6048	74	TLE	37	TLE	40	TLE	39	TLE
washington-200-200-114	201	114	7290	114	TLE	105	TLE	70	TLE	75	TLE
washington-200-225-135	226	135	8928	-	TLE	15	TLE	49	TLE	62	TLE
washington-200-250-109	251	109	12806	-	TLE	19	TLE	45	TLE	55	TLE
washington-200-275-146	276	146	14390	-	TLE	71	TLE	90	TLE	101	TLE
washington-200-300-136	301	136	17418	-	TLE	61	TLE	83	TLE	99	TLE
washington-200-325-170	326	170	18520	-	TLE	77	TLE	114	TLE	128	TLE
washington-200-350-150	351	150	24436	-	TLE	45	TLE	71	TLE	96	TLE

Tabela 5.14: Comparação dos melhores LSs entre as metodologias.

A aplicação do teste de Iman-Davenport nos valores de limitante apresentados na tabela 5.14, indicaram DES entre as amostras. Assim, agrupou-se com base no teste de Nemenyi, as abordagens AB-MRP, RL-SPRC2 e DMFM-MRP com os menores *rank*s médios. Esses dados estão representados na Figura 5.4, que também contém o valor computado pelos testes de Iman-Davenport (ID), o Valor Crítico (VC) e a Diferença

Crítica (DC). Entretanto, diante da diferença de *rank* médio entre o modelo AB-MRP e a relaxação RL-SPRC2, optou-se pela aplicação do teste estatístico de Wilcoxon, que indicou a existência de DES entre as duas amostras. Com isso, pode-se considerar que o modelo AB-MRP consiste na melhor abordagem para obtenção de LSs. Entretanto, não houve domínio total sobre todas as demais abordagens, ou seja, não houve nenhum algoritmo cujos resultados foram os melhores para todas as 40 instâncias.



(a) ID = 3.73, VC = 2.68, DC = 0.74

Figura 5.4: Testes de Iman-Davenport e Nemenyi para os Limitantes Superiores.

O modelo AB-MRP obteve o melhor desempenho dentre as abordagens, garantindo LSs melhores ou iguais às demais metodologias para 37 instâncias do conjunto. Mais especificamente, em 8 instâncias o valor obtido foi melhor dentre todos os algoritmos. Considerando o modelo DMFM-MRP, cujo valor de *rank* médio está bem próximo ao *rank* da relaxação RL-SPRC2, observa-se que a contagem de vitórias entre eles foi idêntica, 11 casos para cada um. Por fim, temos os resultados da versão do BRKGA F3-RL, que obteve limitantes iguais aos melhores obtidos pelas demais abordagens em 18 instâncias, mas que, ainda assim, manteve-se no pior *rank* médio.

Capítulo 6

Considerações Finais

Os objetivos principais deste trabalho consistiram na definição e realização de um estudo computacional do MS-MRP-QoS conduzido a partir de um conjunto de 40 instâncias baseadas em simulação de tráfego e de rede. Foram exploradas técnicas de solução exata utilizando PLI e Relaxações Lagrangianas e métodos heurísticos capazes de gerar limitantes inferiores e superiores.

Considerando os métodos exatos, foram adaptados pré-processamentos, apresentados por Ribeiro et al. [34], para fixação de variáveis. Desenvolveram-se dois modelos matemáticos, DMFM-MRP e AB-MRP. Ambos se mostraram eficazes em resolver de maneira ótima instâncias com até 125 nós e gerando soluções viáveis para todas as 40 instâncias do conjunto de teste. Ainda, o DMFM-MRP serviu de base para o desenvolvimento de um conjunto contendo quatro Relaxações Lagrangianas (RL). Tais relaxações apresentaram bons resultados com baixo consumo de tempo, fato que permite a possibilidade de utilização para instâncias maiores.

Em relação aos métodos heurísticos, foi desenvolvida uma heurística de busca local em arborescências, aplicada conjuntamente ao algoritmo de solução das RLs, objetivando a geração e aperfeiçoamento de soluções viáveis. Além disso, foram desenvolvidas quatro variações da meta-heurística BRKGA, de modo que três delas variam a função de aptidão do decodificador e a última abordagem extrai a frequência de arcos durante a resolução da RL e utiliza essa informação na geração das chaves aleatórias.

Ao final, todos os algoritmos foram comparados através de testes estatísticos não paramétricos. Como não existem resultados para o MS-MRP-QoS presentes na literatura, destaca-se a necessidade da comparação de resultados entre as diferentes metodologias desenvolvidas. De tal modo, é possível validar, utilizando múltiplas abordagens, a qualidade dos limitantes inferiores e superiores. Ao considerar o conjunto de todas as metodologias desenvolvidas, destaca-se o modelo AB-MRP como a melhor abordagem para obtenção de LIs e LSs. As RLs geram bons limitantes inferiores e superiores, não sendo melhores que os modelos para todos os casos, mas o fato de existirem versões combinatórias eficientes, permite a solução de instâncias maiores. Por fim, mesmo testando diversas funções de aptidão e detectando melhoras com a alteração do processo de geração de chaves aleatórias, não foi possível gerar LSs com o BRKGA que estivessem acima das demais abordagens desenvolvidas.

Como trabalhos futuros, pode-se destacar a possibilidade de tornar o modelo AB-

MRP mais forte, por exemplo considerando um conjunto exponencial de restrições para garantir conectividade entre nós. Por ser de aplicação geral em arborescências, a heurística de busca local desenvolvida pode ser aplicada em outras meta-heurísticas. Ainda, pode-se expandir o conjunto de instâncias, tratando de simulações cada vez maiores em quantidade de veículos ou até mesmo considerando dados que se baseiam em cenários reais.

As técnicas desenvolvidas neste trabalho podem ser aplicadas em outras variantes do MS-MRP-QoS como MRP-QoS considerando as mesmas métricas de QoS ou adicionando mais restrições ao conjunto, tais como o número de saltos e a estimativa de duração do enlace. Ainda é possível gerar resultados para novas variantes que combinam o MS-MRP-QoS com o MRP-QoS, ou seja, o objetivo consiste em decidir quais terminais serão atendidos e ao mesmo tempo computar o caminho de menor custo no atendimento desse terminal.

Referências Bibliográficas

- [1] Boost c++ libraries. Disponível em <https://www.boost.org/>. Acessado 29 Janeiro 2020.
- [2] Exemplo de funcionamento de uma vanet. [http://spiropjects.com/blog/cat-view-more.php?blogname=What-is-Vehicular-Ad-hoc-Network-\(VANET\)?&id=29](http://spiropjects.com/blog/cat-view-more.php?blogname=What-is-Vehicular-Ad-hoc-Network-(VANET)?&id=29). Acessado: 30/01/2021.
- [3] James Bean. Genetics and random keys for sequencing and optimization. *ORSA Journal of Computing*, 6, 2006.
- [4] John E. Beasley. Lagrangian relaxation. In Colin R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, chapter 6, pages 243–303. John Wiley & Sons, Inc., New York, NY, USA, 1993.
- [5] Rajashekhar C. Biradar and Sunilkumar S. Manvi. Review of multicast routing mechanisms in mobile ad hoc networks. *Journal of Network and Computer Applications*, 35(1):221–239, 2012.
- [6] Salim Bitam and Abdelhamid Mellouk. Bee life-based multi constraints multicast routing optimization for vehicular ad hoc networks. *Journal of Network and Computer Applications*, 36(3):981–991, 2013.
- [7] Salim Bitam, Abdelhamid Mellouk, and Sherali Zeadally. Hybr: A hybrid bio-inspired bee swarm routing protocol for safety applications in vehicular ad hoc networks (vanets). *Journal of Systems Architecture*, 59(10, Part B):953–967, 2013.
- [8] Thomas Heide Clausen, Philippe Jacquet, Cedric Adjih, Anis Laouiti, Pascale Minet, Paul Muhlethaler, Amir Qayyum, and Laurent Viennot. Optimized link state routing protocol (olsr). 01 2003.
- [9] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.
- [10] S. L. O. B. Correia, J. Celestino, and O. Cherkaoui. Mobility-aware ant colony optimization routing for vehicular ad hoc networks. In *IEEE Wireless Communications and Networking Conference*, pages 1125–1130. IEEE, 2011.
- [11] Alisson Barbosa de Souza. Mav-aodv - um protocolo multicast baseado em colônias de formigas para redes ad hoc veiculares. Master’s thesis, Centro de Ciências e Tecnologia, Universidade Estadual do Ceará, 2012.

- [12] Janez Demšar. Statistical comparisons of classifiers over multiple data sets. 7:1–30, 2006.
- [13] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences*. Cengage Learning, 2011.
- [14] Reinhard Diestel. *Graph Theory*. Springer, 2005.
- [15] Peppino Fazio, Floriano De Rango, Cesare Sottile, and Amilcare Francesco Santamaria. Routing optimization in vehicular networks: A new approach based on multi-objective metrics and minimum spanning tree. *International Journal of Distributed Sensor Networks*, 9(11):59–69, 2013.
- [16] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.
- [17] José Gonçalves and Mauricio Resende. Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics*, 17:487–525, 2011.
- [18] Anuj Gupta, Harsh Sadawarti, and Anil Verma. Implementation of dymo routing protocol. *International Journal of Information Technology, Modeling and Computing*, 1, 06 2013.
- [19] M. Haklay and P. Weber. OpenStreetMap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.
- [20] Frederick S Hillier and Gerald J Lieberman. *Introdução à pesquisa operacional*. McGraw Hill Brasil, 2013.
- [21] Stefan Irnich and Guy Desaulniers. Shortest path problems with resource constraints. In Solomon M.M. Desaulniers G., Desrosiers J., editor, *Column Generation*, chapter 2, pages 33–65. Springer, Boston, MA, 2006.
- [22] Teerawat Issariyakul and Ekram Hossain. *Introduction to Network Simulator NS2*. Springer Publishing Company, Incorporated, 1st edition, 2010.
- [23] Rezwana Karim. Vanet: Superior system for content distribution in vehicular network applications. *Tech. rep.*, 2008.
- [24] Brad Karp and H. T. Kung. Gpsr: Greedy perimeter stateless routing for wireless networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 243–254. Association for Computing Machinery, 2000.
- [25] Gholamreza Khorasani, Ashkan Tatari, Ali Yadollahi, and Milad Rahimi. Evaluation of intelligent transport system in road safety. *International Journal of Chemical, Environmental & Biological Sciences*, 1(1):110–118, 2013.

- [26] Christian Lochert, Martin Mauve, Holger Füller, and Hannes Hartenstein. Geographic routing in city scenarios. *Mobile Computing and Communications Review*, 9:69–72, 2005.
- [27] Nelson Maculan. The Steiner problem in graphs. In Silvano Martello, Gilbert Laporte, Michel Minoux, and Celso Ribeiro, editors, *Surveys in Combinatorial Optimization*, volume 132 of *North-Holland Mathematics Studies*, chapter 6, pages 185–211. North-Holland, 1987.
- [28] Thomas L. Magnanti and Laurence A. Wolsey. *Optimal trees*, volume 7 of *Handbooks in Operations Research and Management Science*, chapter 9, pages 503–615. Elsevier, 1995.
- [29] Bilal. Mustafa and Umar Waqas Raja. Issues of routing in vanet. Master’s thesis, Blekinge Institute of Technology, 2011.
- [30] George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley-Interscience, New York, NY, USA, 1988.
- [31] Carlos A.S. Oliveira and Panos M. Pardalos. A survey of combinatorial optimization problems in multicast routing. *Computers and Operations Research*, 32(8):1953–1981, 2005.
- [32] Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.
- [33] Klaus Plöchl and Hannes Federrath. A privacy aware and efficient security infrastructure for vehicular ad hoc networks. *Computer Standards & Interfaces*, 30(6):390–397, 2008.
- [34] Celso C. Ribeiro, Tiago de A. Santos, and Cid C. de Souza. Multicast routing under quality of service constraints for vehicular ad hoc networks: mathematical formulation and a relax-and-fix heuristic. *International Transactions in Operational Research*, 26(4):1339–1364, 2019.
- [35] George F. Riley and Thomas R. Henderson. The NS-3 network simulator. In Klaus Wehrle, Mesut Güneş, and James Gross, editors, *Modeling and Tools for Network Simulation*, chapter 2, pages 15–34. Springer, Berlin, Heidelberg, 2010.
- [36] G. N. Rouskas and I. Baldine. Multicast routing with end-to-end delay and delay variation constraints. *IEEE Journal on Selected Areas in Communications*, 15(3):346–356, 1997.
- [37] Elizabeth Royer and Charles Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Mobile Computing and Networking*, pages 207–218. IEEE, 2001.

- [38] H. Saleet, R. Langar, K. Naik, R. Boutaba, A. Nayak, and N. Goel. Intersection-based geographical routing protocol for vanets: A proposal and analysis. *IEEE Transactions on Vehicular Technology*, 60(9):4560–4574, 2011.
- [39] A. Sebastian, M. Tang, Y. Feng, and M. Looi. A multicast routing scheme for efficient safety message dissemination in VANET. In *2010 IEEE Wireless Communication and Networking Conference*, pages 1–6. IEEE, 2010.
- [40] Bhupendra Singh and Ankit Gupta. Recent trends in intelligent transportation systems: a review. *Journal of Transport Literature*, 9(2):30–34, 2015.
- [41] T. Taleb, E. Sakhaee, A. Jamalipour, K. Hashimoto, N. Kato, and Y. Nemoto. A stable routing protocol to support its services in vanet networks. *IEEE Transactions on Vehicular Technology*, 56(6):3337–3347, 2007.
- [42] R.F. Toso and Mauricio Resende. A c++ application programming interface for biased random-key genetic algorithms. *Optimization Methods and Software*, 30, 01 2015.
- [43] R. Vaishampayan and J. J. Garcia-Luna-Aceves. Efficient and robust multicast routing in mobile ad hoc networks. In *IEEE International Conference on Mobile Ad-hoc and Sensor Systems*, pages 304–313. IEEE, 2004.
- [44] András Varga and Rudolf Hornig. An overview of the OMNeT++ simulation environment. In *International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, pages 1–10. Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2008.
- [45] P. Winter. Steiner problem in networks: A survey. *Networks*, 17(2):129–167, 1987.