

UNIVERSIDADE FEDERAL FLUMINENSE

EMANUELLA TOLEDO LOPES

**Algoritmos para uma Variante do Problema da
Árvore Geradora Mínima Generalizado com Prêmios
nos Vértices**

NITERÓI

2014

Ficha Catalográfica elaborada pela Biblioteca da Escola de Engenharia e Instituto de Computação da UFF

L864 Lopes, Emanuella Toledo
 Algoritmos para uma variante do problema da árvore geradora
 mínima generalizado com prêmios nos vértices / Emanuella Toledo
 Lopes. – Niterói, RJ : [s.n.], 2014.
 71 f.

 Dissertação (Mestrado em Computação) - Universidade Federal
 Fluminense, 2014.
 Orientadores: Luiz Satoru Ochi, Yuri Abitbol de Menezes Frota.

 1. Metaheurística. 2. Algoritmo. 3. Programação matemática. I.
 Título.

CDD 005.136

UNIVERSIDADE FEDERAL FLUMINENSE

EMANUELLA TOLEDO LOPES

**Algoritmos para uma Variante do Problema da
Árvore Geradora Mínima Generalizado com Prêmios
nos Vértices**

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização.

Orientador:

LUIZ SATORU OCHI

Co-orientador:

YURI ABITBOL DE MENEZES FROTA

NITERÓI

2014

EMANUELLA TOLEDO LOPES

Algoritmos para uma Variante do Problema da Árvore Geradora Mínima Generalizado
com Prêmios nos Vértices

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Algoritmos e Otimização.

Aprovada por:

Prof. Luiz Satoru Ochi - Orientador, UFF

Prof. Yuri Abitbol de Menezes Frota - Co-orientador, UFF

Prof. Luidi Gelabert Simonetti, UFF

Prof. José Elias Claudio Arroyo, UFV

Niterói-RJ, 05 de Setembro de 2014

*À minha amada avó.
(In Memoriam)*

Agradecimentos

Foram muitos os momentos de adversidades e alegrias no decorrer desse trabalho, momentos esses essenciais para o meu crescimento pessoal e profissional. Agradeço à Deus por tê-los vivido e por estar sempre comigo.

Meu amoroso agradecimento à minha família. Aos meus pais, Cícera e Genildo, por toda ajuda e apoio às minhas escolhas. À minha irmã Emanuelle, minha melhor amiga com quem sempre posso contar. Ao meu irmão Emmanuel, por estar cuidando dos nossos pais de perto. Ao meu namorado Phillipe, que esteve ao meu lado em todos os momentos, inclusive na produção deste trabalho. Muito obrigada por todo amor compartilhado, toda ajuda, apoio, paciência e compreensão. Não teria conseguido chegar até aqui sem vocês.

Agradeço aos meus orientadores, Prof. Luiz Satoru e Prof. Yuri Abitbol, pelos ensinamentos, apoio constante e ajuda nos momentos difíceis.

Minha gratidão aos amigos Renathinha, Edcarllos e Eyder, por todo carinho, ajuda e apoio. Sem vocês com certeza tudo teria sido mais difícil.

Aos amigos de Maceió e da UFF, que contribuíram direta ou indiretamente com este trabalho, e à alHouse, pela ajuda no início do mestrado e momentos de descontração.

Um agradecimento especial ao meu orientador da graduação, Prof. Alejandro Frery, e à Profa. Eliana Almeida, pela inspiração, apoio ao mestrado e amizade.

Agradeço também à Capes, por ter financiado meus estudos.

Emanuella Toledo Lopes

Resumo

O Problema da Árvore Geradora Mínima Generalizado (PAGMG) pode ser modelado sobre um grafo G conexo, não direcionado e ponderado cujos vértices estão divididos em componentes. O objetivo do PAGMG é encontrar uma árvore T que cubra exatamente um vértice de cada componente de G , de forma que a soma dos custos das arestas de T seja mínima. A versão *at Least* do PAGMG, L-PAGMG, objetiva encontrar uma árvore T que cubra pelo menos um vértice de cada componente de G . O PAGMG com Coleta de Prêmios (CP-PAGMG), baseado no PAGMG, objetiva minimizar a diferença do custo total das arestas utilizadas pela soma total dos prêmios coletados nos vértices selecionados.

Considerando as versões do PAGMG apresentadas, pode-se afirmar que estas são análogas a muitos problemas na vida real. No entanto, elas ainda não se aplicam a algumas classes de problemas mais específicos, como por exemplo problemas em que é exigido um prêmio (ou capacidade) mínimo para cada componente. Dessa forma, observou-se a necessidade de definir problemas que sejam mais específicos e que possam representar ainda mais problemas da vida real.

Com este intuito, este trabalho propõe uma nova variante para o PAGMG com Prêmios nos Vértices (PAGMG-P). Sua definição é baseada no L-PAGMG, com a característica adicional de que existe um prêmio não negativo associado a cada vértice e cada componente possui um prêmio mínimo a ser adquirido. Não foram encontrados trabalhos sobre o PAGMG-P na literatura. Dessa forma, este trabalho propõe formulações matemáticas, métodos heurísticos e exatos e instâncias do problema para analisar o desempenho dos métodos.

As duas formulações matemáticas propostas foram baseadas em formulações da literatura; a primeira formulação foi baseada no L-PAGMG enquanto a segunda no CP-PAGMG. A segunda formulação gerou melhores resultados, tanto em tempo computacional quanto em custo da solução encontrada. As metaheurísticas implementadas foram GRASP e ILS, e ambas utilizaram o VND como método de aprimoramento. Os resultados alcançados com o ILS se sobressaíram tanto no custo da solução encontrada quanto no tempo de execução. O método exato proposto utiliza os resultados das metaheurísticas como entrada. Esse método alcançou os melhores resultados entre os algoritmos propostos quando a metaheurística utilizada foi a ILS. São apresentados testes computacionais que ilustram a eficiência do método proposto.

Palavras-chave: Metaheurísticas. Programação Matemática. Problema da Árvore Geradora Mínima Generalizado.

Abstract

The Generalized Minimum Spanning Tree Problem (GMSTP) can be modeled in a weighted and undirected graph G whose vertices are divided into components. GMSTP aims to find a tree T that covers exactly one vertex in each component of G , minimizing the sum of edges costs. The GMSTP at Least version, L-GMSTP, aims to find a tree T that covers at least one vertex in each component of G . The Prize-Collecting GMSTP (PC-GMSTP), based on GMSTP, aims to minimize the difference between the sum of the selected edge costs and the sum of the collected prizes in the selected vertices.

We notice that these versions of the GMSTP can model many problems in real life. However, they still do not apply to some classes of more specific problems, such as problems that require a minimum prize (or capacity) in each component. Therefore, we notice the need to define problems that are more specific and that may represent more problems in real life.

In order to reach this goal, we propose a new GMSTP variant with prizes in the vertices (P-GMSTP). This new problem is based on the L-GMSTP variant, with the additional feature that there is a non-negative prize associated to each vertex and each component has a minimum prize to be collected. We did not find any studies about P-GMSTP in the literature. Thus, we propose mathematical formulations, exact and heuristic algorithms and instances for this variant.

The two mathematical formulations proposed were based on formulations in the literature; the first formulation was based on L-PAGMG and the second formulation was based on the PC-PAGMG. The second formulation produced the best results. Furthermore, a GRASP and an ILS metaheuristics were implemented, and both used the VND as an improvement method. The achieved results with the ILS have excelled both in cost of the solution found as at run time. The proposed exact method uses the results of the metaheuristics as input. This method achieved the best results among the proposed algorithms when the ILS metaheuristic was used. Computational experiments are presented to illustrate the efficiency of the methods proposed.

Keywords: Metaheuristics. Mathematical Programming. Generalized Minimum Spanning Tree Problem.

Sumário

Lista de Figuras	viii
Lista de Tabelas	ix
Lista de Abreviaturas e Siglas	x
1 Introdução	1
1.1 Motivação	3
1.2 Objetivos e Contribuições	4
1.3 Organização do Texto	4
2 O Problema da Árvore Geradora Mínima Generalizado com Prêmios nos Vértices	6
2.1 Problema da Árvore Geradora Mínima Generalizado	6
2.2 Problema da Árvore Geradora Mínima	8
2.3 Revisão da Literatura	11
2.4 Definição do Problema	16
2.5 Resumo	17
3 Metodologia Proposta	18
3.1 Formulações Matemáticas para o PAGMG-P	18
3.2 Algoritmos Heurísticos	22
3.2.1 Representação e Avaliação de uma Solução	22
3.2.2 Construção de uma Solução Inicial	22
3.2.3 Estruturas de Vizinhança	26

3.2.3.1	VND	28
3.2.4	Metaheurísticas	29
3.2.4.1	GRASP	30
3.2.4.2	ILS	31
3.3	Método Exato	32
3.4	Resumo	34
4	Resultados Computacionais	35
4.1	Ferramentas	35
4.2	Instâncias	36
4.3	Resultados Experimentais	36
4.3.1	GRASP vs ILS	39
4.4	Resumo	42
5	Conclusões e Trabalhos Futuros	43
	Referências	45
	Apêndice A – Instâncias	50
	Apêndice B – Resultados	53

Lista de Figuras

2.1	Uma solução do PAGMG.	7
2.2	Uma solução do L-PAGMG.	7
2.3	Uma solução do CP-PAGMG.	8
2.4	Ilustrando o funcionamento do algoritmo de PRIM.	10
2.5	Um exemplo de solução para o PAGMG-P.	17
3.1	Ilustrando a heurística construtiva.	24
3.2	Ilustrando a vizinhança <i>1-Opt</i>	27
3.3	Ilustrando a vizinhança <i>2-Opt</i>	27
4.1	Função de distribuição de probabilidade dos algoritmos atingirem uma so- lução alvo com a Instância <i>50al10</i>	41
4.2	Função de distribuição de probabilidade dos algoritmos atingirem uma so- lução alvo com a Instância <i>75al5</i>	41
4.3	Função de distribuição de probabilidade dos algoritmos atingirem uma so- lução alvo com a Instância <i>150al15</i>	41
4.4	Função de distribuição de probabilidade dos algoritmos atingirem uma so- lução alvo com a Instância <i>300al30</i>	42

Lista de Tabelas

4.1	Resultados quantitativos dos algoritmos apresentados no Capítulo 4.	38
4.2	Gap médio considerando a melhor solução encontrada para todas as instâncias.	38
4.3	Gap médio considerando as instâncias onde foram encontradas soluções ótimas.	39
4.4	Média dos tempos computacionais considerando as instâncias onde foram encontradas soluções ótimas.	39
4.5	Média dos tempos computacionais considerando todas as instâncias.	39
A.1	Instâncias geradas utilizadas neste trabalho: $ V $ representa o número de vértices, $ E $ representa o número de arestas e m representa o número de componentes.	50
A.2	Instâncias geradas utilizadas neste trabalho.	51
A.3	Instâncias geradas utilizadas neste trabalho.	52
B.1	Resultados dos algoritmos EX1, EX2, GRA, ILS, E2G e E2I, apresentados no Capítulo 4. A coluna Custo representa o custo da solução encontrada (em negrito: melhor solução encontrada; com “-”: solução não encontrada), T representa o tempo de execução (em segundos) e GAP representa o gap médio (em porcentagem) da solução do algoritmo em relação a melhor solução encontrada.	54
B.2	Resultados dos algoritmos apresentados no Capítulo 4.	55
B.3	Resultados dos algoritmos apresentados no Capítulo 4.	56
B.4	Resultados dos algoritmos apresentados no Capítulo 4.	57
B.5	Resultados dos algoritmos apresentados no Capítulo 4.	58
B.6	Resultados dos algoritmos apresentados no Capítulo 4.	59

Lista de Abreviaturas e Siglas

PAGM	: Problema da Árvore Geradora Mínima;
PAGMG	: Problema da Árvore Geradora Mínima Generalizado;
L-PAGMG	: Versão “at Least” do Problema da Árvore Geradora Mínima Generalizado;
CP-PAGMG	: Problema da Árvore Geradora Mínima Generalizado com Coleta de Prêmios;
PAGMG-P	: Problema da Árvore Geradora Mínima Generalizado com Prêmios nos Vértices;
GRASP	: Greedy Randomized Adaptive Search Procedure;
ILS	: Iterated Local Search;
VND	: Variable Neighborhood Descent.

Capítulo 1

Introdução

Esta dissertação aborda uma variante do Problema da Árvore Geradora Mínima Generalizado (PAGMG). No PAGMG, um conjunto de vértices está dividido em componentes disjuntas e o objetivo é determinar uma árvore de custo mínimo que cubra exatamente um vértice de cada componente. O problema foi introduzido e provado ser $\mathcal{NP}$ -Difícil em [30]. Apesar da sua importância prática, a maior parte dos trabalhos relacionados mais relevantes é bastante recente, mas há um número restrito desses trabalhos de que se tem conhecimento [11].

Uma outra versão abordada na literatura e diretamente relacionada ao PAGMG é o problema de encontrar uma árvore que cubra pelo menos um vértice de cada componente, denominada L-PAGMG. Esse problema foi introduzido por [22] com outra denominação, Problema de Steiner em Grupos. Foi provado que o problema é um caso particular do Problema de Steiner em Grafos Generalizado [40].

Os autores em [15] propuseram uma variante para o PAGMG, o Problema da Árvore Geradora Mínima Generalizado com Coleta de Prêmios (CP-PAGMG), que é definido de maneira semelhante ao problema original, mas tem a característica adicional de que os vértices em cada componente competem entre si, oferecendo um prêmio se o vértice for selecionado. O objetivo é minimizar a diferença entre o custo total das arestas utilizadas e a soma total dos prêmios coletados nos vértices selecionados.

Neste trabalho, propomos uma outra variante do PAGMG que, assim como o CP-PAGMG, possui prêmios nos vértices, mas que se difere do modelo proposto em [15] tanto em restrições quanto em objetivo. A variante aqui proposta exige que seja adquirido um prêmio mínimo P_i em cada componente V_i e, enquanto o CP-PAGMG restringe a escolha de exatamente um vértice por componente, na variante proposta pode ser escolhido pelo

menos um vértice. Além disso, o objetivo da nova versão é minimizar o custo total da árvore solução, sem considerar os prêmios coletados.

A justificativa de se propor uma nova variante do PAGMG com prêmios nos vértices, é a de que em diversas possíveis aplicações reais é exigido um prêmio (ou capacidade) mínimo para cada componente V_i [41].

No contexto de aplicações para o PAGMG, estas podem ser encontradas na área de projeto de síntese de redes de telecomunicações, por exemplo, onde as redes regionais precisam ser interconectadas por uma árvore que contenha uma conexão para cada sub-rede. Para essa interconexão, um vértice de cada sub-rede deve ser selecionado como *gateway* [30].

Outra aplicação do PAGMG está relacionada com vários problemas de localização de instalações que precisam estar conectadas por meio de rodovias ou *links* de comunicação [7, 24]. Um exemplo acontece quando uma empresa quer estabelecer centros regionais de distribuição para suas lojas e precisa selecionar um ponto em cada região a fim de construir uma rede de comunicação interconectando esses centros [43].

Uma aplicação do L-PAGMG, descrita por [4], pode ser encontrado no projeto de redes de irrigação em ambientes desérticos, onde a escolha de pontos de irrigação deve ser feita de maneira que nenhuma parte do ambiente fique sem receber irrigação. Assim, a área a ser irrigada é dividida em áreas menores e o objetivo é encontrar uma rede de irrigação mínima que cubra todas as áreas, cada área com pelo menos um ponto de irrigação.

O CP-PAGMG pode ser aplicado no mesmo contexto do primeiro exemplo descrito para o PAGMG. No entanto, neste caso, os vértices a serem selecionados oferecem uma compensação monetária, ou um “prêmio”, se selecionado como o nó *gateway* para a sua região. O objetivo é minimizar o custo total de *links* usados para conectar as regiões compensando com a soma total de prêmios coletados a partir dos nós *gateway* selecionados para o projeto [15].

Exemplos de compensação monetária na seleção de nós em uma rede de telecomunicações surge em vários contextos do mundo real. Por exemplo, no projeto de redes de cabos submarinos interligando continentes diferentes: nem todos os países ou cidades inclusos na rota da rede podem ser conectados diretamente a ela. Isso se deve aos altos custos de um local ser ligado a uma rede deste porte. Consequentemente, os planejadores destas redes geralmente designam um local a ser selecionado a partir de um determinado conjunto de regiões que fazem parte de sua rota. Dados os potenciais benefícios monetários

associados a ser um local que está diretamente ligado a uma rede transcontinental de fibra óptica, com significativos benefícios econômicos, não é incomum que as cidades ou países disputem uns contra os outros a seleção como um local de interconexão. Estes incentivos monetários são geralmente sob a forma de créditos fiscais e abatimentos para o construtor ou operador da rede de telecomunicações [15].

1.1 Motivação

Considerando as versões do PAGMG apresentadas neste trabalho, pode-se afirmar que estas são análogas a muitos problemas da vida real. No entanto, elas ainda não se aplicam a algumas classes de problemas de redes mais específicos como, por exemplo, o problema de projetar uma rede de internet cabeada para uma cidade com o intuito de oferecer internet *wireless* à população. Sabendo que geralmente uma cidade é dividida em regiões, suponha que a empresa contratada para fazer este serviço deve atender pelo menos uma certa porcentagem da população de cada região de forma obrigatória e que a empresa utilizará pontos de acesso homogêneos em cada região de acordo com a demanda, considerando que cada cliente deve ser atendido por apenas um ponto de acesso. As regiões possuem tamanho e número de moradores distintos, logo o número mínimo de clientes a serem atendidos difere de uma região para outra. Dessa forma, o objetivo é encontrar uma árvore de custo mínimo para a rede cabeada que contenha um ou mais pontos de acesso à rede *wireless* para os moradores de cada região, e a soma dos moradores atendidos pelos pontos de acesso da região deve ser no mínimo uma dada porcentagem de moradores da mesma região.

Modelando o exemplo dado acima para o PAGMG, temos que as regiões da cidade são as componentes, os pontos de acesso à rede *wireless* são os vértices e a árvore a ser minimizada é formada pela rede cabeada, onde o comprimento de cada cabo é o custo de cada aresta. Seguindo a definição do PAGMG, o objetivo é encontrar uma árvore de custo mínimo que cubra exatamente um vértice de cada componente. Porém, o objetivo do exemplo dado não está sendo atendido, pois o objetivo do PAGMG busca apenas minimizar o custo da solução, além de limitar a escolha de um único vértice para cada componente. Apesar de que no L-PAGMG pode-se escolher mais de um vértice por componente, nele a escolha do vértice é feita considerando apenas a minimização dos custos, assim como no PAGMG. Logo, essa versão também não atende as restrições do exemplo dado. No mesmo contexto, pode-se observar que o CP-PAGMG também não atende ao objetivo do exemplo, visto que nele pode-se escolher apenas um vértice por

componente, e como as regiões possuem demandas diferentes, possivelmente alguma das regiões deixaria de ser atendida.

Dessa forma, observa-se a necessidade de definir problemas que sejam mais específicos e que possam representar ainda mais problemas da vida real. Com este intuito, este trabalho propõe uma nova variante para o Problema da Árvore Geradora Mínima Generalizado com Prêmios nos Vértices.

1.2 Objetivos e Contribuições

O objetivo geral deste trabalho é propor uma variante para o PAGMG com Prêmios nos Vértices que possa representar ainda mais problemas da vida real e propor algoritmos para essa variante.

Os objetivos específicos desse trabalho são:

Nova variante: Propor uma nova variante para o PAGMG com Prêmios nos Vértices.

Modelos para o problema: Desenvolver modelos matemáticos para a variante proposta.

Algoritmos heurísticos: Desenvolver heurísticas que gerem soluções viáveis para o problema e forneçam bons resultados.

Algoritmos exatos: Desenvolver algoritmos exatos que utilizem as soluções fornecidas pelas heurísticas.

Instâncias para a nova variante: Criar instâncias para a nova variante baseadas nas instâncias do PAGMG para os testes computacionais.

As principais contribuições desse trabalho estão no fato de que pelo PAGMG com Prêmios nos Vértices ser pouco explorado na literatura, o mesmo tende a ter muitas aplicações. Dessa forma, o estudo de uma nova variante para o problema irá possibilitar a análise de aplicações mais próximas da realidade.

1.3 Organização do Texto

Este trabalho está organizado da seguinte forma. No Capítulo 2, são apresentados os principais conceitos relacionados ao problema abordado. O Capítulo 3 apresenta os modelos

matemáticos e os algoritmos exatos e herísticos propostos neste trabalho. No Capítulo 4, são apresentados os resultados computacionais obtidos. E, por último, no Capítulo 5, são feitas as considerações finais e possíveis linhas de trabalhos futuros são apontadas.

Capítulo 2

O Problema da Árvore Geradora Mínima Generalizado com Prêmios nos Vértices

Neste capítulo são apresentados os principais trabalhos na literatura que abordam o PAGMG e a nova variante proposta é definida. O capítulo está organizado da seguinte forma. Inicialmente, o Problema da Árvore Geradora Mínima Generalizado e os problemas relacionados são definidos e discutidos. Em seguida, são listados os trabalhos que tiveram impactos relevantes no estudo de tais temas. Por fim, é apresentada a definição do problema proposto.

2.1 Problema da Árvore Geradora Mínima Generalizado

O Problema da Árvore Geradora Mínima Generalizado (PAGMG), como o nome sugere, é uma generalização do Problema da Árvore Geradora Mínima. Tal problema foi introduzido por [30] e também é denominado *Equality Generalized Minimum Spanning Tree Problem* (E-GMSTP).

O PAGMG pode ser definido da seguinte forma. Seja um grafo conexo, não direcionado e ponderado $G = (V, E)$, onde V representa o conjunto dos vértices e E o conjunto das arestas. Cada aresta $e \in E$ possui um custo associado $c_e \in \mathbb{R}^+$. O conjunto de vértices V é particionado em m componentes disjuntas e não vazias $V_1, \dots, V_m$. O objetivo do problema é determinar uma árvore de custo mínimo que cubra exatamente um vértice de cada componente.

Diferentemente do Problema da Árvore Geradora Mínima, vastamente abordado na literatura e resolvido de maneira eficiente em tempo polinomial [25], o PAGMG é classi-

ficado como NP-Difícil [30] e há um número restrito de trabalhos relacionados de que se tem conhecimento [11].

Na Figura 2.1, apresentada por [11], é ilustrado um exemplo de uma solução para o PAGMG em um grafo cujo conjunto de vértices está particionado em quatro componentes. No exemplo, as arestas $(5, 3)$, $(3, 7)$ e $(7, 11)$ compõem a solução, também dita Árvore Geradora Generalizada [11].

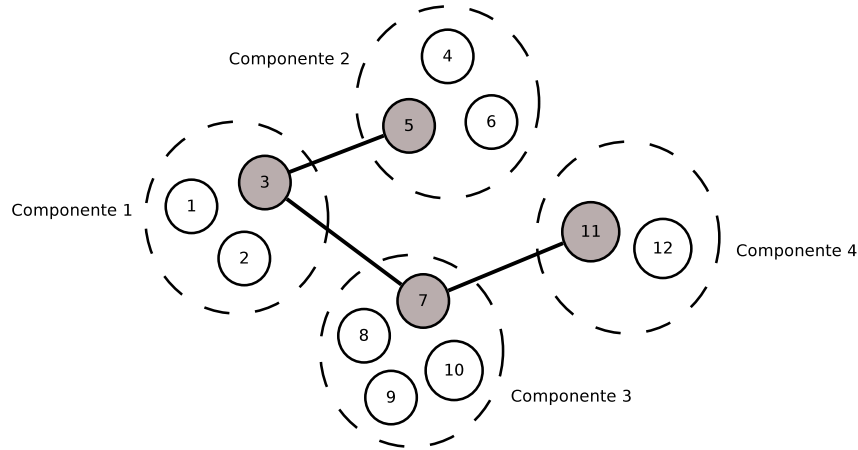


Figura 2.1: Uma solução do PAGMG.

Outra generalização do Problema da Árvore Geradora Mínima, também conhecida na literatura, é o problema de encontrar uma árvore de custo mínimo que cubra pelo menos um vértice de cada componente. Muitos autores denominam essa variante como L-PAGMG (L vem de “*at Least*”).

Na Figura 2.2 é exibido um exemplo de solução para o L-PAGMG. As arestas $(5, 3)$, $(3, 2)$, $(2, 8)$, $(8, 7)$, $(7, 10)$, $(10, 11)$ e $(11, 12)$ formam uma solução.

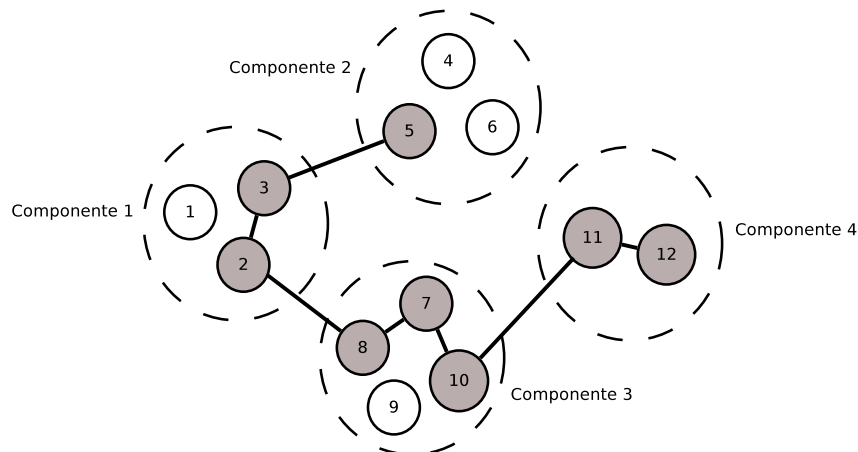


Figura 2.2: Uma solução do L-PAGMG.

Quando as intâncias são definidas sobre grafos completos e respeitam a desigualdade triangular, as soluções ótimas do L-PAGMG sempre terão apenas um vértice em cada grupo. Nesses casos, as duas generalizações do Problema da Árvore Geradora Mínima terão a mesma solução ótima. Uma conclusão direta disso é que o PAGMG é um caso particular do L-PAGMG [8].

Em [15] os autores propuseram o Problema da Árvore Geradora Mínima Generalizado com Coleta de Prêmios (CP-PAGMG), outra variante do PAGMG, definido de maneira semelhante ao problema original, mas com a característica adicional de que os vértices em cada componente competem entre si, oferecendo um prêmio se o vértice for selecionado. O objetivo do CP-PAGMG é minimizar a diferença entre custo total das arestas utilizadas e a soma total dos prêmios coletados nos vértices selecionados.

Um exemplo de solução do CP-PAGMG é exibido na Figura 2.3. Cada vértice v é associado a um prêmio p_v . Na figura, esta relação é representada por v/p_v . Assim, um dado vértice v possui o prêmio p_v . Em cada componente V_i , os vértices competem entre si oferecendo o prêmio caso ele seja selecionado. Nessa variante, apenas um vértice pode ser selecionado por componente. No exemplo, uma solução para o CP-PAGMG pode ser representada pelas arestas $(4, 2)$, $(2, 7)$ e $(7, 11)$.

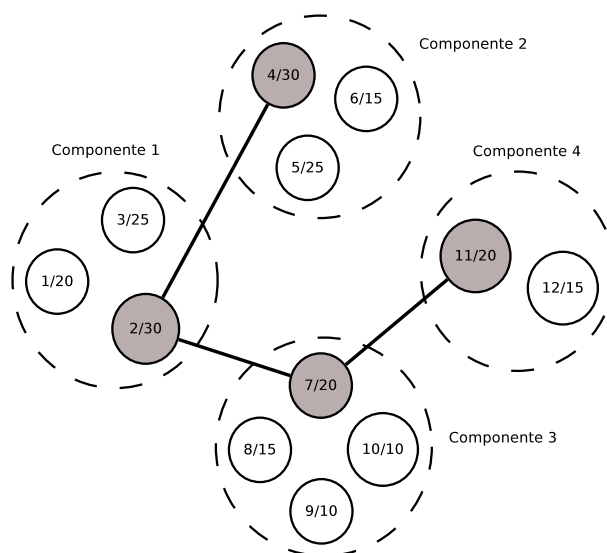


Figura 2.3: Uma solução do CP-PAGMG.

2.2 Problema da Árvore Geradora Mínima

O Problema da Árvore Geradora Mínima (PAGM) pode ser definido sobre um grafo conexo $G = (V, E)$, onde cada aresta $e \in E$ possui um custo $c_e \in \mathbb{R}$. O objetivo é encontrar uma

árvore T que cubra cada vértice em V de forma a minimizar $\sum_{e \in T} c_e$.

Esse problema é considerado um dos mais importantes em otimização combinatória, devido sua grande variedade de aplicações reais nas mais diversas áreas e, em muitos casos, surge como sub-problema para resolução de problemas mais complexos [1]. Dentre essas aplicações, pode-se citar a síntese de redes de transporte, energia, computadores e telecomunicações; agrupamento de dados; planejamento de produção; problemas de roteamento em geral; biologia computacional, etc.

Três algoritmos se destacam na resolução do problema, são eles: Kruskal [25], Prim [38] e Sollin [1]. Na prática, tais algoritmos são bastante eficientes e apresentam complexidade de pior caso polinomial, o que motiva a aplicação do Problema da Árvore Geradora Mínima também como parte da resolução de vários problemas de otimização em grafos. A seguir, tem-se a descrição do algoritmo de Prim que, além de ser um dos mais abordados na literatura, mostrou bons resultados sendo aplicado na resolução do PAGMG [11].

O algoritmo de Prim mantém uma árvore cobrindo um sub-conjunto S de V e adiciona a cada passo o vértice $v_i \in V \setminus S$ mais próximo de algum vértice $v_j \in S$. A cada iteração, v_i é selecionado através da identificação da aresta e de menor custo do corte $[S, V \setminus S]$. Assim, e é inserido na solução T e v_i é adicionado a S . O algoritmo termina quando $S = V$.

Prim executa $|V| - 1$ operações de consulta à aresta de custo mínimo no corte $[S, V \setminus S]$, o que resulta em um tempo de pior caso de $O(|V||E|)$, se todas as arestas forem analisadas a cada iteração. Utilizando uma *heap* para encontrar o vértice com menor distância ao seu predecessor, a complexidade do algoritmo se reduz para $O(|E| \log |V|)$.

O Algoritmo 2.1 apresenta o pseudocódigo de Prim. O vetor δ armazena o predecessor de cada vértice na árvore. A função *ProximoVertice()* retorna o vértice $v \in V \setminus S$ com menor valor de $c_{(v, \delta_v)}$. O conjunto $E(v)$ contém todas as arestas que partem de v . Considera-se que $c_{(v, v)} = 0$ e $c_{(v, 0)} = \infty$.

Algoritmo 2.1 PRIM

```

1: para  $i \in \{2, \dots, |V|\}$  faça
2:    $\delta_i \leftarrow 0$ ;
3: fim para
4:  $\delta_1 \leftarrow 1$ ;
5:  $T \leftarrow \emptyset$ ;
6:  $S \leftarrow \{1\}$ ;
7: enquanto  $|S| < |V|$  faça
8:    $v \leftarrow \text{ProximoVertice}(V \setminus S, \delta)$ ;
9:    $S \leftarrow S \cup v$ ;
10:   $T \leftarrow T \cup (v, \delta_v)$ ;
11:  para  $e(u, v) \in E(v)$  faça
12:    se  $c_e < c_{(u, \delta_u)}$  então
13:       $\delta_u \leftarrow v$ ;
14:    fim se
15:  fim para
16: fim enquanto

```

Na Figura 2.4 tem-se um exemplo de execução do algoritmo, apresentado por [11]. Inicialmente, pode-se considerar $S = \{1\}$. O corte $[S, V \setminus S]$ contém então duas arestas, $(1, 2)$ e $(1, 3)$, e o algoritmo seleciona a primeira delas. Assim, S passa a ter dois elementos, $\{1, 2\}$, e o corte $[S, V \setminus S]$ contém as arestas $(1, 3)$ e $(2, 3)$ e $(2, 4)$. A aresta $(2, 4)$ é selecionada por possuir o menor custo dentre as três. Nas próximas iterações, são adicionadas as arestas $(4, 3)$ e $(3, 5)$. Como $S = \{1, 2, 3, 4, 5\}$, o algoritmo encerra.

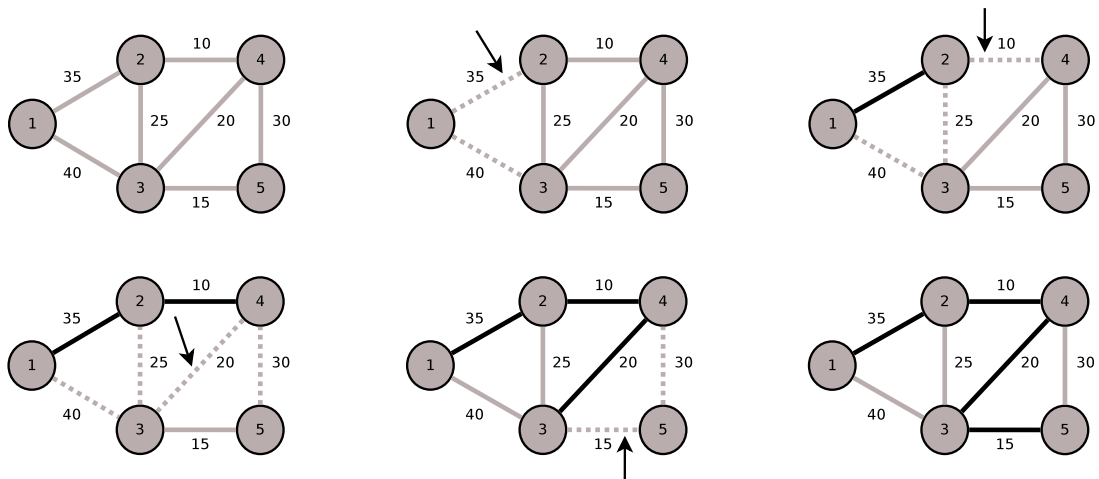


Figura 2.4: Ilustrando o funcionamento do algoritmo de PRIM.

2.3 Revisão da Literatura

Nesta seção pretende-se apresentar os principais trabalhos conhecidos na literatura sobre o PAGMG, suas versões “at least” e com coleta de prêmios.

O PAGMG foi primeiramente abordado por [30]. Através de uma redução ao Problema de Cobertura de Vértices, o autor mostra que o PAGMG é fortemente NP-Difícil. Não é possível construir um algoritmo aproximativo com tempo de pior caso polinomial para o problema, a não ser que $\mathbb{P} = \text{NP}$. Myung também propôs quatro formulações de programação inteira mista e um algoritmo *branch-and-bound* que resolveu instâncias de até 100 vértices.

Em 1999, os autores em [22] introduziram o primeiro trabalho sobre o L-PAGMG, com uma outra denominação, Problema de Steiner em Grupos. Foi provado que o problema é um caso particular do Problema de Steiner em Grafos Generalizado [40].

Os autores em [4] foram um dos poucos a proporem métodos heurísticos para o L-PAGMG. Os autores propuseram duas formulações de programação inteira e um algoritmo para encontrar limites inferiores baseados em relaxação lagrangeana. Nesse mesmo trabalho, também são propostas quatro heurísticas e um algoritmo genético. Os testes foram realizados sob 20 instâncias geradas pelos autores, com número de vértices variando entre 25 e 500, número de componentes entre 4 e 50 e arestas com pesos aleatórios. Os resultados indicaram desempenho bem melhor da metaheurística em relação às outras heurísticas, apesar de seu tempo de execução ser também bem maior. Posteriormente, [6] em 2001 comparou os resultados deste algoritmo genético com os valores ótimos das instâncias e demonstrou que as soluções se encontraram, em média, a 6.53% da otimalidade. Apesar desses algoritmos terem sido propostos para o L-PAGMG, os mesmos podem ser aplicados a quaisquer instâncias do PAGMG, com exceção apenas das instâncias com distâncias euclidianas.

Em [35] uma formulação de programação inteira foi proposta, alternativa a de [30], utilizada para resolver instâncias com até 75 vértices. Esse trabalho também descreveu um algoritmo de relaxação que apresentou bons resultados preliminares. Os mesmos autores de [37] demonstraram, através de um algoritmo polinomial, que o PAGMG pode ser aproximado por um fator de 2ρ se o número de elementos das componentes for limitado por ρ . Tal algoritmo é baseado no método descrito por [28] para o *Vertex Weighted Steiner Tree Problem*.

Feremans é uma das autoras que mais contribuiu para o PAGMG. Com Laporte e Labbé [8], a autora mostrou que, como o problema se trata de um caso particular do L-PAGMG, algumas demonstrações de [4] derivam diretamente do trabalho de [30]. Além disso, eles demonstraram que três das quatro formulações propostas por [4] estavam incorretas e propuseram mais uma formulação válida para o L-PAGMG.

Os autores em [9] fizeram uma análise comparativa de oito formulações do problema, algumas delas propostas por [30]. Os modelos são: duas formulações baseadas em propriedades de árvore propostas por [30] (*Undirected Cutset* e *Undirected Subpacking*); suas correspondentes para o caso direcionado, sendo uma de [30] (*Directed Cutset*) e uma proposta pelos próprios autores (*Directed Subpacking*); uma formulação baseada em controle de fluxo (*Multiflow Formulation*), de [30], e sua projeção em um espaço mais restrito (*Flow Cut Formulation*); e mais uma formulação de [30] (*Undirected Cluster Subpacking*) e sua correspondente para a versão direcionada (*Directed Cluster Subpacking*).

Os autores provaram que quatro dessas formulações são equivalentes e, além disso, provêm limites duais melhores em relação às outras quatro, e a formulação *Undirected Cluster Subpacking* é a mais compacta em termos de variáveis.

Em sua tese de doutorado [6], Feremans fez uma investigação da estrutura poliédrica do PAGMG. A tese também propôs um algoritmo *branch-and-cut*, que utiliza a formulação *Undirected Cluster Subpacking*, e quatro famílias de desigualdades válidas: desigualdades de Hammcock, de ciclos ímpares, de casamento de ciclos ímpares e de “buracos” ímpares. Com a análise poliédrica, pôde-se provar que algumas das desigualdades da formulação juntamente com as de Hammcock definem facetas da envoltória convexa do problema. Comparando com o algoritmo de [30], Feremans apresentou limites melhores e conseguiu resolver instâncias bem maiores.

A tese de Feremans também propôs uma formulação para o L-PAGMG, reconhecida-mente “fraca”, pela autora, em termos de relaxação linear, dedução que pode ser feita por analogia ao trabalho de [28]. Mas as restrições de corte tornam o procedimento fácil de ser resolvido por meio do problema de fluxo máximo.

Além disso, Feremans apresenta ainda uma forma de transformar instâncias do L-PAGMG em instâncias do PAGMG. Foi provado que a solução ótima da instância gerada por tal transformação pode ser convertida em uma solução viável do L-PAGMG. Assim, tal solução é um limite superior para a solução ótima do L-PAGMG. O *branch-and-cut* proposto também foi testado em instâncias do L-PAGMG. Os resultados foram comparados com o algoritmo genético de [4] e o *branch-and-cut* encontrou a solução ótima de

todas as instâncias, em um tempo computacional razoável.

Em [24] foi apresentada mais uma aplicação para o L-PAGMG, dessa vez em física. Em 2002, [39] demonstrou que o PAGMG pode ser modelado como um Problema de Steiner em Grafos com a inserção de vértices fictícios em cada componente. Segundo o autor, a relaxação linear da formulação resultante é equivalente às quatro formulações apresentadas por [9].

Em 2003, [44] apresentaram uma metaheurística baseada no método colônia de formigas para o L-PAGMG. O método mostrou-se eficiente em produzir boas soluções aproximadas. [36], em 2004, demonstrou que além de ser NP-Difícil, o PAGMG não pode ser resolvido em tempo polinomial nem se o grafo for uma árvore, diferentemente do que acontece com vários problemas NP-Completo [14]. Além disso, a autora propõe um algoritmo exato baseado na relaxação de [30], denominado *Rooting Procedure*. O algoritmo foi testado em instâncias com custos aleatórios apresentadas por [30] e comparado com seu trabalho e com o de [6]. O *Rooting Procedure* encontrou a solução ótima de todas as instâncias testadas, com até 240 vértices, algumas delas ainda em aberto.

Os autores de [10], em 2005, discutiram a relação entre o PAGMG e o L-PAGMG e apresentaram melhorias no *branch-and-cut* proposto anteriormente. Os testes foram realizados para comparar quatro versões do *branch-and-cut* para o PAGMG e L-PAGMG. As instâncias utilizadas foram as mesmas de [4]. Os resultados mostraram que as mudanças trouxeram melhoras efetivas no comportamento do *branch-and-cut*, principalmente em relação ao tempo computacional. Também em 2005, [16] propuseram um algoritmo para o cálculo de limites duais do PAGMG, além de uma comparação minuciosa de alguns algoritmos construtivos e duas heurísticas. No mesmo ano, [19] apresentaram uma abordagem de busca em vizinhança variável (VNS) para solução do PAGMG. Testes em instâncias com custos aleatórios e distâncias euclidianas indicaram, especialmente em casos com muitos vértices por componente, vantagens significativas desse algoritmo sobre as abordagens metaheurísticas publicadas anteriormente.

Ainda em 2005, [15] introduziram o PAGMG com Coleta de Prêmios (CP-PAGMG). Os autores descreveram várias estratégias heurísticas, incluindo busca local e um algoritmo genético. Além disso, é apresentado um algoritmo *branch-and-cut* simples e computacionalmente eficiente. O estudo computacional indica que esse algoritmo encontra as melhores soluções para grafos com até 200 vértices dentro de duas horas de tempo de CPU, enquanto os procedimentos de busca heurística encontram rapidamente soluções quase ótimas para todas as instâncias testadas.

Em 2006, [17] apresentaram vários algoritmos para o cálculo de limites duais para o L-PAGMG, todos baseados em uma formulações para o Problema de Steiner. Os algoritmos foram comparados com os trabalhos de [4] e [44], utilizando as instâncias apresentadas por [4] e instâncias dos próprios autores. No mesmo ano, [31] propuseram um procedimento para o cálculo de limites duais e um algoritmo de busca tabu para as duas generalizações, além de instâncias novas e de maior porte para o PAGMG. Os testes de comparação com a literatura foram realizados com 101 instâncias propostas pelos autores. O algoritmo tabu foi comparado com o proposto por [16], onde em todas as instâncias a proposta de Oncan atingiu resultados melhores ou iguais aos de Golden. O mesmo algoritmo também foi adaptado para o L-PAGMG e comparado com o trabalho de [17] sobre as instâncias propostas por [4]. O algoritmo tabu não superou os resultados de [17], mas os autores consideraram os resultados promissores.

Em 2007, [11] propôs versões da metaheurística GRASP, para o PAGMG, que utilizam mais de um algoritmo construtivo de forma adaptativa, além de mecanismos adicionais de aprimoramento, como reconexão de caminhos e busca local iterada. Os algoritmos propostos resultaram em soluções melhores que algumas das melhores soluções conhecidas, em um tempo computacional razoavelmente baixo. Também foi implementado um algoritmo de geração de cortes baseado em formulação para o Problema de Steiner em Grafos Direcionado. Com esse algoritmo, foi possível encontrar limites duais para 82 instâncias em aberto.

No mesmo ano, [32] descreveu um algoritmo exato de tempo exponencial para o CP-PAGMG, onde foram apresentadas várias formulações de programação inteira e inteira mista. Além disso, foram estabelecidas relações entre os politopos correspondentes às suas relaxações lineares e foi apresentado um procedimento de solução eficiente, que encontra soluções ótimas para o CP-PAGMG para grafos com até 240 vértices. Ainda em 2007, [34] apresentaram uma nova abordagem para os problemas de projeto de redes generalizados com base na distinção entre conexões globais (conexões entre componentes) e conexões locais (conexões entre os vértices de componentes diferentes).

Em 2008, [18] combinaram VNS e Programação Linear Inteira. Essa combinação foi feita através de uma abordagem do VNS que utiliza três tipos diferentes de vizinhança. Duas delas trabalham de forma complementar, a fim de maximizar a efetividade da busca. A terceira vizinhança aplica Programação Inteira Mista para otimizar partes locais dentro de árvores solução candidatas. Testes em instâncias com custos aleatórios e distâncias euclidianas com até 1280 vértices, indicaram, especialmente em casos com muitos vértices

por componente, vantagens significativas desse algoritmo sobre as abordagens metaheurísticas publicadas anteriormente.

Em 2010, [23] apresentaram um algoritmo de busca baseado em um conjunto dinâmico de candidatos. Extensos experimentos indicaram que o algoritmo proposto superou ligeiramente os algoritmos heurísticos existentes, em termos de qualidade da solução. Em 2011, [33] propuseram um algoritmo memético para resolver o PAGMG, que combina o conceito de população de algoritmos genéticos com um método de busca local rápido. O algoritmo proposto é competitivo com outras heurísticas publicadas até à data, tanto na qualidade da solução quanto no cálculo do tempo computacional. Os resultados computacionais para vários *benchmarks* dos problemas são relatados e os resultados apontam que o algoritmo memético é um método adequado para explorar o espaço de busca de um problema tão complexo e que esse algoritmo encontra boas soluções em um período de tempo razoável.

Os autores de [20], em 2012, apresentaram um conceito de melhorar um algoritmo evolutivo (AE) com um arquivo contendo informações completas das soluções avaliadas durante a otimização, a fim de detectar duplicatas e convertê-las de forma eficiente em soluções ainda desconsideradas. Além disso, os autores propuseram duas abordagens distintas para resolver o PAGMG. As abordagens funcionam com uma “camada superior” e uma “camada inferior” da solução. A solução da camada superior é desenvolvida pelo processo evolutivo, enquanto que a solução ótima do problema da camada inferior que corresponde a um ponto de pesquisa específico da camada superior, pode ser encontrado em tempo polinomial, utilizando algoritmos determinísticos. Os resultados dos testes computacionais indicam a força deste conceito.

Em 2013, [3] propuseram a análise de tempo de execução de algoritmos evolutivos para problemas de otimização de dois níveis. Para isso, foi realizada uma análise das duas abordagens apresentadas por [20] para o PAGMG, no contexto da complexidade parametrizada, em relação ao número de componentes, que distingue as abordagens pela representação de soluções possíveis. Os resultados mostraram que a abordagem em dois níveis do AE trabalhando com a representação de vértices geradores não é um AE de parâmetro fixo para o problema, enquanto que a representação de estrutura global permite resolver o problema com parâmetro de tempo fixo. Além disso, os autores apresentaram vários exemplos para cada abordagem e mostraram que as duas são altamente complementares, provando que elas resolvem suas instâncias de forma muito eficiente.

2.4 Definição do Problema

O Problema da Árvore Geradora Mínima Generalizado com Prêmios nos Vértices (PAGMG-P) aqui proposto é uma variante do PAGMG e sua definição é baseada no L-PAGMG, com a característica adicional de que cada vértice $v \in V$ possui um prêmio associado $p_v \in \mathbb{R}^+$ e cada componente V_i possui um prêmio mínimo P_i a ser adquirido.

Formalmente, o PAGMG-P é definido da seguinte forma. Seja um grafo conexo, não direcionado e ponderado $G = (V, E)$, onde V representa o conjunto dos vértices e E o conjunto das arestas. Para todo vértice $v \in V$, definimos $N(v) = \{j \mid (v, j) \text{ e } (j, v) \in E\}$ como o conjunto de vértices vizinhos a v . Cada aresta $e \in E$ possui um custo associado $c_e \in \mathbb{R}^+$ e cada vértice $v \in V$ possui um prêmio associado $p_v \in \mathbb{R}^+$. O conjunto de vértices V é particionado em m componentes disjuntas e não vazias $V_1, \dots, V_m$. O objetivo do PAGMG-P é determinar uma árvore de custo mínimo que cubra pelo menos um vértice de cada componente V_i , cuja soma dos prêmios dos vértices selecionados em cada componente V_i deve ser maior ou igual a um valor predefinido P_i (prêmio mínimo da componente V_i).

Diferentemente do PAGMG, que é amplamente abordado na literatura, de nosso conhecimento não existem trabalhos sobre o PAGMG-P aqui proposto até o presente momento.

Modelando o exemplo dado na Seção 1.1, temos as componentes representando as regiões da cidade, os vértices representando os pontos de acesso à rede *wireless* e as arestas representando os cabos de rede e formando a árvore a ser minimizada. O número de moradores atendidos por um ponto de acesso é representado pelo valor do prêmio no vértice, e o número mínimo obrigatório de moradores a ser atendido em uma região é representado pelo prêmio mínimo da componente. Assim, o objetivo é determinar uma rede cabeada de custo mínimo que tenha pelo menos um ponto de acesso à rede *wireless* em cada região, sendo que a soma dos moradores atendidos pelos pontos de acesso em uma dada região deve ser maior ou igual ao número mínimo obrigatório de moradores atendidos da mesma região.

Uma aplicação interessante da literatura para o modelo aqui proposto é encontrado no Instituto Brasileiro de Geografia e Estatística (IBGE), onde um grafo, que modela uma região, é particionado em componentes, as quais são classificadas em cidades ou bairros, cada uma com uma capacidade mínima exigida [42].

Um exemplo de instância para a variante proposta do PAGMG-P é ilustrado na Figura 2.5. Cada vértice v é associado a um prêmio $p_v \geq 0$. Na figura, esta relação é representada

por v/p_v . Assim, um dado vértice v possui o prêmio p_v . Em cada componente V_i , a soma dos prêmios nos vértices presentes na árvore solução deve ser no mínimo P_i e o custo total da árvore deve ser minimizado. No exemplo, uma solução obtida pode ser representada pelas arestas $(5, 3)$, $(3, 2)$, $(2, 7)$, $(7, 8)$, $(7, 10)$, $(7, 11)$ e $(11, 12)$.

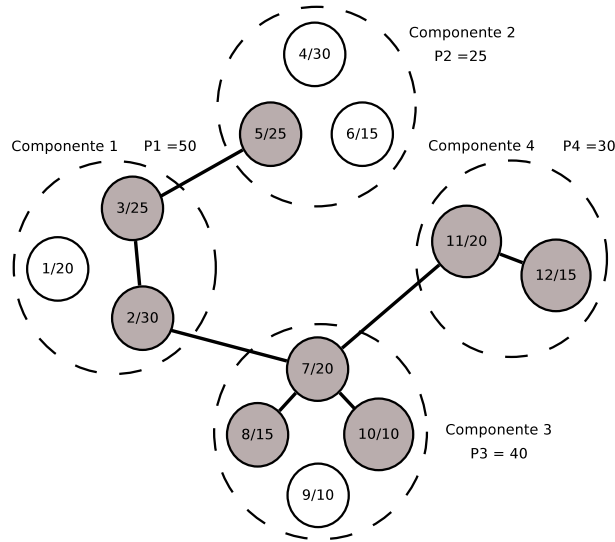


Figura 2.5: Um exemplo de solução para o PAGMG-P.

2.5 Resumo

Este capítulo apresentou uma descrição do Problema da Árvore Geradora Mínima Generalizado e seus problemas relacionados. Também foi feita uma revisão sucinta dos trabalhos sobre o tema encontrados na literatura. Em seguida a variante proposta foi definida. O capítulo seguinte traz os algoritmos propostos e implementados no presente trabalho para o PAGMG-P.

Capítulo 3

Metodologia Proposta

Neste capítulo é apresentada a metodologia proposta na resolução do problema abordado. Inicialmente foram definidas formulações matemáticas para o problema. Em seguida, foram desenvolvidas heurísticas para construção de uma solução inicial e estruturas de vizinhança. Posteriormente, foram implementadas duas metaheurísticas, são elas: GRASP e ILS. Além dos resultados produzidos por tais metaheurísticas, as mesmas também foram utilizadas na separação das restrições de subciclo do modelo matemático, gerando outros algoritmos. Os resultados obtidos com cada algoritmo bem como a análise dos mesmos são exibidos no Capítulo 4.

3.1 Formulações Matemáticas para o PAGMG-P

Até o momento não se tem conhecimento de trabalhos sobre o problema abordado, nem da existência de formulação matemática na literatura. Por isso, esse trabalho propõe duas formulações para o problema. Tais formulações se diferenciam, principalmente, pelas diferentes abordagens das restrições de eliminação de sub-ciclos. Os resultados obtidos com ambas as formulações são apresentados no Apêndice B.

A primeira formulação proposta é baseada em [6], onde é definido um modelo matemático para o L-PAGMG.

Na formulação representada por (3.1) - (3.9): c_{ij} representa o custo fixo da aresta (i, j) ; u_{ij} são variáveis binárias e recebem valor 1 se a solução contém a aresta entre os vértices i e j , e valor 0 caso contrário; se o vértice i pertence a solução, então a variável binária z_i vale 1, caso contrário z_i vale 0; p_i representa o prêmio fixo associado ao vértice i e P_k representa o prêmio mínimo fixo a ser adquirido na componente V_k .

$$\text{Minimizar } \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \quad (3.1)$$

Sujeito a:

$$\sum_{i \in V_k} z_i \geq 1 \quad \forall \quad k = 1, \dots, m, \quad (3.2)$$

$$\sum_{(i,j) \in E} u_{ij} = \sum_{i \in V} z_i - 1, \quad (3.3)$$

$$\sum_{i \in V_k} p_i z_i \geq P_k \quad \forall \quad k = 1, \dots, m, \quad (3.4)$$

$$\sum_{(i,j) \in E(S)} u_{ij} \leq \sum_{i \in S - \{l-1\}} z_i \quad \forall \quad S \subset V, 2 \leq |S| < n, \forall l \in S, \quad (3.5)$$

$$u_{ij} \leq z_i \quad \forall \quad (i, j) \in E, i \in V, \quad (3.6)$$

$$u_{ij} \leq z_j \quad \forall \quad (i, j) \in E, j \in V, \quad (3.7)$$

$$u_{ij} \in \{0, 1\} \quad \forall \quad (i, j) \in E, \quad (3.8)$$

$$z_i \in \{0, 1\} \quad \forall \quad i \in V. \quad (3.9)$$

Em (3.1) temos a função objetivo que requer a minimização do custo total das arestas da árvore. As restrições (3.2) e (3.3) garantem que a árvore solução cobre pelo menos um vértice de cada componente. O conjunto de restrições (3.4) garante que o somatório dos prêmios dos vértices selecionados em cada componente V_k seja pelo menos P_k . As restrições (3.5) são as restrições de eliminação de sub-ciclos. As restrições (3.6) e (3.7) garantem que se uma aresta pertence à solução, então os vértices que a compõem também pertencem. E, por fim, as restrições (3.8) e (3.9) descrevem a integralidade das variáveis u e z . Sobre a implementação das restrições de eliminação de sub-ciclos, elas foram adicionadas ao modelo a medida que o solver encontrava uma solução, então um grafo era criado com pesos nas arestas, sendo o valor da aresta na solução, e um algoritmo de fluxo

máximo era executado para encontrar um corte mínimo.

A segunda formulação proposta é baseada na formulação para o CP-PAGMG, definida por [15]. Tal formulação considera uma árvore direcionada e é descrita pelas equações (3.10) - (3.21).

Nesta formulação, c_{ij} , u_{ij} , z_i , p_i e P_k são definidas igualmente à formulação anterior; w_{kij} são variáveis binárias que valem 1 se o vértice j é um antecessor do vértice i na árvore direcionada com raiz no vértice k , e valem 0 caso contrário.

$$\text{Minimizar } \sum_{\forall (i,j) \in E} c_{ij} u_{ij} \quad (3.10)$$

Sujeito a:

$$\sum_{i \in V_k} z_i \geq 1 \quad \forall \quad k = 1, \dots, m, \quad (3.11)$$

$$\sum_{(i,j) \in E} u_{ij} = \sum_{i \in V} z_i - 1, \quad (3.12)$$

$$\sum_{i \in V_k} p_i z_i \geq P_k \quad \forall \quad k = 1, \dots, m, \quad (3.13)$$

$$u_{ij} \leq z_i \quad \forall \quad (i,j) \in E, i \in V, \quad (3.14)$$

$$u_{ij} \leq z_j \quad \forall \quad (i,j) \in E, j \in V, \quad (3.15)$$

$$u_{ij} = w_{kij} + w_{kji} \quad \forall \quad (i,j) \in E, k \in V, \quad (3.16)$$

$$\sum_{j \in N(i)} w_{kij} \leq 1 \quad \forall \quad k \in V, i \in V - \{k\}, \quad (3.17)$$

$$w_{kkj} = 0 \quad \forall \quad k \in V, j \in N(k), \quad (3.18)$$

$$u_{ij} \in \{0, 1\} \quad \forall \quad (i,j) \in E, \quad (3.19)$$

$$z_i \in \{0, 1\} \quad \forall \quad i \in V, \quad (3.20)$$

$$w_{kij} \in \{0, 1\} \quad \forall \quad i, j, k \in V. \quad (3.21)$$

Em (3.10) temos a função objetivo que requer a minimização da árvore solução. As restrições (3.11) - (3.15) são definidas semelhantemente as restrições (3.2) - (3.7), exceto a (3.5), respectivamente. O conjunto de restrições (3.16) garante que na árvore direcionada com raiz no vértice k , ou o vértice j é antecessor do vértice i , ou o vértice i é antecessor do vértice j (i. e. ou o vértice j é “pai” de i na árvore ou vice-versa), caso a aresta (i, j) faça parte da solução. As restrições (3.17) garantem que cada vértice tem no máximo um “pai”, exceto o vértice raiz. O conjunto de restrições (3.18) garante que o vértice raiz não possui “pai”, ou seja, ele não possui antecessor. As restrições (3.16), (3.17) e (3.18) são as restrições de eliminação de sub-ciclos. E, por fim, as restrições (3.19), (3.20) e (3.21) descrevem a integralidade das variáveis u , z e w .

Sobre as restrições de eliminação de sub-ciclos, temos que cada conjunto de variáveis w_{kij} , para um determinado vértice k , representa um conjunto de variáveis que define uma árvore direcionada com raiz no vértice k , onde uma variável individual w_{kij} é igual a 1 se o vértice j é antecessor do vértice i na árvore direcionada com raiz no vértice k . As restrições (3.17) e (3.18), em conjunto com (3.12) e (3.16), garantem que cada vértice na árvore direcionada com raiz no vértice k tem no máximo um vértice antecessor, enquanto que o vértice k , pela restrição (3.18), não tem antecessor. As restrições (3.16) à (3.18) garantem que não existem ciclos, pois se houvesse um ciclo significaria que para algum vértice raiz k , e o conjunto de variáveis w_{kij} correspondente, existe um vértice com dois antecessores, ou um vértice raiz com um antecessor, o que não é válido segundo a definição das restrições.

Sabe-se que a forma imediata de resolver problemas de otimização combinatória seria simplesmente enumerar todas as soluções possíveis e selecionar a melhor, seguindo um critério de avaliação. No entanto, devido a complexidade e o dinamismo dos problemas práticos que, de maneira geral, possuem um grande número de soluções candidatas, esta listagem exaustiva torna-se inviável para a maioria dos casos. Por isso, o desenvolvimento de métodos que, ao seu final, forneçam uma solução de boa qualidade, mas não necessariamente garantam ser a melhor solução existente, torna-se indispensável. Com este intuito, dentre os métodos de solução utilizados na literatura, destacam-se os algoritmos heurísticos.

3.2 Algoritmos Heurísticos

Segundo [46], heurísticas são procedimentos dedicados a resolução de problemas cuja estrutura pode ser interpretada e explorada de forma inteligente, permitindo assim que uma solução razoavelmente boa seja obtida por meio de uma abordagem intuitiva. As duas categorias de heurísticas mais comumente encontradas na literatura são as de Construção e Busca Local. A ideia básica de uma heurística construtiva é de incrementalmente adicionar componentes individuais a uma solução parcial até obter uma solução completa viável. Diferentemente das heurísticas construtivas, as heurísticas de busca local tentam melhorar uma solução viável ao manipular os elementos que a compõem. Devido a estas características, é comum a utilização destes dois tipos de heurísticas de forma combinada [47]. As metaheurísticas baseiam-se nestas ideias de combinar heurísticas.

3.2.1 Representação e Avaliação de uma Solução

Na abordagem proposta, uma solução s para o PAGMG-P é composta por uma sequência de tuplas (i, j) que correspondem aos identificadores das arestas $(i, j) \in E$, onde (i, j) compõe a solução se somente se existe uma aresta entre os vértices $i, j \in V$.

A solução s é avaliada por uma função f , dada na equação (3.22), que deve ser minimizada.

$$f(s) = \sum_{\forall e \in s} c(e) \quad (3.22)$$

Onde f é a função que avalia a solução s e $c(e)$ representa o custo da aresta $e = (i, j) \in s$ sendo i e j vértices de V .

3.2.2 Construção de uma Solução Inicial

Uma heurística de construção gera uma solução viável de um problema de forma incremental. Passo a passo um componente individual (por exemplo vértices, arestas, etc.) é escolhido e em seguida inserido na solução até gerar uma solução viável. No método guloso procura-se o melhor componente. Ou seja, dado algum critério de avaliação, o componente escolhido é o melhor candidato.

Se tratando das soluções finais geradas por esse tipo de heurística, como não há nenhum componente aleatório, dada uma instância, a cada execução do algoritmo encontra-

se a mesma solução (algoritmo determinístico). Para obter diferentes soluções, consequentemente soluções melhores, é introduzida uma componente aleatória.

A heurística de construção aleatorizada é uma tentativa de conseguir soluções distintas para uma mesma instância. Assim, tais soluções podem ser avaliadas e a melhor solução escolhida, seguindo algum critério de avaliação. Em geral, essa heurística apresenta resultados melhores do que a construção sem a componente aleatória, pois o conjunto de soluções da mesma é mais diversificado.

Neste trabalho, o algoritmo implementado para construir uma solução inicial foi adaptado do algoritmo de Prim, apresentado na Seção 2.2 para o Problema da Árvore Geradora Mínima. [6] foram os primeiros autores a citar a possibilidade dessa adaptação. O algoritmo adaptado pouco se difere do original. Em sua versão para o PAGMG-P, o algoritmo procede com os vértices das componentes de forma semelhante com que procede com os vértices na versão para o PAGM.

Ao início do algoritmo, a solução T é composta apenas por um vértice v de uma componente V_i . O vértice $v \in V_i$ é um dos vértices que compõe a aresta de menor custo em E . A cada passo, uma aresta de menor custo é inserida em T , agregando à solução ou um vértice da mesma componente V_i ou um vértice de uma componente V_j ainda não coberta. Na primeira iteração são analisadas todas as arestas entre os vértices de V_i e das outras componentes. Nas iterações seguintes, considera-se apenas as arestas que unem os vértices presentes em T e os vértices de componentes que ainda não fazem parte de T ou que ainda não atingiram o prêmio mínimo exigido. O procedimento termina quando todas as componentes estão cobertas pela árvore e em cada componente foi atingido o prêmio mínimo exigido.

Para entender melhor o funcionamento dessa heurística, tomemos a Figura 3.1. O grafo é o mesmo utilizado em exemplos anteriores. Ao início da execução, considere $T = \emptyset$. No exemplo dado, $i = 1$, ou seja, o objetivo é encontrar a aresta de menor custo do grafo para, assim, selecionar um dos vértices que compõe essa aresta (Figura 3.1(a)). A partir daí, verifica-se se o prêmio mínimo foi atingido na componente do vértice selecionado. No exemplo, a componente 1. Caso o prêmio mínimo tenha sido atingido, então na próxima iteração considera-se apenas as arestas que unem os vértices de T às componentes ainda não cobertas. Essa verificação ocorre a cada iteração.

Na Figura 3.1(b), todas as arestas que partem do vértice selecionado são analisadas e (2,3) é selecionada (Figura 3.1(c)). Dessa forma, o prêmio mínimo da componente 1 é atingido, logo as arestas internas da componente não serão mais consideradas nas

próximas iterações. Na iteração seguinte, tem-se $T = \{(2, 3)\}$ e procura-se a menor aresta que agregue outro vértice a T . Após analisadas todas as arestas, a aresta $(3, 5)$ é selecionada e na Figura 3.1(d) $T = \{(2, 3), (3, 5)\}$. Com isso, o prêmio mínimo é atingido na componente 2. O algoritmo procura o vértice mais próximo de T e adiciona $(2, 7)$ à solução. Nas iterações seguintes, o algoritmo adiciona as arestas $(7, 10)$ (Figura 3.1(f)), $(7, 11)$ (Figura 3.1(g)), $(11, 12)$ (Figura 3.1(h)) e $(7, 8)$ (Figura 3.1(i)). O algoritmo retorna a árvore $T = \{(2, 3), (3, 5), (2, 7), (7, 10), (7, 11), (11, 12), (7, 8)\}$ ilustrada na Figura 3.1(i).

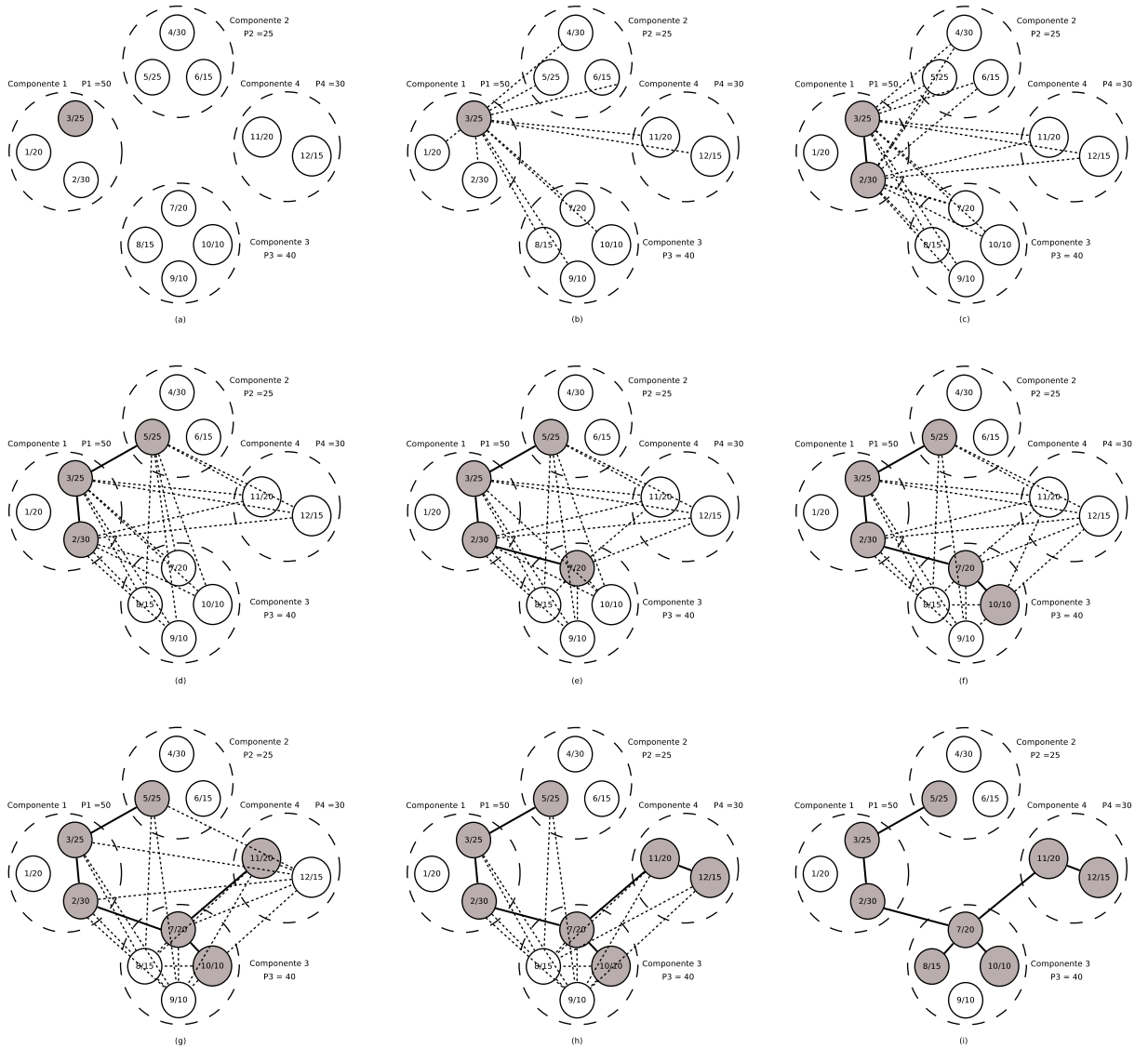


Figura 3.1: Ilustrando a heurística construtiva.

Pode-se perceber que a heurística apresentada é um algoritmo guloso, pois a cada iteração a “melhor” aresta candidata é analisada. Assim, foi implementada também uma versão aleatória da heurística construtiva, que seleciona cada aresta aleatoriamente a partir de um subconjunto $L \subset E$. Esse subconjunto é chamado de Lista de Candidatos

Restrita (LCR). A LCR é criada a cada iteração, antes da seleção de cada aresta.

Chamemos de c_{min} e c_{max} respectivamente o menor e o maior custo das arestas ainda não avaliadas durante o processo de construção. A LCR é composta de todas as arestas cujo custo seja menor que $d_{max} = c_{min} + \alpha(c_{max} - c_{min})$. O valor de α é estabelecido no intervalo $[0, 1]$ e indica o quão gulosa será a seleção das arestas. É fácil perceber que se $\alpha = 0$, o algoritmo precederá da mesma forma que a adaptação de Prim descrita anteriormente. [11] analisaram valores de α entre 0.05 e 0.9, onde os melhores resultados ocorreram para $0.05 \leq \alpha \leq 0.3$. Sendo assim, neste trabalho, o valor de α é selecionado aleatoriamente no intervalo de $[0.05, 0.3]$ antes de cada execução do algoritmo.

O pseudocódigo da versão aleatória está apresentado no Algoritmo 3.1. O algoritmo é caracterizado como *first improvement* e o número de execuções máximo ($max_iteracoes$) foi definido, empiricamente, igual a 10. A cada iteração $i, i \in \{1, \dots, max_iteracoes\}$, uma solução T é construída. O conjunto K contém todos os vértices que podem ser selecionados na iteração corrente. A função *ConstroiLCR()* insere na LCR L os vértices de K que estão à distância máxima d_{max} de algum vértice de T . A função *Rand()* escolhe aleatoriamente uma aresta $e(u, v) \in L$, de acordo com as seguintes situações: (a) u não é coberto por T e v o é; ou (b) u é coberto por T mas a componente de u ainda não atingiu o prêmio mínimo. A partir da inserção de e em T , a componente V^u passa a ser coberta pela solução, dado que V^u indica o índice da componente de que u faz parte. Assim, nas próximas iterações somente a situação (b) será analisada para essa componente. Cada árvore T é concluída quando K não possui mais vértices, ou seja, quando $|T|$ possui vértices que cobrem todas as componentes e que, somado os prêmios adquiridos em cada vértice, o prêmio mínimo de cada componente foi atingido. *Somatorio()* retorna a soma dos prêmios adquiridos em uma dada componente até o momento e *PremioMinimo()* retorna o valor fixo do prêmio mínimo a ser atingido na componente. A função $c()$ calcula $\sum_{e \in T} c_e$. O algoritmo retorna a melhor solução T^* dentre as $max_iteracoes$ soluções construídas.

Algoritmo 3.1 ADAPTAÇÃO DE PRIM ALEATÓRIA

```

1:  $T^* \leftarrow \emptyset$ ;
2: para  $i \in \{1, \dots, max\_iteracoes\}$  faça
3:    $T \leftarrow \emptyset$ ;
4:    $K \leftarrow V$ ;
5:   enquanto  $K \neq \emptyset$  faça
6:      $L \leftarrow ConstroiLCR(K, T, \alpha)$ ;
7:      $e(u, v) \leftarrow Rand(L)$ ;
8:      $T \leftarrow T \cup e$ ;
9:     se  $Somatorio(V^u) \geq PremioMinimo(V^u)$  então
10:       $K \leftarrow K \setminus V^u$ ;
11:   fim se
12: fim enquanto
13: se  $c(T) < c(T^*)$  então
14:    $T^* \leftarrow T$ ;
15: fim se
16: fim para

```

3.2.3 Estruturas de Vizinhança

Após a construção de uma solução viável s^* , a busca local “navega” iterativamente pelo espaço de busca movendo-se, em cada passo, de uma solução s^* para uma solução vizinha s . O espaço de busca pode ser visto como um grafo onde os vértices são as soluções e existem arestas entre pares de vértices associados a soluções vizinhas.

Seja S o espaço de busca do problema e s^* uma solução do problema. A função de vizinhança é uma função $N(s^*)$ que mapeia cada solução $s^* \in S$ para um subconjunto $N(s^*) \subseteq S$. Um elemento qualquer de $N(s^*)$ é denominado vizinho de s^* .

Um ótimo local é uma solução tão boa ou melhor do que qualquer das soluções vizinhas. Assim, em um problema de minimização, s^* é um ótimo local quando $f(s^*) \leq f(s), \forall s \in N(s^*)$. Já o ótimo global, ou solução ótima de s^* , é a melhor solução entre todas as vizinhanças, ou seja, $f(s^*) \leq f(s), \forall s \in S$.

A fim de explorar o espaço de soluções, foi desenvolvido um conjunto $N = \{N^1, N^2, \dots, N^t\}$ composto por t estruturas de vizinhança, tal que $N^t(s)$ é o conjunto de soluções na t -ésima vizinhança da solução s . As estruturas de vizinhança foram baseadas

na vizinhança ***t-Opt***, considerada clássica na literatura que aborda o PAGMG. A primeira estrutura de vizinhança é a ***1-Opt***.

- ***1-Opt*** - N^1 : 1 aresta é removida da árvore e outra aresta é criada para gerar uma nova árvore solução. O movimento no qual a aresta (3, 5) é removida da solução e a aresta (3, 4) é criada é ilustrado na Figura 3.2.

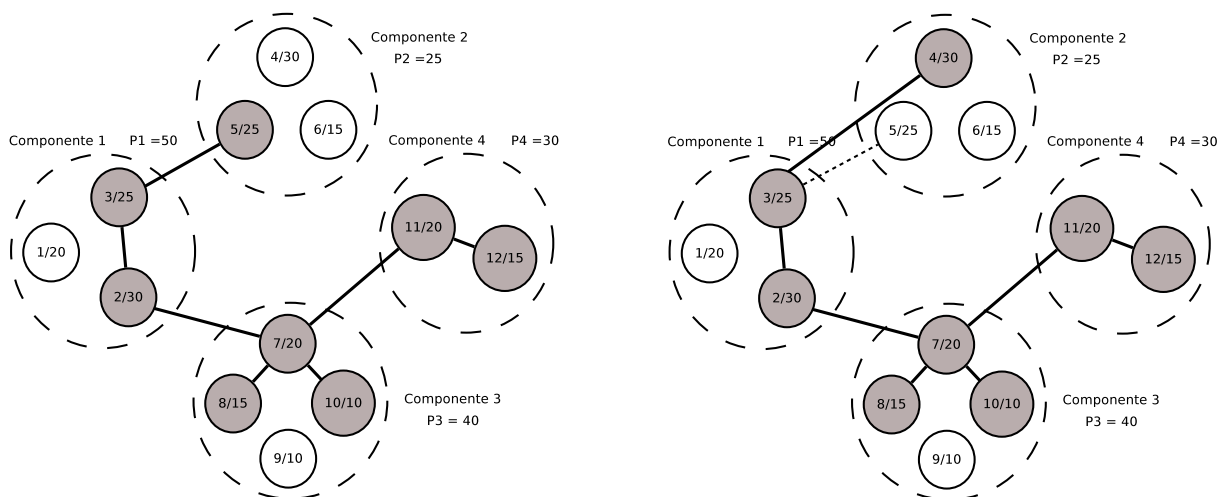


Figura 3.2: Ilustrando a vizinhança ***1-Opt***.

A segunda estrutura de vizinhança é a ***2-Opt***.

- ***2-Opt*** - N^2 : 2 arestas não adjacentes são removidas e outras 2 arestas são criadas para gerar uma nova árvore solução. O movimento no qual as arestas (3, 5) e (7, 10) são removidas da solução e as arestas (3, 4) e (7, 9) são criadas é ilustrado na Figura 3.3.

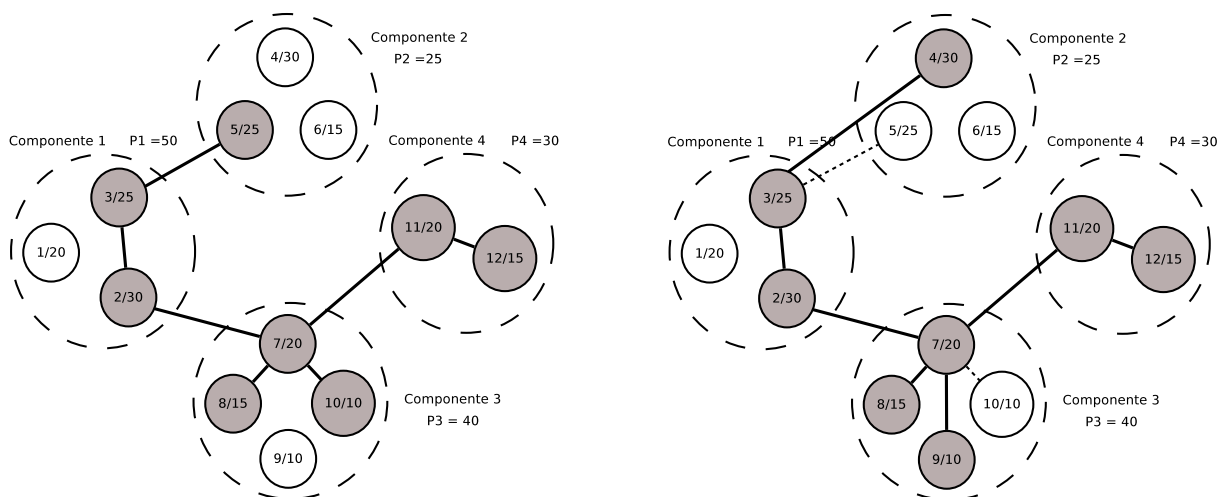


Figura 3.3: Ilustrando a vizinhança ***2-Opt***.

A vizinhança N^t é a ***t-Opt***.

- ***t-Opt*** - N^t : t arestas não adjacentes são removidas e outras t arestas são criadas para gerar uma nova árvore solução. O valor de t foi definido empiricamente e é tal que:

$$t = \min\{|V_i|, V_i \subseteq V, i \in m\}. \quad (3.23)$$

3.2.3.1 VND

O procedimento *Variable Neighborhood Descent* (VND) [29] envolve uma exploração exaustiva do espaço de soluções através de trocas sistemáticas de estruturas de vizinhança. Durante a busca local, apenas a solução que é melhor do que a solução atual é aceita. Quando uma solução melhor é encontrada, o método reinicia a busca, começando com a primeira estrutura de vizinhança.

O método VND baseia-se em três princípios: (i) um ótimo local para uma dada estrutura de vizinhança não corresponde necessariamente a um ótimo local de uma outra estrutura de vizinhança; (ii) um ótimo global corresponde a um ótimo local para todas as estruturas de vizinhança; e (iii) para muitos problemas, o ótimo local de uma dada estrutura de vizinhança está próximo do ótimo local de outras estruturas de vizinhança.

Este último princípio, de natureza empírica, indica que um ótimo local frequentemente fornece algum tipo de informação sobre o ótimo global. Este é o caso em que ótimos local e global compartilham uma série de variáveis com o mesmo valor.

A versão clássica do VND, utilizada neste trabalho, busca soluções ótimas locais seguindo uma ordem fixa de estruturas de vizinhança. Esta estratégia é amplamente aplicada e os resultados da literatura confirmam sua eficiência [19, 18].

O pseudódigo do VND é apresentado no Algoritmo 3.2. O algoritmo requer que o número de vizinhanças a serem selecionadas, *max_buscas_locais*, seja maior do que zero e inicia tomando s_0 como melhor solução encontrada até o momento, sendo s_0 a solução retornada pelo algoritmo construtivo. O valor de *max_buscas_locais* é definido igualmente a t em (3.23). Em seguida, define-se que a primeira estrutura de vizinhança, $N^t, t = 1$, será analisada. O laço nas linhas 3 – 11 é repetido até que todas as vizinhanças sejam analisadas sem sucesso. As vizinhanças são caracterizadas como *first improvement*. Na linha 4, uma busca local na vizinhança N^t é aplicada a s^* , gerando a solução s . Se s for melhor do que s^* , então a solução s é atribuída a s^* e o processo é reiniciado com a

vizinhança N^1 , como é mostrado nas linhas 6 e 7. Caso contrário, o processo continua com a vizinhança N^{t+1} . Ao final da execução, o algoritmo retorna a melhor solução encontrada s^* .

Algoritmo 3.2 VND

```

1:  $s^* \leftarrow s_0$ ;
2:  $t \leftarrow 1$ ;
3: enquanto  $t \leq \text{max\_buscas\_locais}$  faça
4:    $s \leftarrow \text{BuscaLocal}(s^*, N^t)$ ;
5:   se  $f(s) < f(s^*)$  então
6:      $s^* \leftarrow s$ ;
7:      $t \leftarrow 1$ ;
8:   senão
9:      $t \leftarrow t + 1$ ;
10:  fim se
11: fim enquanto

```

Heurísticas construtivas são, em geral, mais rápidas do que métodos de busca local. Contudo, a busca local tende a fornecer soluções de melhor qualidade. No entanto, quanto maior for a complexidade das soluções, maior será o risco do algoritmo cair em ótimos locais. Neste contexto, podem ser utilizadas abordagens que aproveitam elementos aleatórios para evitar a armadilha dos ótimos locais. Entre tais abordagens estão as metaheurísticas, definidas a seguir.

3.2.4 Metaheurísticas

Metaheurísticas são métodos de solução que coordenam procedimentos que empregam estratégias para escapar de mínimos locais em espaços de busca de soluções complexas. Assim, uma metaheurística visa produzir um resultado satisfatório para um problema, porém sem qualquer garantia de otimalidade. As metaheurísticas são estratégias de alto nível que, por meio de heurísticas subordinadas, tentam explorar o espaço de busca eficientemente, visando encontrar soluções de boa qualidade que estejam próximos do ótimo ou sejam ótimas. Além das funções desempenhadas durante o processo de busca, as heurísticas subordinadas servem também para modelar o conhecimento específico referente a cada problema [11, 45].

Diferentemente das heurísticas clássicas, as metaheurísticas incorporam mecanismos

para tentar evitar ficarem presas em ótimos locais de baixa qualidade. As metaheurísticas implementadas neste trabalho são GRASP e ILS.

3.2.4.1 GRASP

GRASP (*Greed Randomized Adaptive Search Procedure*) é uma metaheurística *multistart*, proposta por [5], em que cada iteração é composta de uma fase de construção e de uma fase aprimoramento (busca local). Durante o processo, a solução incubente é armazenada e atualizada sempre que a fase de aprimoramento resulta em uma solução melhor. Ao final de um número estabelecido de iterações, o algoritmo retorna a melhor solução encontrada [11].

A fase de construção gera uma solução viável para o problema através de um procedimento parcialmente guloso e parcialmente aleatório. A cada etapa da construção, a seleção dos elementos que vão compor a solução é feita aleatoriamente em um subconjunto dos melhores elementos candidatos, chamado Lista de Candidatos Restrita (LRC). A fase de aprimoramento é uma busca local.

As iterações GRASP são totalmente independentes, não havendo nenhum tipo de memória nem aprendizagem no decorrer da execução. Daí muitas vezes ele ser chamado de *multistart*. Consequentemente, nenhuma informação relevante obtida nas iterações anteriores são incorporadas no processo de busca das iterações remanescentes.

No Algoritmo 3.3 é apresentado o pseudocódigo do GRASP implementado neste trabalho. *GeraSolucaoInicial()* é responsável por gerar soluções iniciais e é baseado na heurística construtiva apresentada no Algoritmo 3.1. A fase de aprimoramento, *BuscaLocal()*, é realizada pelo VND, apresentado no Algoritmo 3.2. A função $c(s)$ retorna o custo da solução s e s^* é a solução incubente. O algoritmo requer que o número de iterações, *max_iteracoes*, seja maior do que zero; neste trabalho, definido empiricamente igual 10. Ao final da sua execução, o algoritmo retorna a melhor solução encontrada s^* .

Algoritmo 3.3 GRASP

```

1:  $s^* \leftarrow \emptyset$ ;
2:  $c(s^*) \leftarrow \infty$ ;
3: para  $i \in \{1, \dots, max\_iteracoes\}$  faça
4:    $s_0 \leftarrow GeraSolucaoInicial()$ ;
5:    $s \leftarrow BuscaLocal(s_0)$ ;
6:   se  $c(s) < c(s^*)$  então
7:      $s^* \leftarrow s$ ;
8:   fim se
9: fim para

```

3.2.4.2 ILS

O ILS (*Iterated Local Search*) [27] é uma metaheurística baseada na ideia de que um procedimento de busca local pode ser melhorado, gerando-se novas soluções de partida, as quais são obtidas por meio de perturbações na solução ótima local. A perturbação precisa ser suficientemente forte para permitir que a busca local explore diferentes soluções, mas também fraca o suficiente para evitar um reinício aleatório.

Segundo [27], para desenvolver um algoritmo ILS, quatro componentes devem ser especificadas, são elas:

GeraSoluçãoInicial: responsável pela construção de soluções iniciais. É desejável que este método seja rápido e que as soluções obtidas sejam bons pontos de partida para a busca local.

BuscaLocal: responsável por explorar o espaço de busca e encontrar bons ótimos locais.

Perturbação: responsável por gerar soluções modificadas de tal forma que, após a aplicação da busca local, o novo ótimo local seja diferente do usado como ponto de partida.

CritérioAceitação: avalia se o novo ótimo local será utilizado como nova solução de partida na iteração seguinte ou se será descartado.

Assim como os procedimentos de busca local clássicos, o ILS parte de uma solução inicial, na qual um procedimento de busca será aplicado até que um ótimo local seja obtido. Este ótimo local será a solução de partida do laço principal do ILS. A cada

iteração deste, a solução de partida é perturbada. Em seguida, a solução é refinada através do procedimento de busca e, por fim, este novo ótimo local é avaliado segundo um critério de aceitação que determinará se ele será utilizado como nova solução de partida na próxima iteração ou se será descartado. O Algoritmo 3.4 apresenta a arquitetura básica do ILS, implementado neste trabalho.

Algoritmo 3.4 ILS

```

1:  $s_0 \leftarrow \text{GeraSolucaoInicial}();$ 
2:  $s^* \leftarrow \text{BuscaLocal}(s_0);$ 
3: enquanto  $i \in \{1, \dots, \text{max\_iteracoes}\}$  faça
4:    $s_1 \leftarrow \text{Perturbacao}(s^*, \text{historico});$ 
5:    $s_1^* \leftarrow \text{BuscaLocal}(s_1);$ 
6:    $s^* \leftarrow \text{CriterioAceitacao}(s^*, s_1^*, \text{historico});$ 
7: fim enquanto
  
```

No algoritmo ILS desenvolvido, a solução inicial é gerada com base no Algoritmo 3.1 e a busca local do ILS é feita pelo VND, apresentado no Algoritmo 3.2. O valor de *max_iteracoes* foi definido empiricamente igual a 10. A perturbação é realizada removendo-se (i) 1/4 das arestas ou (ii) 1/4 dos vértices, gerando uma solução parcial. A escolha entre (i) e (ii) é feita aleatoriamente. Em seguida, a solução é reconstruída de forma aleatória até que tenha-se uma solução viável. Como critério de aceitação, foi definido que são aceitas soluções com distância máxima de 40% da melhor solução encontrada.

3.3 Método Exato

O método exato proposto foi implementado fazendo a união da metaheurística com a formulação matemática. Assim, o método utiliza o conjunto de vértices pertencentes a solução da metaheurística como parâmetro de entrada do algoritmo exato. Para isso, as metaheurísticas utilizadas foram as apresentadas na Seção 3.2.4, GRASP e ILS, gerando dois métodos exatos.

A formulação matemática usada para resolver o problema foi a segunda formulação apresentada na Seção 3.1, baseada no CP-PAGMG [15]. A escolha da formulação foi feita empiricamente, onde os resultados produzidos pelas duas formulações foram analisados e verificou-se que os resultados da segunda foram mais expressivos do que os obtidos com

a primeira, tanto em tempo computacional quanto em custo da solução encontrada. Tais resultados podem ser vistos no Apêndice B.

O método exato proposto é apresentado no Algoritmo 3.5. Inicialmente, uma metaheurística é executada (*ExecutaMetaheuristica()*) gerando um conjunto de vértices V' , $V' \subseteq V$. Dessa forma, para cada $k \in V'$, gera-se as restrições de subciclo (*GeraRestricaoSubciclo()*), com a árvore com raiz no vértice k , respeitando os seguintes critérios:

$$u_{ij} = w_{kij} + w_{kji} \quad \forall (i, j) \in E, \quad (3.24)$$

$$\sum_{j \in N(i)} w_{kij} \leq 1 \quad \forall i \in V - \{k\}, \quad (3.25)$$

$$\sum_{i \in N(k)} w_{kik} = 0. \quad (3.26)$$

Em seguida, gera-se o modelo (*GeraModelo()*) e resolve-se o problema (*Resolve()*), produzindo uma solução s composta por variáveis com valores inteiros. Assim, constroi-se um grafo G' a partir de s (*GeraGrafo()*) - utilizando as variáveis u_{ij} e $z_i \neq 0$ do modelo - e faz-se uma busca em largura em G' (*BuscaEmLargura()*), para verificar se não existe ciclo no grafo. Caso não exista ciclo no grafo, ou seja, caso G' seja uma árvore (*EhArvore()*), então a solução ótima foi encontrada e o algoritmo é encerrado retornando a solução ótima s^* . Caso seja encontrado algum ciclo em G' , insere-se um vértice i , pertencente ao ciclo, em V' e gera-se as restrições de subciclo para i . Em seguida, o método é reiniciado a partir da linha 5 do algoritmo (*VaiPara()*).

Algoritmo 3.5 MÉTODO EXATO

```

1:  $V' \subseteq V, V' \leftarrow ExecutaMetaheuristica();$ 
2: para  $k \in V'$  faça
3:    $GeraRestricaoSubciclo(k);$ 
4: fim para
5:  $GeraModelo();$ 
6:  $s \leftarrow Resolve(), s \in \mathbb{Z};$ 
7:  $G' \leftarrow GeraGrafo(s);$ 
8:  $BuscaEmLargura(G');$ 
9: se  $EhArvore(G')$  então
10:   $s^* \leftarrow s; \{solução ótima encontrada\}$ 
11: senão
12:   $GeraRestricaoSubciclo(i), i \in V;$ 
13:   $VaiPara(linha 5);$ 
14: fim se

```

3.4 Resumo

Neste capítulo foram apresentados os algoritmos implementados para o Problema da Árvore Geradora Mínima Generalizado com Prêmios nos Vértices proposto neste trabalho. Foram propostas duas formulações matemáticas e, como propostas heurísticas, foram implementados um algoritmo construtivo baseado no Algoritmo de Prim e procedimentos de busca local selecionados a partir do VND. A partir desses algoritmos, foram desenvolvidas duas metaheurísticas, GRASP e ILS. Ambas as metaheurísticas foram utilizadas no desenvolvimento de métodos exatos. A formulação baseada no CP-PAGMG foi utilizada para compor o método exato. No próximo capítulo serão apresentados e analisados os resultados obtidos com cada algoritmo e a comparação entre eles.

Capítulo 4

Resultados Computacionais

Neste capítulo são apresentados os resultados obtidos com os algoritmos propostos. Primeiramente serão descritas as ferramentas utilizadas e as características das instâncias criadas; em seguida, são apresentados os resultados alcançados nos testes computacionais; e, por fim, é realizada uma comparação entre as metaheurísticas implementadas.

4.1 Ferramentas

A ferramenta utilizada para a execução do método exato foi o CPLEX [21]. O CPLEX é um dos pacotes de software de otimização linear mista mais utilizados na literatura. Ele é responsável por gerenciar todo o processo e, se necessário, adicionar novos cortes. Neste trabalho, todo o gerenciamento da execução do método é feito por esta ferramenta, como a realização do *Branch-and-Bound* [26], a escolha de variáveis para o *branch* e a adição de novos cortes. Inicialmente, os cortes adicionados são as próprias restrições do modelo, pois o algoritmo começa com o modelo vazio.

Os testes computacionais foram executados em um computador Intel Core i7 – 2600 com 3.40 GHz e 32 Gb de memória RAM, rodando um sistema operacional Arch Linux 3.9.2 – 1, sendo executado em apenas uma *thread*. Todos os algoritmos foram implementados na linguagem de programação C++ e compilados com g++ versão 4.8.0. O método exato utilizou, ainda, a API ILOG CPLEX 12.5.0.0, onde o pré-processamento, a heurística e a geração de cortes automáticos do CPLEX foram desabilitados.

4.2 Instâncias

Como ainda não se tem conhecimento de trabalhos para o PAGMG-P, um conjunto de instâncias foi criado. No entanto, existem instâncias para o problema geral, o PAGMG. As instâncias para o PAGMG disponíveis na literatura e utilizadas por [11, 13] possuem grafos completos e simétricos.

Considerando essas informações prévias, foi implementado um gerador de instâncias para o PAGMG-P. O gerador de instâncias segue, resumidamente, os seguintes passos: (i) particionar aleatoriamente o número de vértices de cada componente de maneira uniforme; (ii) definir o prêmio de cada vértice; (iii) definir o prêmio mínimo de cada componente; (iv) criar um grafo completo intracomponente e definir o custo de cada aresta; e (v) criar um grafo completo intercomponente e definir o custo de cada aresta. O número de vértices e o número de componentes são dados na entrada. O prêmio de cada vértice, o prêmio mínimo de cada componente e o custo de cada aresta são calculados aleatoriamente em um dado intervalo.

Foram geradas 221 instâncias com $10 \leq |V| \leq 1000$ e $2 \leq m \leq 30$. Seguindo a literatura do PAGMG, todas as instâncias geradas possuem grafos completos e simétricos.

4.3 Resultados Experimentais

Nesta seção, estão apresentados os resultados computacionais obtidos por meio de testes realizados com os algoritmos apresentados no Capítulo 3. Os testes foram realizados sobre 221 instâncias, cujas dimensões são apresentadas no Apêndice A.

Os resultados obtidos com os algoritmos apresentados na seção anterior, serão apresentados com a seguinte nomenclatura:

- EX1** : Formulação matemática baseada no L-PAGMG
- EX2** : Formulação matemática baseada no CP-PAGMG
- GRA** : Metaheurística GRASP
- ILS** : Metaheurística ILS
- E2G** : MÉTODO EXATO utilizando a combinação EX2 + GRA
- E2I** : MÉTODO EXATO utilizando a combinação EX2 + ILS

O critério de parada estipulado para os algoritmos exatos foi encontrar a solução ótima ou chegar ao tempo limite de execução, o que ocorresse primeiro. Se a execução chegasse ao tempo limite, a melhor solução viável encontrada até o momento era retornada. Para os algoritmos combinados, o tempo computado foi o tempo total dos algoritmos. Para as

metaheurísticas, o tempo limite considerado foi de 1800 segundos (30 minutos), enquanto para os algoritmos exatos o tempo limite foi de 7200 segundos (2 horas). Os resultados apresentados pelas heurísticas são as médias de 10 execuções. Dentre as instâncias utilizadas, foi encontrada a solução ótima de 149 instâncias, enquanto para as outras 72 foram encontradas soluções viáveis.

O gap de uma solução s em cada algoritmo é dado pela função (4.1).

$$gap(s) = (s - s^*) \div s^*, \quad (4.1)$$

onde s é a solução encontrada pelo algoritmo e s^* é a melhor solução encontrada entre todos os algoritmos. Multiplicando o resultado da função (4.1) por 100, temos o resultado em porcentagem.

Os resultados estão descritos nas Tabelas B.1, B.2, B.3, B.4, B.5 e B.6, apresentadas no Apêndice B. A primeira coluna das tabelas apresenta a denominação da instância. Nas três colunas seguintes têm-se o custo da solução encontrada, tempo de execução (em segundos) e gap médio (em porcentagem) do algoritmo EX1. Essa organização se repete nas colunas seguintes para os outros algoritmos. Nas tabelas, as soluções ótimas diferem das soluções viáveis por estarem em negrito. E as soluções com valor “-” significam que o algoritmo não conseguiu encontrar uma solução viável dentro do tempo limite considerado.

Na Tabela 4.1, são apresentados os resultados quantitativos dos algoritmos. A primeira coluna apresenta a nomenclatura do algoritmo. Na coluna seguinte tem-se a quantidade de soluções ótimas encontradas e, na próxima coluna, tem-se o número de soluções viáveis encontradas, quando o ótimo não é encontrado. E na última coluna tem-se a quantidade de instâncias em que não foi encontrada solução viável.

Analisando as tabelas, apenas os algoritmos EX1 e EX2 não conseguiram encontrar solução viável para algumas instâncias, o que já era esperado, visto que tais algoritmos são modelados apenas com as formulações matemáticas propostas. Os melhores resultados foram gerados com o algoritmo E2I, onde foram encontradas 149 soluções ótimas, além de 72 soluções viáveis. Tais resultados ilustram a eficiência do método proposto. O motivo do algoritmo E2I ter encontrado as melhores soluções foi que o mesmo utiliza as soluções do ILS como entrada, e essa metaheurística atingiu, geralmente, melhores soluções do que a GRASP.

Para ilustrar o quanto cada algoritmo se aproximou da melhor solução encontrada em todas as instâncias, foi feita a média do gap de todas as soluções encontradas por cada

Algoritmo	Ótima	Viável	Não encontrada
EX1	91	79	51
EX2	106	82	33
GRA	37	184	-
ILS	45	176	-
E2G	128	83	-
E2I	149	72	-

Tabela 4.1: Resultados quantitativos dos algoritmos apresentados no Capítulo 4.

algoritmo. O gap médio é apresentado na Tabela 4.2, onde observa-se que o algoritmo E2I apresentou gap médio igual a 0, pois o mesmo obteve os melhores resultados para todas as instâncias. Dos algoritmos metaheurísticos, GRA e ILS, o ILS mostrou-se mais eficiente, gerando um valor menor de gap médio.

Algoritmo	Gap Médio (%)
EX1	1740.07
EX2	1169.92
GRA	1296.74
ILS	1057.11
E2G	301.58
E2I	0.00

Tabela 4.2: Gap médio considerando a melhor solução encontrada para todas as instâncias.

Com o intuito de avaliar as soluções geradas por cada algoritmo em relação as instâncias que tiveram solução ótima conhecida, foi feita a média do gap das soluções encontradas por cada algoritmo. Na Tabela 4.3 é apresentado o gap médio considerando apenas as instâncias onde foram encontradas soluções ótimas. Observa-se que o gap médio dos algoritmos EX2 e ILS se aproximaram numericamente. No entanto, a média de tempo de execução para chegar às soluções foi bastante diferente. Enquanto o EX2 levou, aproximadamente, 3004.57 segundos para atingir a solução, o ILS levou 1236.21 segundos em média, como ilustrado na Tabela 4.4.

Na Tabela 4.5 são ilustradas as médias dos tempos computacionais dos algoritmos considerando todas as instâncias. Observa-se que o tempo de execução dos algoritmos exatos são maiores quando comparados aos algoritmos heurísticos. Essa característica era esperada, devido a grande diferença entre os limites de tempo de execução desses algoritmos.

Como visto nas Tabelas, o algoritmo que gerou melhores resultados foi o E2I. O algoritmo que mais se aproximou de seus resultados foi o E2G. As modelagens desses

Algoritmo	Gap Médio (%)
EX1	1854.55
EX2	1310.26
GRA	1616.15
ILS	1300.41
E2G	317.39
E2I	0.00

Tabela 4.3: Gap médio considerando as instâncias onde foram encontradas soluções ótimas.

Algoritmo	Tempo Médio (segundos)
EX1	3865.54
EX2	3004.57
GRA	1319.45
ILS	1236.21
E2G	2845.08
E2I	2407.96

Tabela 4.4: Média dos tempos computacionais considerando as instâncias onde foram encontradas soluções ótimas.

Algoritmo	Tempo Médio (segundos)
EX1	4366.00
EX2	3924.30
GRA	1499.13
ILS	1441.72
E2G	4305.47
E2I	4004.62

Tabela 4.5: Média dos tempos computacionais considerando todas as instâncias.

algoritmos se diferem pela metaheurística utilizada, enquanto E2G utiliza GRASP, E2I utiliza ILS. Para melhor avaliar as soluções geradas pelas metaheurísticas, será realizada uma comparação entre elas.

4.3.1 GRASP vs ILS

Nesta subseção é apresentada uma comparação das metaheurísticas GRASP e ILS. Para isso, a ferramenta `tttplots` é utilizada. O `tttplots` [2] fornece a probabilidade acumulada de um algoritmo atingir uma determinada solução alvo em relação ao tempo. O propósito é verificar o tempo necessário para que os algoritmos encontrem soluções de boa qualidade.

Diante do fato de que o funcionamento do GRASP não se baseia em aprendizagem sobre a sua execução enquanto o ILS utiliza informações de execuções anteriores, o objetivo foi analisar a influência da utilização dessas informações.

Foram consideradas 4 instâncias para análise dos algoritmos. Os critérios para escolha das instâncias foram: (i) o número de vértices da instância deve ser no mínimo 50 e (ii) o tempo de execução da instância presente na tabela de resultados deve estar entre os maiores. Sendo assim, as seguintes instâncias foram escolhidas: 50al10, 75al5, 150al15 e 300al30. As dimensões das instâncias podem ser vistas no Apêndice A e os resultados no Apêndice B. Para cada instância, cada algoritmo foi executado 200 vezes, valor aconselhado pelos autores em [2].

A partir do `tttplots`, foram gerados 4 gráficos, cada um representando uma das instâncias consideradas. Nos gráficos, o eixo das ordenadas representa a probabilidade acumulada do algoritmo atingir a solução alvo e o eixo das abscissas representa o tempo para atingir uma solução. Para esta análise, a solução alvo é a melhor solução encontrada entre os algoritmos considerados. O critério de parada foi de encontrar a solução alvo ou chegar ao tempo limite de 1800 segundos.

A análise dos gráficos pode ser feita verificando o alinhamento das curvas: a curva mais a esquerda indica o algoritmo que converge mais rápido para o valor alvo. Pode-se perceber que o ILS encontra as soluções alvo mais rapidamente do que o GRASP. Os Gráficos 4.3.1 e 4.3.1 apresentam as curvas das metaheurísticas semelhantes em alguns pontos, apesar de possuírem valores distintos, mostrando que, para tais instâncias, os algoritmos se comportaram de forma um tanto similar. No entanto, a medida que o tamanho das instâncias crescem (aumentando o número de vértices), essa semelhança diminui, como visto nas Figuras 4.3.1 e 4.3.1.

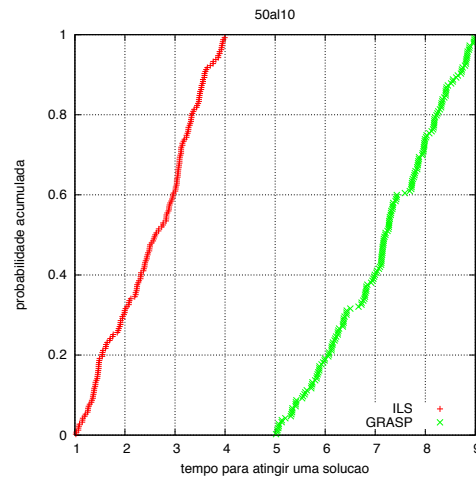


Figura 4.1: Função de distribuição de probabilidade dos algoritmos atingirem uma solução alvo com a Instância 50a110.

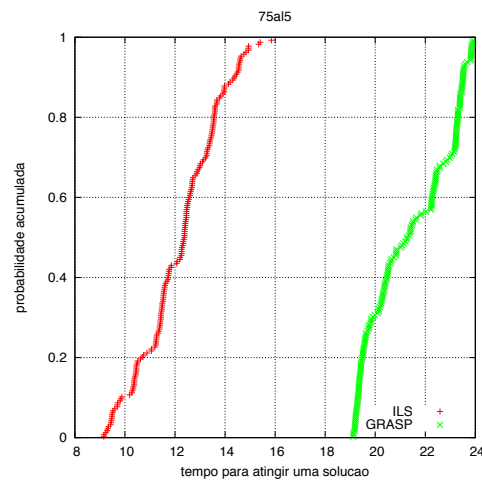


Figura 4.2: Função de distribuição de probabilidade dos algoritmos atingirem uma solução alvo com a Instância 75a15.

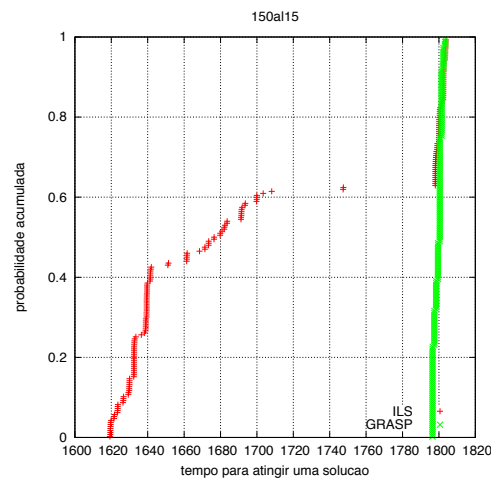


Figura 4.3: Função de distribuição de probabilidade dos algoritmos atingirem uma solução alvo com a Instância 150a115.

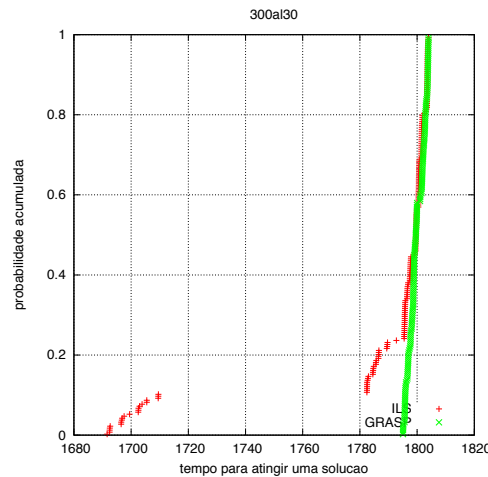


Figura 4.4: Função de distribuição de probabilidade dos algoritmos atingirem uma solução alvo com a Instância 300al30.

Em relação ao tempo computacional especificamente, na maioria dos casos o GRASP apresenta tempos computacionais maiores do que o ILS para encontrar a solução alvo. Esse resultado era o esperado, pois na maioria das instâncias consideradas o GRASP já havia apresentado tempos de execução maiores do que o ILS e, além disso, para todas as instâncias havia gerado uma solução igual ou pior do que a solução alvo.

Mesmo com as diferenças tanto em custo da solução quanto em tempo computacional apresentadas pelas metaheurísticas, com o ILS gerando as melhores soluções, para cada instância analisada os algoritmos variaram bastante, caracterizando uma falta de robustez de ambos.

4.4 Resumo

Este capítulo apresentou os resultados computacionais dos testes realizados com os algoritmos propostos neste trabalho. Primeiramente descreveu-se as ferramentas e as instâncias utilizadas na execução dos testes computacionais. Em seguida foram apresentados os resultados dos algoritmos propostos. Também foram apresentados e discutidos resultados de testes comparativos entre os algoritmos GRASP e ILS. No próximo capítulo serão feitas as considerações finais deste trabalho e possíveis linhas de trabalhos futuros serão apontadas.

Capítulo 5

Conclusões e Trabalhos Futuros

Este trabalho propôs uma nova variante para o Problema da Árvore Geradora Mínima Generalizado (PAGMG) com Prêmios nos Vértices, o PAGMG-P. O PAGMG é amplamente estudado na literatura, mas, apesar de sua aplicação ser abrangente, esta abordagem as vezes não contempla casos particulares de alguns problemas da realidade. A variante proposta foi motivada pelas possíveis aplicações reais existentes.

Foram propostas duas formulações matemáticas para a nova variante e, além disso, foi implementado um gerador de instâncias, as quais foram utilizadas neste trabalho. Foram implementadas duas metaheurísticas, GRASP e ILS, que forneceram bons resultados.

O ILS conseguiu alcançar melhores resultados do que o GRASP para a maioria das instâncias, tanto em custo da solução quanto em tempo computacional. Além disso, esta metaheurística encontrou soluções de boa qualidade mais rapidamente do que o GRASP. No entanto, a análise mais aprofundada desses algoritmos sugeriu uma falta de robustez de ambos.

Ambas as metaheurísticas também foram utilizadas como entrada para o método exato proposto. Os melhores resultados obtidos neste trabalho se deram pela união da metaheurística ILS com o método exato. Os resultados quantitativos expostos ilustraram a eficiência desse método proposto.

Como trabalhos futuros, propõe-se o estudo de heurísticas que utilizem técnicas de memória adaptativa, para a implementação de algoritmos que possam ser configurados no decorrer da execução, propiciando calibrar o algoritmo às características de cada instância. Os autores em [12] implementaram um GRASP com memória adaptativa para solução do PAGMG e obtiveram bons resultados.

Neste trabalho foi implementada uma adaptação do algoritmo de Prim como algoritmo

construtivo. Assim, como trabalhos futuros, propõe-se a implementação da adaptação dos algoritmos de Kruskal e Solin e comparação de seu desempenho.

Propõe-se também a implementação de algoritmos híbridos, utilizando, por exemplo, um método exato como uma das buscas locais do VND, como feito em [18] para o PAGMG. Além disso, um pré-processamento nas instâncias poderia ser aplicado, para que o número de arestas da instância pudesse ser reduzido, como feito por [11] para o PAGMG. Propõe-se também a adaptação das instâncias do PAGMG existentes na literatura para a variante abordada.

Referências

- [1] AHUJA, R. K., MAGNANTI, T. L. *Network Flows: Theory, Algorithms and Applications*. Prentice-Hall, New Jersey, 1993.
- [2] AIEX, R. M., RESENDE, M. G. C., RIBEIRO, C. C. Tttplots: A perl program to create time-to-target plots. *Optimization Letters* 1 (2006), 10–1007.
- [3] CORUS, D., LEHRE, P. K., NEUMANN, F. The generalized minimum spanning tree problem: A parameterized complexity analysis of bi-level optimisation. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2013), GECCO '13, ACM, p. 519–526.
- [4] DROR, M., HAOUARI, M., CHAOUACHI, J. Generalized spanning trees. *European Journal of Operational Research* 120 (2000), 583–592.
- [5] FEO, T., RESENDE, M., SMITH, S. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research* 42 (1994), 860–878.
- [6] FEREMANS, C. Generalized spanning trees and extensions. Dissertação de Mestrado, Université Libre de Bruxelles, 2001.
- [7] FEREMANS, C., LABBÉ, M., LAPORTE, G. The generalized minimum spanning tree problem: Polyhedral analysis and branch-and-cut algorithm. Technical report, Université Libre de Bruxelles, Bruxelles, Bélgica, 2000.
- [8] FEREMANS, C., LABBÉ, M., LAPORTE, G. On generalized minimum spanning trees. *European Journal of Operational Research* 134 (2001), 48–50.
- [9] FEREMANS, C., LABBÉ, M., LAPORTE, G. A comparative analysis of several formulations for the generalized minimum spanning tree problem. *Networks* 39, 1 (2002), 29–34.
- [10] FEREMANS, C., LODI, A., TOTH, P., TRAMOTANI, A. Improving on branch-and-cut algorithms for generalized minimum spanning trees. *Pacific Journal of Optimization* 1 (2005), 491–508.
- [11] FERREIRA, C. M. S. Algoritmos para o problema da árvore geradora mínima generalizado. Dissertação de Mestrado, Universidade Federal Fluminense, 2007.
- [12] FERREIRA, C. M. S., OCHI, L. S. Metaheurística grasp com memória adaptativa para a solução do problema da árvore geradora mínima generalizado. *Anais do XXVII Congresso da SBC* 27 (2007), 1202–1211.

- [13] FERREIRA, C. S., OCHI, L. S., PARADA, V., UCHOA, E. A grasp-based approach to the generalized minimum spanning tree problem. *Expert Systems with Applications* 39 (February 2012), 3526–3536.
- [14] GAREY, M. R., JOHNSON, D. S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.
- [15] GOLDEN, B., RAGHA VAN, S., STANOJE VIC, D. The prize-collecting generalized minimum spanning tree problem. *Journal of Heuristics* 14 (2008), 69–93.
- [16] GOLDEN, B., RAGHAVAN, S., STANOJEVIĆ, D. Heuristic search for the generalized minimum spanning tree problem. *INFORMS J. on Computing* 17, 3 (julho de 2005), 290–304.
- [17] HAOUARI, M., SIALA, J. C. Upper and lower bounding strategies for the generalized minimum spanning tree problem. *European Journal of Operational Research* 171, 2 (2006), 632–647.
- [18] HU, B., LEITNER, M., RAIDL, G. Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics* 14, 5 (2008), 473–499.
- [19] HU, B., LEITNER, M., RAIDL, G. R. Computing generalized minimum spanning trees with variable neighborhood search. In *18th Mini Euro Conference on VNS* (2005).
- [20] HU, B., RAIDL, G. An evolutionary algorithm with solution archive for the generalized minimum spanning tree problem. In *Computer Aided Systems Theory - EUROCAST 2011* (2012), R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., vol. 6927 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, p. 287–294.
- [21] IBM, I. B. M. C. Ibm ilog cplex optimization studio, 2013. Página visitada em 12-05-2013.
- [22] IHLER, E., REICH, G., WIDMAYER, P. Class steiner trees and vlsi-design. *Discrete Applied Mathematics* 90 (1999), 173–194.
- [23] JIANG, H., CHEN, Y. An efficient algorithm for generalized minimum spanning tree problem. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation* (New York, NY, USA, 2010), GECCO '10, ACM, p. 217–224.
- [24] KANSAL, A. R., TORQUATO, S. Globally and locally minimal weight spanning tree networks. *Physica A* 301 (2001), 601–619.
- [25] KRUSKAL, J. B. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society* 7 (1956), 48–50.
- [26] LAND, A. H., DOIG, A. G. An automatic method of solving discrete programming problems. *Econometrica* 28 (1960), 497–520.

- [27] LOURENÇO, H. R., MARTIN, O. C., STUTZLE, T. Iterated local search. In *Handbook of Metaheuristics* (2003), S. US, Ed., p. 320–353.
- [28] MAGNANTI, T. L., WOLSEY, L. A. Optimal trees. In *Handbooks in Operations Research and Management Science* (1995), C. L. Ball, C. L. Momma, and G. L. Nemhauser, Eds., vol. 7, Elsevier Science, p. 503–615.
- [29] MLADENOVIC, N., HANSEN, P. A variable neighborhood search. *Computers and Operations Research* 24 (1997), 1097–1100.
- [30] MYUNG, Y. S., LEE, C. H., TCHA, D. W. On the generalized minimum spanning tree problem. In *Networks* 26 (1995), p. 231–241.
- [31] ONCAN, T., CORDEAU, J. F., LAPORTE, G. A tabu search heuristic for the generalized minimum spanning tree problem. Relatório Técnico, Center for Research on Transportation, HEC Montréal, Montréal, Canada, 2006.
- [32] POP, P. On the prize-collecting generalized minimum spanning tree problem. *Annals of Operations Research* 150, 1 (2007), 193–204.
- [33] POP, P., MATEI, O., SABO, C. A memetic algorithm for solving the generalized minimum spanning tree problem. In *Soft Computing in Industrial Applications*, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, and L. Costa, Eds., vol. 96 of *Advances in Intelligent and Soft Computing*. Springer Berlin Heidelberg, 2011, p. 187–194.
- [34] POP, P., SITAR, C. P., POP, N., ZELINA, I. A new approach to generalized network design problems. In *Proceedings of the 6th WSEAS International Conference on Electronics, Hardware, Wireless and Optical Communications* (Stevens Point, Wisconsin, USA, 2007), EHAC’07, World Scientific and Engineering Academy and Society (WSEAS), p. 1–5.
- [35] POP, P. C. *The Generalized Minimum Spanning Tree Problem*. Tese de Doutorado, University of Twente, Netherlands, 2002.
- [36] POP, P. C. New models of the generalized minimum spanning tree problem. *Journal of Mathematical Modelling and Algorithms* 3, 2 (2004), 153–166.
- [37] POP, P. C., STILL, G., KERN, W. An approximation algorithm for the generalized minimum spanning tree problem with bounded cluster size. In *Algorithms and Complexity in Durham Proceedings of the first ACiD Workshop* (2005), H. Broersma, M. Johnson, and S. Szeider, Eds., vol. 4, Algorithms and Complexity in Durham, Kings College Publications, p. 115–121.
- [38] PRIM, R. C. Shortest connection networks and some generalizations. *Bell System Technical Journal* 36 (1957), 1389–1401.
- [39] RAGHAVAN, S. On modeling the generalized minimum spanning tree. Relatório Técnico, The Robert H. Smith School of Business, University of Maryland, College Park, MD, 2002.
- [40] SALAZAR, J. J. A note on the generalized steiner tree polytope. *Discrete Applied Mathematics* 100 (1991), 137–144.

- [41] SANTOS, N. M., OCHI, L. S., SEMAAN, G. S. Metaheuristic grasp with path-relinking to the solution of the graph partitioning problem with capacity and connectivity constraints. *Lecture Notes in Computer Science (LNCS) 7435* (2012), 630–641.
- [42] SEMAAN, G. S., BRITO, J. A. M., OCHI, L. S. Efficient algorithms for the capacity and connectivity graph partition problem. *Mecánica Computacional 29* (2010), 9499–9509.
- [43] SHYU, S. J., YIN, P., LIN, B. M. T., HAOUARI, M. Upper and lower bounding strategies for the generalized minimum spanning tree problem. *European Journal of Operational Research 171* (2006), 632–647.
- [44] SHYU, S. J., YIN, P. Y., LIN, B. M. T., HAOUARI, M. Ant-tree: an ant colony optimization approach to the generalized minimum spanning tree problem. *Journal of Experimental and Theoretical Artificial Intelligence 15* (2003), 103–112.
- [45] SILVA, M. M. Uma heurística baseada em iterated local search para o problema de roteamento de veículos com entregas fracionárias. Dissertação de Mestrado, Universidade Federal Fluminense, 2013.
- [46] SILVER, E. A., VICTOR, R., VIDAL, V., DE WERRA, D. A tutorial on heuristic methods. *European Journal of Operational Research 5 3* (1980), 153–162.
- [47] ZANAKIS, S. H., EVANS, J. R., VAZACOUPOLOS, A. A. Heuristic methods and applications: a categorized survey. *European Journal of Operational Research 43 1* (1989), 88–110.

APÊNDICE A – Instâncias

Instância	V	E	m	Instância	V	E	m	Instância	V	E	m
10a13	10	45	3	60a115	60	1770	15	75a13	75	2775	3
15a14	15	105	4	60a120	60	1770	20	75a15	75	2775	5
20a15	20	190	5	64a14	64	2016	4	75a110	75	2775	10
25a16	25	300	6	64a15	64	2016	5	75a115	75	2775	15
30a15	30	435	5	64a18	64	2016	8	75a120	75	2775	20
30a110	30	435	10	64a110	64	2016	10	75a125	75	2775	25
35a17	35	595	7	65a15	65	2080	5	75a130	75	2775	30
40a14	40	780	4	65a110	65	2080	10	78a14	78	3003	4
40a18	40	780	8	65a115	65	2080	15	78a15	78	3003	5
40a110	40	780	10	65a118	65	2080	18	78a18	78	3003	8
45a16	45	990	6	65a125	65	2080	25	78a110	78	3003	10
45a19	45	990	9	67a14	67	2211	4	78a115	78	3003	15
45a110	45	990	10	67a110	67	2211	10	78a120	78	3003	20
45a115	45	990	15	67a120	67	2211	20	78a130	78	3003	30
50a12	50	1225	2	69a13	69	2346	3	80a12	80	3160	2
50a13	50	1225	3	69a19	69	2346	9	80a14	80	3160	4
50a15	50	1225	4	69a115	69	2346	15	80a15	80	3160	5
50a110	50	1225	10	70a14	70	2415	4	80a18	80	3160	8
50a115	50	1225	15	70a15	70	2415	5	80a110	80	3160	10
50a125	50	1225	25	70a17	70	2415	7	80a115	80	3160	15
52a18	52	1326	8	70a110	70	2415	10	80a118	80	3160	18
55a15	55	1485	5	70a115	70	2415	15	80a120	80	3160	20
55a19	55	1485	9	70a118	70	2415	18	80a122	80	3160	22
55a112	55	1485	12	70a120	70	2415	20	80a130	80	3160	30
58a14	58	1653	4	70a125	70	2415	25	83a14	83	3403	4
58a18	58	1653	8	72a13	72	2556	3	83a15	83	3403	5
58a110	58	1653	10	72a15	72	2556	5	83a18	83	3403	8
58a115	58	1653	15	72a18	72	2556	8	83a110	83	3403	10
60a12	60	1770	2	72a110	72	2556	10	83a115	83	3403	15
60a14	60	1770	4	72a115	72	2556	15	83a118	83	3403	18
60a15	60	1770	5	72a118	72	2556	18	83a124	83	3403	24
60a16	60	1770	6	72a120	72	2556	20	83a130	83	3403	30
60a110	60	1770	10	72a125	72	2556	25	85a14	85	3570	4

Tabela A.1: Instâncias geradas utilizadas neste trabalho: $|V|$ representa o número de vértices, $|E|$ representa o número de arestas e m representa o número de componentes.

Instância	V	E	m	Instância	V	E	m	Instância	V	E	m
85a15	85	3570	5	110a10	110	5995	10	160a25	160	12720	25
85a18	85	3570	8	110a15	110	5995	15	160a30	160	12720	30
85a110	85	3570	10	110a20	110	5995	20	170a5	170	14365	5
85a115	85	3570	15	110a25	110	5995	25	170a10	170	14365	10
85a118	85	3570	18	110a30	110	5995	30	170a15	170	14365	15
85a22	85	3570	22	120a5	120	7140	5	170a20	170	14365	20
85a25	85	3570	25	120a10	120	7140	10	170a25	170	14365	25
85a30	85	3570	30	120a15	120	7140	15	170a30	170	14365	30
90a3	90	4005	3	120a20	120	7140	20	180a5	180	16110	5
90a4	90	4005	4	120a25	120	7140	25	180a10	180	16110	10
90a5	90	4005	5	120a30	120	7140	30	180a15	180	16110	15
90a8	90	4005	8	130a5	130	8385	5	180a20	180	16110	20
90a10	90	4005	10	130a10	130	8385	10	180a24	180	16110	24
90a15	90	4005	15	130a15	130	8385	15	180a30	180	16110	30
90a20	90	4005	20	130a20	130	8385	20	190a5	190	17955	5
90a25	90	4005	25	130a24	130	8385	24	190a10	190	17955	10
90a30	90	4005	30	130a28	130	8385	28	190a15	190	17955	15
95a3	95	4465	3	140a5	140	9730	5	190a20	190	17955	20
95a5	95	4465	5	140a10	140	9730	10	190a25	190	17955	25
95a8	95	4465	8	140a15	140	9730	15	190a30	190	17955	30
95a10	95	4465	10	140a20	140	9730	20	200a5	200	19900	5
95a15	95	4465	15	140a25	140	9730	25	200a10	200	19900	10
95a20	95	4465	20	140a30	140	9730	30	200a15	200	19900	15
95a25	95	4465	25	150a5	150	11175	5	200a20	200	19900	20
95a28	95	4465	28	150a10	150	11175	10	200a25	200	19900	25
100a4	100	4950	4	150a15	150	11175	15	200a30	200	19900	30
100a5	100	4950	5	150a20	150	11175	20	210a5	210	21945	5
100a10	100	4950	10	150a25	150	11175	25	210a10	210	21945	10
100a15	100	4950	15	150a30	150	11175	30	210a15	210	21945	15
100a20	100	4950	20	160a5	160	12720	5	210a20	210	21945	20
100a25	100	4950	25	160a10	160	12720	10	210a25	210	21945	25
100a28	100	4950	28	160a15	160	12720	15	210a30	210	21945	30
110a5	110	5995	5	160a20	160	12720	20	220a5	210	21945	5

Tabela A.2: Instâncias geradas utilizadas neste trabalho.

Instância	V	E	m
220a10	220	24090	10
220a15	220	24090	15
220a20	220	24090	20
220a25	220	24090	25
220a30	220	24090	30
300a5	300	44850	5
300a10	300	44850	10
300a15	300	44850	15
300a20	300	44850	20
300a25	300	44850	25
300a30	300	44850	30
400a5	400	79800	5
400a10	400	79800	10
400a15	400	79800	15
400a20	400	79800	20
400a25	400	79800	25
400a30	400	79800	30
1000a5	1000	499500	5
1000a10	1000	499500	10
1000a15	1000	499500	15
1000a20	1000	499500	20
1000a25	1000	499500	25
1000a30	1000	499500	30

Tabela A.3: Instâncias geradas utilizadas neste trabalho.

APÊNDICE B - Resultados

Instância	EX1			EX2			GRA			ILS			E2G			E2I		
	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP
10a13	141	0.03	0	141	0.04	0	141	0.02	0	141	0.02	0	141	0.02	0	141	0.02	0
15a14	239	0.04	0	239	0.12	0	239	0.06	0	239	0.06	0	239	0.06	0	239	0.07	0
20a15	172	0.04	0	172	0.43	0	172	0.22	0	172	0.23	0	172	0.22	0	172	0.23	0
25a16	186	1.98	0	186	1.62	0	186	0.81	0	186	0.86	0	186	0.83	0	186	0.98	0
30a15	60	1.23	0	60	1.29	0	60	0.65	0	60	0.69	0	60	0.66	0	60	0.7	0
30a110	179	1.95	0	179	1.73	0	179	0.46	0	179	0.92	0	179	0.48	0	179	0.94	0
35a17	135	5.64	0	135	3.29	0	135	0.73	0	135	1.32	0	135	0.76	0	135	1.62	0
40a14	77	25.8	0	77	14.25	0	77	1.44	0	77	1.23	0	77	2.7	0	77	3.31	0
40a18	165	29.32	0	165	23.77	0	165	2.42	0	165	1.57	0	165	2.89	0	165	1.61	0
40a110	126	8.57	0	126	5.86	0	126	1.36	0	126	2.82	0	126	1.41	0	126	2.88	0
45a16	93	17.18	0	93	13.52	0	93	1.01	0	93	1.19	0	93	2.15	0	93	2.31	0
45a19	134	42.57	0	134	37.25	0	134	2.81	0	134	4.12	0	134	2.51	0	134	3.48	0
45a110	131	101.99	0	131	92.87	0	131	16.78	0	131	9.56	0	131	13.96	0	131	11.01	0
45a115	176	16.53	0	176	11.32	0	176	2.76	0	176	0.62	0	176	2.35	0	176	0.54	0
50a12	82	26.37	0	82	17.89	0	82	1.53	0	82	2.39	0	82	1.44	0	82	3.16	0
50a13	135	73.37	0	135	23.89	0	135	3.05	0	135	0.16	0	135	3.53	0	135	1.31	0
50a15	95	150.66	0	95	55.16	0	95	6.59	0	95	8.27	0	95	8.13	0	95	11.62	0
50a110	155	149.53	0	155	51.13	0	155	8.02	0	155	2.34	0	155	11.63	0	155	2.96	0
50a115	123	63.92	0	123	19.5	0	123	0.62	0	123	1.56	0	123	0.46	0	123	1.47	0
50a125	157	88.84	0	157	34.72	0	157	4.82	0	157	0.07	0	157	4.24	0	157	0.56	0
52a18	94	255.54	0	94	85.79	0	94	12.92	0	94	8.85	0	94	14.39	0	94	5.75	0
55a15	114	825.05	0	114	408.97	0	220	81.26	92.98	114	60.92	0	114	100.51	0	114	94.1	0
55a19	152	442.01	0	152	210.5	0	152	41.81	0	152	31.53	0	152	49.86	0	152	25.2	0
55a112	133	185.21	0	133	70.68	0	133	13.78	0	133	4.34	0	133	22.69	0	133	5.67	0
58a14	131	357.35	0	131	162.16	0	131	30.9	0	131	22.34	0	131	29.68	0	131	24.89	0
58a18	106	287.03	0	106	106.28	0	106	18.37	0	106	14.73	0	106	17.76	0	106	15.41	0
58a110	173	247.07	0	173	108.92	0	173	18.46	0	173	14.19	0	173	18.95	0	173	17.45	0
58a115	188	881.73	0	188	391.67	0	357	1834.07	90.05	188	62.25	0	188	1960.05	0	188	89.11	0
60a12	109	782.3	0	109	293.14	0	109	56.44	0	109	46.18	0	109	58.98	0	109	43.21	0
60a14	140	479.14	0	140	174.79	0	140	33.43	0	140	25.83	0	140	37.39	0	140	31.79	0
60a15	151	2476.25	0	151	832.96	0	197	1831.64	30.42	155	1802.74	2.65	151	2106.09	0	151	2076.01	0
60a16	147	584.69	0	147	248.97	0	147	45.53	0	147	39.34	0	147	53.72	0	147	48.15	0
60a110	157	1454.11	0	157	551.12	0	250	1829.88	59.51	157	87.19	0	157	2009.46	0	157	68.15	0
60a115	147	136.68	0	147	54.9	0	147	7.39	0	147	6.92	0	147	6.35	0	147	10.82	0
60a120	145	201.9	0	145	65.85	0	145	10.17	0	145	7.68	0	145	16.43	0	145	8.49	0
64a14	123	2512.85	0	123	884.15	0	156	1800.23	26.71	189	1820.52	53.66	123	2090.23	0	123	2111.2	0
64a15	91	6706.76	0	91	2289	0	152	1820.66	67.26	147	1798.92	61.54	91	2580.9	0	91	2558.91	0
64a18	147	4281.05	0	147	1612.31	0	201	1835.51	36.74	163	1797.01	10.88	147	2368.83	0	147	2329.68	0

Tabela B.1: Resultados dos algoritmos EX1, EX2, GRA, ILS, E2G e E2I, apresentados no Capítulo 4. A coluna Custo representa o custo da solução encontrada (em negrito: melhor solução encontrada; com “-”: solução não encontrada), T representa o tempo de execução (em segundos) e GAP representa o gap médio (em porcentagem) da solução do algoritmo em relação a melhor solução encontrada.

	EX1			EX2			GRA			ILS			E2G			E2I		
Instância	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP
64a10	129	2165.71	0	129	881.32	0	255	1798.82	97.56	176	1812.05	36.43	129	2087.61	0	129	2100.84	0
65a5	104	350.44	0	104	130.63	0	107	23.84	2.88	107	16.55	2.88	104	65.51	0	104	15.18	0
65a10	105	3082.34	0	105	1374.79	0	156	1833.22	48.21	111	1808.19	5.71	105	2287.94	0	105	2263.08	0
65a15	159	492.22	0	159	215.91	0	238	1804.8	49.79	159	29.33	0	159	1874.99	0	159	24.25	0
65a18	75	2530.67	0	75	1194.5	0	84	1827.51	12.38	82	1830.84	9.33	75	2221.5	0	75	2222.57	0
65a25	218	988.75	0	218	409.25	0	221	1812.56	1.55	218	68.03	0	218	1946.72	0	218	69.48	0
67a4	72	255.38	0	72	116.88	0	72	19.82	0	72	15.74	0	72	18.55	0	72	18.55	0
67a10	115	3339.38	0	115	1131.73	0	187	1833.88	62.85	124	1812.98	7.83	115	2209.76	0	115	2185.89	0
67a20	171	758.76	0	171	326.32	0	253	1825.89	47.71	171	51.55	0	171	1930.8	0	171	56.92	0
69a3	85	736.87	0	85	307.62	0	163	1827.99	91.76	85	46.68	0	85	1924.42	0	85	47.27	0
69a9	156	3390.79	0	156	1196.3	0	272	1824.21	74.39	175	1835.57	12.18	156	2220.57	0	156	2228.23	0
69a15	184	3275.63	0	184	1615.74	0	212	1799.7	15.28	203	1804.64	10.33	184	2331.54	0	184	2336.72	0
70a4	117	5032.7	0	117	2219.2	0	211	1814.27	80.33	119	1805.55	1.71	117	2553.47	0	117	2542.44	0
70a5	130	5170.44	0	130	1797.52	0	143	1814.09	10.31	164	1799.48	26.15	130	2409.66	0	130	2396.57	0
70a7	122	2833.38	0	122	1116.38	0	127	1814.87	4.12	158	1826.25	29.51	122	2181.91	0	122	2193.66	0
70a10	115	303.45	0	115	104.97	0	115	18.59	0	115	16.05	0	115	19.78	0	115	13.67	0
70a15	166	906.64	0	166	307.39	0	166	58.86	0	166	45.83	0	166	62.49	0	166	42.43	0
70a18	126	3415.7	0	126	1135.39	0	249	1797.44	97.59	132	1826.85	4.76	126	2169.82	0	126	2202.11	0
70a20	211	1293.09	0	211	562.13	0	239	1801.49	13.47	319	1819.73	51.18	211	1987.51	0	211	2004.18	0
70a25	127	549.97	0	127	193.79	0	127	34.23	0	127	28.98	0	127	31.72	0	127	30.92	0
72a3	123	6204.2	0	123	2278.78	0	159	1838	29.62	140	1828.83	13.82	123	2593.35	0	123	2586.43	0
72a5	137	637.6	0	137	247.67	0	271	1823.46	97.69	137	35.04	0	137	1900.1	0	137	32.11	0
72a8	111	276.64	0	111	100.01	0	111	15.49	0	111	12.29	0	111	16.5	0	111	13.73	0
72a10	135	3148.15	0	135	1335.95	0	139	1803.29	2.9	142	1815.79	5.19	135	2245.11	0	135	2255.44	0
72a15	124	1280.68	0	124	457.36	0	149	1817.83	20.53	165	1805.66	33.06	124	1967.39	0	124	1954.77	0
72a18	165	1912.05	0	165	792.73	0	170	1822.36	3.07	170	1800.26	3.03	165	2081.62	0	165	2061.05	0
72a20	96	3560.13	0	96	1497.86	0	129	1818.38	34.42	99	1821.62	3.13	96	2315.43	0	96	2317.94	0
72a25	195	2308.06	0	195	890.82	0	346	1798.92	77.58	208	1815.65	6.67	195	2090.48	0	195	2111.89	0
75a3	92	6894.99	0	92	3644.64	0	113	1829.98	23.16	95	1800.06	3.26	92	3042.69	0	92	3010.44	0
75a5	69	328.08	0	69	108.09	0	69	20.75	0	69	13.71	0	69	22.93	0	69	19.07	0
75a10	139	2016.93	0	139	810.16	0	263	1833.61	89.05	143	1822.63	2.88	139	2102.95	0	139	2091.04	0
75a15	120	6433.27	0	120	6430.24	0	134	1828.54	11.38	159	1836.62	32.5	120	3967.63	0	120	3974.11	0
75a20	147	1011.73	0	147	452.62	0	183	1831.64	24.72	172	1809.4	17.01	147	1980.01	0	147	1958.96	0
75a25	157	2760.05	0	157	946.31	0	170	1817.94	8.1	158	1806.78	0.64	157	2129.46	0	157	2118.48	0
75a30	171	2472.34	0	171	1087.4	0	201	1798.22	17.45	197	1799.24	15.2	171	2158.42	0	171	2158.33	0
78a4	129	2438.58	0	129	1186.32	0	204	1831.12	58.12	135	1810.69	4.65	129	2220.18	0	129	2203.53	0
78a5	124	6219.55	0	124	2525.97	0	215	1801.05	73.38	155	1798.55	25	124	2638.67	0	124	2636.54	0
78a8	123	7033.98	0	123	2483.13	0	175	1807.13	42.07	151	1829.16	22.76	123	2631.87	0	123	2651.36	0

Tabela B.2: Resultados dos algoritmos apresentados no Capítulo 4.

	EX1			EX2			GRA			ILS			E2G			E2I		
Instância	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP
78a110	3565	7201.44	2428.37	1960	7203.25	1290.07	3258	1799.91	2210.98	2438	1821.58	1628.84	141	4197.43	0	141	4219.76	0
78a115	151	3436.42	0	151	2448.33	0	260	1825.15	71.91	202	1816.65	33.77	151	2639.37	0	151	2629.56	0
78a120	170	4323.41	0	170	4172.99	0	312	1811.69	83.4	193	1801.15	13.53	170	3197.61	0	170	3189.35	0
78a130	2341	7207.31	4156.36	356	7008.3	547.27	464	1827.78	743.65	202	1836.47	266.86	55	4160.13	0	55	4167.88	0
80a12	102	4208.66	0	102	2477.8	0	188	1798.57	84.75	137	1832.82	34.31	102	2618.91	0	102	2653.12	0
80a14	92	3464.77	0	92	646.88	0	138	1831.71	50.35	98	1820.94	6.52	92	2040.27	0	92	2032.99	0
80a15	99	4329.52	0	99	779.62	0	136	1808.72	37.05	102	1835.26	3.03	99	2064.46	0	99	2089.95	0
80a18	143	3632.46	0	143	2797.84	0	234	1801.4	63.73	163	1829.19	13.99	143	2731.99	0	143	2758.47	0
80a110	134	3683.73	0	134	5988.94	0	181	1803.85	34.8	140	1835.5	4.48	134	3793.2	0	134	3830.53	0
80a115	162	4113.3	0	162	1491.03	0	287	1816.99	76.92	214	1812.5	31.94	162	2309.62	0	162	2308.19	0
80a118	154	3957.16	0	154	1627.6	0	246	1827.91	59.91	169	1815.28	9.74	154	2369.71	0	154	2355.43	0
80a120	4534	7198.03	2963.51	4051	7203.21	2637.16	5682	1836.62	3739.2	2276	1802.03	1437.91	148	4232.37	0	148	4196.26	0
80a122	155	3932.12	0	155	3570.63	0	183	1799.65	17.8	165	1806.93	6.45	155	2986.55	0	155	2990.7	0
80a130	191	3869.39	0	191	4090.25	0	304	1831.35	59.35	202	1810.58	5.76	191	3193.26	0	191	3170.88	0
83a14	142	4276.44	0	142	1318.3	0	276	1810.32	94.67	153	1805.37	7.75	142	2246.76	0	142	2240.28	0
83a15	5744	7204.79	4608.2	4273	7201.75	3402.46	6080	1805.86	4883.57	4341	1812.72	3458.06	122	4203.06	0	122	4207.96	0
83a18	138	3834.34	0	138	1105.91	0	249	1822.89	80.62	167	1816.69	21.01	138	2186.58	0	138	2178.53	0
83a110	204	7213.61	249.8	97	7200.35	66.33	169	1797.98	189.05	144	1825.2	147.37	97	7210.09	66.33	58	7200.79	0
83a115	4276	7204.3	615.84	3492	7200.61	484.59	3602	1808.52	502.99	2023	1814.29	238.64	973	7205.2	62.93	597	7197.59	0
83a118	527	7201.15	750	491	7200.01	691.94	833	1832.65	1243.3	394	1809.61	536.16	62	4231.32	0	62	4202.25	0
83a124	208	7195.05	147.62	84	6172.76	0	102	1819.47	20.86	96	1799.8	14.29	84	3876.13	0	84	3852.61	0
83a130	181	1682.21	0	181	959.36	0	306	1804.63	68.81	193	1808.99	6.63	181	2122.93	0	181	2124.81	0
85a14	163	7217.92	24.43	131	3161.1	0	229	1820.64	74.64	182	1817.13	38.93	131	2868.82	0	131	2866.65	0
85a15	111	6608.35	0	111	4206.25	0	170	1797.51	52.74	129	1820.94	16.22	111	3194.71	0	111	3219.26	0
85a18	4685	7196.1	3618.25	4442	7202.74	3425.4	6687	1806.97	5206.78	3996	1830.99	3071.14	2661	7206.22	2012.3	126	4228.34	0
85a110	1508	7202.69	382.6	1229	7202.81	293.31	1244	1827.99	298.06	1210	1825.54	287.13	748	7199.96	139.32	312	7217.21	0
85a115	232	7196.82	23.4	188	6052.72	0	365	1815.44	94.3	201	1822.73	6.91	188	3831.45	0	188	3835.15	0
85a118	166	7218.12	12.16	148	3155.47	0	271	1829.02	83.25	217	1829.78	46.62	148	2875.24	0	148	2877.91	0
85a122	198	7211.77	18.56	167	5214.28	0	261	1819.35	56.02	181	1809.3	8.38	167	3553.47	0	167	3544.05	0
85a125	201	7202.08	8.06	186	4196.9	0	362	1825.8	94.61	201	1804.13	8.06	186	3220.19	0	186	3197.08	0
85a130	229	7217.03	3.15	222	3482.8	0	424	1831.85	90.84	268	1810.1	20.72	222	2987.2	0	222	2966.59	0
90a13	155	7212.46	143.7	105	7201.25	65.09	130	1809.91	104.37	126	1814.63	98.1	103	7197.72	61.94	64	7215	0
90a14	162	7204.02	19.12	136	513.14	0	277	1831.67	103.73	157	1812.23	15.44	136	1997.5	0	136	1976.36	0
90a15	109	7220.31	17.2	93	3030.31	0	149	1814.33	60.53	99	1813.64	6.45	93	2816.87	0	93	2818.9	0
90a18	132	7207.19	8.2	122	3110.92	0	151	1799.64	23.49	179	1828.5	46.72	122	2835.27	0	122	2861.11	0
90a110	285	7221.61	387.76	267	7200.24	356.95	348	1817.26	496.35	248	1802.64	324.43	126	7198.08	116.19	58	7210.47	0
90a115	224	7214.36	17.89	190	3010.28	0	330	1799.34	73.59	255	1822.8	34.21	190	2801.47	0	190	2822.08	0
90a120	1946	7205.58	65.83	1899	7203.85	61.83	3390	1815.29	188.91	1959	1829.34	66.98	1392	7217.97	18.61	1173	7220.32	0

Tabela B.3: Resultados dos algoritmos apresentados no Capítulo 4.

	EX1			EX2			GRA			ILS			E2G			E2I		
Instância	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP
90al25	202	7199.44	13.48	178	2792.77	0	356	1836.93	100.27	220	1815.81	23.6	178	2764.79	0	178	2745.48	0
90al30	176	7201.51	15.03	153	3076.63	0	292	1812.48	90.87	218	1822.57	42.48	153	2834.71	0	153	2842.32	0
95al3	5323	7216.11	236.63	3644	7202.8	130.45	6932	1834.3	338.4	4685	1812.73	196.25	2176	7202.88	37.58	1581	7204.32	0
95al5	4845	7211.74	196.01	4669	7201.63	185.25	9302	1805.77	468.32	3689	1818.07	125.39	3401	7213.97	107.8	1637	7197.81	0
95al8	231	7221.93	124.27	103	7109.97	0	211	1831.08	105.22	115	1805.68	11.65	103	4987.53	0	103	3533.92	0
95al10	6324	7220.17	4449.64	5839	7203.26	4100.72	8799	1832.84	6229.9	8369	1834.11	5921.05	139	5031.77	0	139	3647.1	0
95al15	2836	7217.45	2366.09	2670	7203.58	2221.74	4211	1826.16	3561.72	2470	1816.21	2047.71	115	5664.58	0	115	3207.45	0
95al20	5148	7212.39	2448.51	4060	7201.8	1909.9	6036	1809.1	2888.13	5020	1801.65	2385.28	2623	7205.72	1198.72	202	3947.03	0
95al25	3868	7200.49	2258.23	3647	7202.96	2123.78	7178	1817.29	4276.84	2909	1823.1	1673.5	164	4217.62	0	164	3771.29	0
95al28	182	7206.23	21.33	150	2900.23	0	165	1796.76	9.98	174	1822.11	16	150	2762.62	0	150	2901.39	0
100al4	9221	7201.64	10378.78	4902	7196.67	5470.45	9532	1836.89	10731.8	6336	1823.1	7099.66	4672	7204.72	5209.12	88	3814.46	0
100al5	7478	7202.38	6346.34	5465	7200.78	4611.21	6130	1818.14	5184.83	7971	1831.86	6771.85	116	4213.32	0	116	3906.78	0
100al10	9790	7219.46	6426.62	9220	7201.98	6046.67	9305	1827.07	6103.27	9886	1829.99	6490.35	5536	7207.64	3590.37	150	3827.53	0
100al15	7905	7208.32	6714.77	4963	7201.52	4178.45	5018	1825.65	4225.76	4296	1806.88	3603.31	1946	7214.95	1577.58	116	4019.97	0
100al20	8290	7206.75	5463.82	5454	7201.09	3560.4	8685	1824.29	5728.98	6116	1812.13	4004.68	149	4219.97	0	149	3810.97	0
100al25	10157	7207.23	5874.99	9962	7199.76	5760	14151	1816.39	8224.14	11464	1822.71	6643.28	170	4214.1	0	170	3759.74	0
100al28	6186	7197.41	6276.94	3874	7200.05	3893.81	6636	1822.75	6741.41	2924	1815.52	2914.55	1842	7197.33	1799.26	97	4378.8	0
110al5	20644	7219.73	14438.27	10667	7201.3	7411.97	17174	1835.11	11994.53	10812	1802.74	7513.87	5967	7222.25	4101.87	142	5109.16	0
110al10	12419	7201.94	7037.16	10672	7199.96	6033.33	16713	1797.34	9504.97	12663	1822.39	7177.45	11923	7202.27	6752.23	174	3739.44	0
110al15	12203	7214.46	9287.19	10879	7199.47	8268.46	14849	1835.79	11322.22	8477	1833.71	6420.41	130	4233.41	0	130	4138.69	0
110al20	9271	7221.32	4993.75	8944	7199.99	4814.29	17020	1805.68	9251.85	12903	1798.65	6989.48	11256	7212.54	6084.42	182	3869.07	0
110al25	12029	7217.57	8744.82	11193	7199.6	8130.15	7163	1809.85	5167.24	6605	1813.05	4756.78	136	5647.61	0	136	6157.83	0
110al30	14546	7218.89	7848.9	10394	7199.09	5579.78	6843	1814.26	3639.59	10744	1835.06	5771.11	183	4863.36	0	183	3878.25	0
120al5	19448	7207.26	21271.17	10943	7202.98	11925.27	15478	1816.85	16908.59	14564	1811.18	15904.08	91	5240.16	0	91	4265.18	0
120al10	16416	7214.81	13139.01	10999	7199.97	8770.16	15461	1807.96	12368.47	8848	1800.97	7035.67	124	4860.85	0	124	3911.82	0
120al15	13387	7215.88	11746.66	10087	7200.21	8826.55	11953	1808.51	10477.56	9077	1823.49	7932.79	113	6604.76	0	113	6507.21	0
120al20	12052	7200.04	6989.38	11257	7200.5	6521.76	8467	1834.7	4880.35	6431	1798.04	3682.72	4967	7220.24	2821.59	170	3904.81	0
120al25	21723	7211.28	12456.83	11757	7200.21	6695.95	11447	1810.39	6516.66	10818	1804.66	6153.4	4930	7204.65	2749.97	173	5347.86	0
120al30	16352	7201.25	9407.2	10298	7200.32	5887.21	10566	1832.04	6042.9	11795	1817.85	6757.63	172	5256.41	0	172	3778.87	0
130al5	20942	7217.32	16261	12004	7202.47	9278.13	11094	1818.17	8566.85	7527	1808.81	5780.34	128	6614.99	0	128	4440.34	0
130al10	18304	7194.51	14781.01	12632	7200.03	10169.92	6802	1803.98	5429.75	10803	1817.78	8683.23	123	6000.15	0	123	4139.78	0
130al15	16736	7201.19	12675.91	13327	7200.9	10073.28	15312	1808.56	11588.25	18150	1828.92	13755.29	131	5641.67	0	131	3940.86	0
130al20	25226	7220.29	15005.33	13092	7203.43	7739.52	10418	1831.44	6138.58	16342	1802.81	9685.82	8759	7200.29	5145.13	167	5894.36	0
130al24	20382	7220.44	13671.55	12903	7200.51	8618.24	11653	1810.77	7773.47	9726	1830.54	6471.94	148	6610.32	0	148	6689.61	0
130al28	-	7212.39	-	12600	7200.43	6900	17868	1812.56	9826.63	8606	1821.46	4680.91	8440	7195.81	4589.07	180	5140.36	0
140al5	14995	7207.11	127.27	14179	7202.02	114.9	9960	1829.2	50.96	15379	1822.26	133.09	15344	7208.64	132.56	6598	7217.2	0
140al10	-	7203.53	-	12059	7203.91	792.3	21098	1800.2	1461.13	6291	1818.07	365.5	3896	7200.29	188.31	1351	7206.62	0
140al15	21632	7203.49	978.04	12543	7201.25	525.09	17563	1818.08	775.28	16281	1802.98	711.37	8893	7215.8	343.19	2007	7196.38	0

Tabela B.4: Resultados dos algoritmos apresentados no Capítulo 4.

Instância	EX1			EX2			GRA			ILS			E2G			E2I		
	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP
140al20	19299	7214.65	153.18	14143	7203.14	85.54	9573	1825.01	25.58	17465	1820.01	129.13	14344	7202.04	88.17	7623	7196.28	0
140al25	19948	7204.79	1028.69	12549	7202.54	610.06	10857	1815.01	514.34	13237	1800.29	649	6662	7203.83	276.93	1767	7198.83	0
140al30	15461	7214.25	131.67	13467	7199.47	101.8	14643	1816.54	119.41	18377	1817.45	175.37	13362	7204.35	100.22	6674	7211.55	0
150al5	-	7204.92	-	13663	7205.1	942.83	14943	1806.82	1040.51	11062	1812.05	744.33	6353	7215.08	384.93	1310	7206.94	0
150al10	-	7199.93	-	14430	7204.34	806.96	21876	1822.16	1274.97	11738	1806.13	637.74	5740	7197.55	260.76	1591	7215.42	0
150al15	-	7196.79	-	14539	7202.51	369.2	17773	1801	473.57	9578	1811.51	209.09	5748	7214.44	85.5	3099	7201.77	0
150al20	-	7197.8	-	14204	7202.39	314.98	18075	1821.24	428.07	18293	1837.26	434.45	8029	7197.97	134.56	3423	7201.08	0
150al25	14837	7195.18	732.92	14266	7201.01	700.84	15640	1800.54	777.97	14345	1823.96	705.26	10666	7222.83	498.75	1781	7218.56	0
150al30	17241	7198.66	573.99	14327	7201.47	460.09	16885	1820.74	560.09	15386	1817.84	501.49	7789	7220.96	204.49	2558	7208.92	0
160al5	-	7199.22	-	15657	7204.89	1087.26	26155	1818.94	1883.3	13734	1814.33	941.41	8588	7197.21	551.23	1319	7199.65	0
160al10	-	7207.28	-	14426	7204.59	1125.29	18349	1821.77	1458.48	19548	1809.53	1560.31	8838	7214.29	650.7	1177	7218.62	0
160al15	16290	7203.19	315.33	16096	7202.71	310.39	15294	1811.96	289.95	10949	1814.84	179.16	8145	7217.49	107.68	3922	7204.15	0
160al20	-	7218.56	-	14505	7200.3	251.58	21072	1802.28	410.75	20831	1836.12	404.91	9500	7195.55	130.26	4126	7200.3	0
160al25	16409	7212.38	387.61	15463	7204.47	359.51	20522	1819.56	509.85	15393	1819.1	357.42	6407	7209.48	90.4	3365	7199.38	0
160al30	22210	7196.98	1426.27	15813	7202.04	986.65	14071	1805.22	866.91	21429	1800.42	1372.55	10056	7219.65	591.04	1455	7206.44	0
170al5	-	7207.5	-	-	7205.12	-	43245	1830.93	178.53	43756	1798.31	181.82	27845	7211.42	79.35	15526	7205.08	0
170al10	-	7217.63	-	-	7212.3	-	64322	1800.57	728.47	81136	1804.24	945.03	53903	7204.98	594.27	7764	7205.04	0
170al15	22249	7199.08	1134.83	17135	7205.29	850.98	18938	1836.19	951.05	24519	1809.4	1260.78	11864	7212.44	558.43	1802	7207.88	0
170al20	-	7220.96	-	15221	7203.27	820.22	13771	1798.45	732.57	11821	1831.38	614.66	11090	7198.78	570.5	1654	7203.7	0
170al25	-	7208.17	-	18758	7201.37	153.36	14986	1797.35	102.41	20503	1805.16	176.93	13263	7215.6	79.14	7404	7219.09	0
170al30	25237	7207.2	1675.16	16422	7203.38	1055.12	13652	1828.2	860.29	13400	1807.17	842.53	7450	7212.91	424.04	1422	7195.32	0
180al5	-	7208.06	-	-	7211.56	-	32453	1803.81	1200.17	18228	1830.49	630.25	10495	7195.48	320.48	2496	7220.65	0
180al10	-	7209.08	-	-	7208.6	-	64322	1825.53	597.2	75560	1804.99	719.01	40138	7207.98	335.06	9226	7200.52	0
180al15	-	7212.31	-	-	7199.11	-	43534	1803.66	501.41	21929	1826.46	202.95	13234	7212.8	82.83	7239	7197.33	0
180al20	-	7219.6	-	-	7210.24	-	85469	1823.45	129.05	116496	1833.33	212.2	107068	7208.59	186.94	37314	7214.62	0
180al24	24618	7215.7	473.27	18956	7204.12	341.42	24207	1828.46	463.69	13036	1812.48	203.56	7896	7208.39	83.87	4294	7222.16	0
180al30	25002	7220.2	303.32	18294	7204.89	195.11	25153	1820.31	305.76	24355	1814.66	292.88	13429	7220.58	116.63	6199	7211.68	0
190al5	-	7207.56	-	-	7213.08	-	56452	1805.54	209.43	60294	1802.85	230.48	37281	7221.99	104.34	18244	7199.72	0
190al10	-	7218.19	-	-	7206.58	-	75464	1808.58	967.45	42594	1833.43	502.5	23651	7210.02	234.55	7070	7207.66	0
190al15	-	7222.84	-	-	7206.78	-	77683	1799.06	383.93	86438	1837.96	438.47	49498	7200.97	208.35	16053	7210.91	0
190al20	-	7196.82	-	-	7205.96	-	12334	1833.74	459.7	10647	1813.03	383.13	9812	7218.75	345.24	2204	7210.87	0
190al25	25421	7213.79	200.4	17910	7205.7	111.65	20681	1816.02	144.39	17436	1823.27	106.04	15799	7213.36	86.7	8462	7208.83	0
190al30	31538	7201.63	1450.9	17911	7297.34	780.78	14883	1802.92	631.88	12059	1802.86	493.03	4906	7218.65	141.24	2034	7207.9	0
200al5	-	7206.33	-	-	7199.77	-	24353	1820.42	506.56	30060	1810.95	648.71	16245	7219.06	304.61	4015	7218.5	0
200al10	-	7203.87	-	-	7205.12	-	75651	1825.21	360.48	45193	1822.12	175.08	35184	7215.32	114.16	16429	7217.53	0
200al15	-	7202.48	-	-	7209.1	-	36576	1832.98	1132.94	23516	1799.75	692.69	10003	7203.77	237.19	2967	7217.72	0
200al20	20432	7210.47	68.92	18306	7204.29	51.34	14163	1824.13	17.09	24593	1802.78	103.32	21284	7197.71	75.97	12096	7220.45	0
200al25	20644	7201.08	1443.4	20462	7209.15	1429.8	19645	1832.86	1368.7	12543	1800.71	837.75	6492	7198.12	385.36	1338	7199.24	0

Tabela B.5: Resultados dos algoritmos apresentados no Capítulo 4.

Instância	EX1			EX2			GRA			ILS			E2G			E2I		
	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP	Custo	T	GAP
200al30	-	7202.78	-	-	7206.73	-	56586	1809.14	310.83	74488	1812.69	440.81	52302	7204.56	279.73	13773	7207.58	0
210al5	-	7213.3	-	-	7206.92	-	25468	1828.94	840.55	26802	1799.59	889.82	14826	7221.62	447.53	2708	7215.41	0
210al10	-	7213.93	-	-	7206.08	-	75428	1829.11	451.82	70610	1796.97	416.57	50320	7210.93	268.13	13669	7211.55	0
210al15	-	7208.27	-	-	7203.42	-	34569	1804.8	599.53	20807	1817.87	321.05	15767	7195.03	219.06	4942	7215.44	0
210al20	-	7210.89	-	-	7201.74	-	24368	1824.81	129.49	31387	1810.4	195.58	19574	7216.89	84.34	10619	7198.88	0
210al25	26223	7205.88	752.96	21364	7209.68	594.91	29048	1805	844.84	25793	1831.41	738.97	15453	7208.64	402.63	3074	7197.22	0
210al30	26902	7195.77	257.48	20101	7209.88	167.11	18838	1804.44	150.32	19531	1813.67	159.53	16290	7207.48	116.47	7525	7202.27	0
220al5	-	7204.24	-	-	7202.91	-	67687	1817.19	826.6	49593	1804.67	578.91	25441	7216.57	248.27	7305	7210.64	0
220al10	-	7216.06	-	-	7203.31	-	46758	1799.04	911.5	46616	1831.64	908.44	37144	7212	703.52	4623	7209.11	0
220al15	-	7194.04	-	-	7202	-	72532	1800.49	716.14	62082	1816.97	598.56	30858	7207.97	247.22	8887	7214.05	0
220al20	-	7217.89	-	20801	7211.69	699.69	20108	1816.42	673.04	29580	1827.61	1037.19	12214	7217.73	369.56	2601	7195.18	0
220al25	-	7199.28	-	20676	7212.68	1349.86	28339	1814.34	1887.17	16963	1835.75	1089.48	7135	7209.08	400.33	1426	7216.61	0
220al30	45227	7204.88	801	22278	7205.9	343.81	30249	1808.69	502.61	29557	1810.03	488.82	14100	7195.69	180.89	5020	7204.13	0
300al5	-	7219.84	-	-	7202.6	-	45674	1823.04	229.31	58445	1807.02	321.39	48032	7219.29	246.32	13869	7212.56	0
300al10	-	7198.3	-	-	7208.62	-	21352	1813.12	522.54	16670	1833.73	386.02	16219	7199.09	372.89	3430	7203.71	0
300al15	-	7220.9	-	-	7206.54	-	86574	1815.48	337.45	90982	1831.57	359.72	56566	7214.08	185.82	19791	7206.54	0
300al20	-	7196.31	-	-	7205.38	-	65657	1821.4	288.98	67770	1805.91	301.5	45296	7218.33	168.35	16879	7198.37	0
300al25	-	7211.47	-	24512	7200.9	117.83	33949	1825.85	201.69	37452	1819.43	232.82	20154	7203.81	79.1	11253	7222.49	0
300al30	36534	7197.2	1358.52	27900	7205.22	1013.83	17296	1827.81	590.47	16811	1814.24	571.13	14383	7197.7	474.22	2505	7221.51	0
400al5	-	7205.37	-	-	7202.59	-	65853	1829.93	246.36	93383	1823.52	391.15	40075	7220.77	110.78	19013	7221.15	0
400al10	-	7196.56	-	-	7203	-	54351	1810	415.23	33710	1830.02	219.57	22304	7202.86	111.44	10549	7213.1	0
400al15	-	7197.16	-	-	7202.04	-	24368	1815.02	659.22	35932	1835.45	1019.52	18937	7200.5	489.99	3210	7214.93	0
400al20	-	7198.98	-	29992	7210.15	109.26	35361	1806.98	146.72	44150	1810.31	208.05	43494	7198.02	203.46	14332	7213.49	0
400al25	-	7200.91	-	29032	7198.6	72.68	24858	1820.63	47.85	30133	1798.4	79.23	29663	7195.57	76.43	16813	7216.52	0
400al30	-	7208.42	-	34118	7200.87	485.6	34939	1797.42	499.7	36562	1837.06	527.54	15225	7209	161.31	5826	7203.05	0
1000al5	-	7221.74	-	-	7206.8	-	77685	1818.98	138.67	87843	1804.46	169.88	64303	7201.66	97.56	32549	7201.61	0
1000al10	-	7212.32	-	-	7204.64	-	34456	1813.04	220.17	29878	1809.29	177.63	19069	7219.57	77.19	10762	7197.11	0
1000al15	-	7208.08	-	-	7206.53	-	43789	1824.45	1034.24	54044	1801.39	1299.87	29647	7219.79	667.93	3861	7197.94	0
1000al20	-	7210.78	-	-	7203.94	-	65783	1810.64	283.83	85406	1802.71	398.32	42780	7212.58	149.61	17139	7212.57	0
1000al25	-	7221.54	-	-	7203.2	-	125786	1827.54	132.72	153290	1804.13	183.6	141395	7196.65	161.6	54050	7215.82	0
1000al30	-	7210.84	-	52987	7220.1	95.99	89232	1832.38	230.05	65512	1825.14	142.31	57917	7218.5	114.22	27036	7211.3	0

Tabela B.6: Resultados dos algoritmos apresentados no Capítulo 4.