UNIVERSIDADE FEDERAL FLUMINENSE

LEONARDO MARICATO MUSMANNO

Algoritmos aproximados para o problema do grafo mediano generalizado

NITERÓI

2013

UNIVERSIDADE FEDERAL FLUMINENSE

LEONARDO MARICATO MUSMANNO

Algoritmos aproximados para o problema do grafo mediano generalizado

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:
CELSO CARNEIRO RIBEIRO

NITERÓI

2013

LEONARDO MARICATO MUSMANNO

Algoritmos aproximados para o problema do grafo mediano generalizado

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 em Março de 2013.

BANCA EXAMINADORA

Prof. Celso Carneiro Ribeiro - Orientador, IC-UFF

Prof. Fábio Protti, IC-UFF

Prof. Lilian Markenzon, PPGI-UFRJ

Prof. Maria Claudia Silva Boeres, PPGI-UFES

Prof. Simone Martins, IC-UFF

Niterói

2013

Agradecimentos

Aos professores do Instituto de Computação UFF pelo conhecimento recebido durante todo o curso de mestrado, em especial ao professor Celso Carneiro Ribeiro por me aceitar como seu orientando, pelo apoio, dedicação e tempo despendidos, que foram de extrema importância para a conclusão desse trabalho com êxito. Agradeço pela convivência, pela disposição em compartilhar seu conhecimento e principalmente por acreditar em mim. Agradeço aos meus pais pela paciência, simplicidade e sabedoria. Aos meus irmãos pela compreensão, carinho e conselhos. A minha namorada, pela força, companheirismo e motivação. Aos os meus amigos de infância, que sempre me estenderam a mão para me apoiar. Agradeço aos queridos amigos da UFF que conheci ao longo dessa trajetória, pela amizade e parceria, que foram de essenciais para me estimular nessa jornada. Aos funcionários da UFF pela cooperação e disposição em me auxiliar sempre que precisei.

Resumo

O conceito de similaridade entre objetos é fundamental na área de reconhecimento de padrões. Na chamada "abordagem estrutural", grafos são frequentemente utilizados para representar objetos. Assim sendo, é preciso definir um meio de medir a similaridade entre dois grafos. Uma das ferramentas mais utilizadas para realizar essa medição é a "distância de edição", que consiste em medir a distância entre dois grafos de acordo com o grau de distorção necessário para transformar um grafo no outro. O grafo mediano generalizado de um conjunto de grafos S é aquele que minimiza a soma das distâncias dos grafos de S a ele e que melhor captura as informações desse conjunto de grafos, podendo ser considerado um representante deste conjunto. O conceito de grafo mediano já foi aplicado com sucesso em áreas como reconhecimento de símbolos gráficos, síntese de símbolos gráficos e clusterização de imagens, entre outros. No entanto, computar o grafo mediano generalizado de um conjunto de grafos é uma tarefa complexa. Por si só, a versão de decisão do problema de cálculo da distância de edição entre dois grafos é NP-Completo. Algoritmos exatos lidam apenas com grafos de tamanho relativamente pequeno, sendo de pouca utilidade prática. Nesta dissertação são propostos três algoritmos aproximados para o problema, sendo dois baseados em estratégias gulosas e outro baseado no princípio do algoritmo A^*. Os resultados obtidos indicam que os algoritmos podem ser utilizados para encontrar soluções aproximadas de boa qualidade em tempos computacionais razoáveis.

Palavras-chave: Reconhecimento de padrões, correspondência entre grafos, distância de edição, algoritmos gulosos, algoritmo A^*, grafo mediano.

Abstract

The concept of similarity is of great importance in pattern recognition. In the structural approach of problems, graphs are frequently used to represent objects. Thus, it is necessary to define a way to measure the similarity between two graphs. One of the most used tools to measure graph similarity is the graph edit distance, which basically consists in measuring the distance between two graphs based on the amount of distortion needed to transform one graph G_1 into another graph G_2. The generalized median graph of a set S of graphs is the graph that minimizes the sum of distances between that graph and the graphs in S. The generalized median graph of a set S is the graph that best captures the information contained in S, that is, it may be regarded as a representative of the set. The concept of the generalized median graph has been successfully applied in optical character recognition, graphical symbol recognition, among others. However, the task of computing the generalized median graph is a complex task. The search space of the problem grows exponentially as the number of graphs in S grows. In addition, the decision version of the graph edit distance problem is NP-Complete. Exact methods deal only with small graphs, having little practical value. In this work, three approximate algorithms are proposed, two of them based on greedy strategies and the other based on the principle of the A^* algorithm. Results indicate that the proposed heuristics can be used to obtain good approximate solutions in low computational time.

Keywords: Pattern recognition, graph matching, graph edit distance, greedy algorithms, A^* algorithm, median graph.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

$MaxSub(G_1, G_2)$	:	máximo subgrafo comum dos grafos G_1 e G_2;
$MaxSub(S)$	:	máximo subgrafo comum dos grafos do conjunto S;
$MinSuper(G_1, G_2)$	:	mínimo supergrafo comum dos grafos G_1 e G_2;
$MinSuper(S)$	:	mínimo supergrafo comum dos grafos do conjunto S;
$\hat{G}$	:	grafo mediano;
$\bar{G}$	:	grafo mediano generalizado;

Sumário

Capítulo 1

Introdução

1.1 Motivação

Duas abordagens são mais frequentemente usadas em problemas de reconhecimento de padrões: a abordagem estatística e a abordagem estrutural. Segundo a abordagem estatística, os objetos são representados por vetores rotulados. Nesse caso, um objeto é formalmente representado por um vetor consistindo de n medidas, que pode ser visto como um ponto $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. As vantagens de representar objetos dessa forma decorrem das operações matemáticas bem definidas no espaço de vetores (tais como, a soma, o produto ou a distância entre dois vetores) e na eficiência com que se pode computar essas operações. Entretanto, também há desvantagens em se utilizar a representação vetorial. Não é possível, por exemplo, representar relações entre as partes de um objeto. Além disso, objetos de tamanhos e complexidades distintas são representados por vetores de mesmo tamanho.

A abordagem estrutural consiste em utilizar grafos na representação dos padrões ou objetos. A utilização de grafos permite superar os problemas da representação vetorial, uma vez que o número de vértices e arestas não é limitado a priori, podendo indicar as relações entre as partes de um objeto ou ser adaptados para representar a complexidade de qualquer padrão ou objeto. Uma desvantagem da representação por grafos é o fato de pouca estrutura matemática padronizada existir nesse domínio. Não existe, por exemplo, uma forma padronizada de somar ou fazer o produto entre dois grafos.

A combinação das vantagens dessas duas abordagens vêm sendo foco de atenção da comunidade de reconhecimento de padrões. Um passo nesse sentido foi utilizar grafos para representar objetos e associar rótulos a seus vértices e arestas. Daí decorre a definição

padrão de grafo rotulado utilizada em reconhecimento de padrões como uma quádrupla $G = (V, E, \mu, \upsilon)$, onde V é o conjunto de vértices, E é o conjunto de arestas, μ é uma função que associa rótulos aos vértices e υ é uma função que associa rótulos às arestas [9].

O conceito de similaridade entre objetos exerce um papel fundamental na área de reconhecimento de padrões. Assim sendo, é preciso definir um meio de medir a similaridade entre grafos, isto é, é preciso encontrar uma forma de comparar grafos e calcular o seu grau de semelhança. Para realizar as comparações, a "distância de edição" entre grafos é uma das ferramentas mais conhecidas e utilizadas. A ideia básica da distância de edição é atribuir custos às operações de edição (inserção e eliminação de vértices e inserção e eliminação de arestas) necessárias para transformar um grafo rotulado em outro.

Dado um conjunto de objetos, o conceito de mediana é frequentemente utilizado para indicar o elemento que melhor represente o conjunto, isto é, aquele que melhor capture as informações contidas nos elementos. O grafo mediano é um conceito bastante utilizado quando se quer encontrar um representante de um conjunto de grafos. Dado um conjunto S de grafos, o grafo mediano $\hat{G}$ é aquele que apresenta a menor soma das distâncias (SOD) para os grafos de S:

$$\hat{G} = \text{argmin}_{G' \in S} \sum_{G \in S} d(G', G),$$

onde $d(G', G)$ denota a distância de edição entre os grafos G' e G.

O grafo mediano de S é portanto um de seus elementos. Já o grafo mediano generalizado de S é definido como o grafo que apresenta a menor soma das distâncias para os grafos de S, independentemente de pertencer ou não a S.

O conceito de grafo mediano possui grande potencial para aplicações, podendo ser utilizado em problemas de classificação ou em qualquer situação em que seja preciso encontrar um representante para um conjunto de grafos [19, 20].

1.2 Objetivo

O objetivo principal deste trabalho é desenvolver algoritmos aproximados para o problema de encontrar o grafo mediano generalizado de um conjunto de grafos.

Os poucos algoritmos exatos existentes para o problema trabalham com grafos de no

máximo 20 vértices. Outro objetivo dessa dissertação consiste em desenvolver algoritmos capazes de lidar com grafos maiores em tempos computacionais reduzidos, sem perda de qualidade dos resultados obtidos.

Com esse fim, três algoritmos aproximados foram propostos. Dois deles são baseados em estratégias gulosas e o outro é baseado no princípio do algoritmo A^*.

1.3 Organização

No Capítulo 2 são apresentados os conceitos básicos e a notação que será usada no resto do trabalho. São definidos os conceitos de grafo e subgrafo e é introduzido o tema de correspondência exata ou inexata entre grafos. Também é apresentada a distância de edição, que será usada como a ferramenta para medir a similaridade entre os grafos. Finalmente, o conceito de grafo mediano é introduzido e são apresentadas algumas de suas propriedades. O capítulo contém também uma breve discussão sobre a complexidade do problema.

No Capítulo 3 são definidos os custos das operações de edição e o cálculo da distância entre dois grafos decorrente da escolha desses custos. São apresentadas propriedades do grafo mediano. Discute-se o espaço de busca do problema e os algoritmos auxiliares para solução do problema são apresentados. Também é apresentada a forma utilizada pelos algoritmos para acelerar a avaliação da qualidade de um candidato a grafo mediano. Finalmente, são apresentados os algoritmos propostos.

O Capítulo 4 mostra as instâncias utilizadas nos testes e apresenta uma análise dos resultados númericos obtidos pelos algoritmos.

No Capítulo 5 são apresentadas as conclusões e possíveis extensões do trabalho.

O apêndice A ilustra o funcionamento de um dos algoritmos gulosos. No Apêndice B encontra-se a descrição das instâncias utilizadas.

Capítulo 2

Grafo Mediano

2.1 Introdução

Neste capítulo serão introduzidos os conceitos necessários para a definição do problema de determinar o grafo mediano e o grafo mediano generalizado de um conjunto de grafos. Além disso, com o objetivo de mostrar a origem e a importância do problema, serão abordados conceitos de correspondência entre grafos. Realizar uma correspondência entre grafos consiste basicamente em verificar a similaridade entre as estruturas de dois grafos [13]. Dependendo do objetivo da correspondência, pode-se fazer uma distinção entre correspondência exata e correspondência inexata.

Na correspondência exata entre dois grafos, o objetivo é identificar se dois grafos são isomorfos ou se um subgrafo de um deles é isomorfo ao outro grafo. Na correspondência inexata utiliza-se uma medida da similaridade para obter-se a melhor correspondência entre os dois grafos.

2.2 Grafos e subgrafos

Um grafo $G = (V, E)$ é definido por um conjunto V de vértices e um conjunto E de arestas (representando pares de vértices), podendo as arestas possuírem pesos ou não. Essa definição nem sempre é adequada para representar objetos do mundo real. Para que grafos possam representar mais fielmente os objetos do mundo real, mais informações devem estar associadas a seus vértices e arestas. Com esse objetivo, são associados atributos (rótulos) aos vértices e arestas dos grafos.

Definição 2.1 *Dado um alfabeto finito L de atributos para vértices e arestas, um grafo*

rotulado $G = (V, E, \mu, \upsilon)$ *pode ser definido por uma quádrupla, onde* V *é um conjunto finito de vértices,* $E \subseteq V \times V$ *é um conjunto de arestas,* $\mu : V \to L$ *é uma função que associa atributos aos vértices, e* $\upsilon : E \to L$ *é uma função que associa atributos às arestas.*

O número de vértices $|V|$ de um grafo $G = (V, E)$ será denotado também por $|G|$, de forma a simplicar o texto, mesmo que isso caracterize um abuso de linguagem. Os vértices do conjunto V são representados por seus identificadores. As arestas são pares de nós $\{u, v\}$, onde $u, v \in V$.

O conjunto L de atributos pode ser de qualquer natureza, sem restrições sobre seus atributos. Por exemplo, pode-se ter $L = \mathbb{R}^n$ ou $L = \{A, B, C, \epsilon\}$. Esses atributos podem incluir o valor "nulo" (muitas vezes representado por ϵ), utilizado quando não é necessário associar atributo algum a um vértice ou aresta.

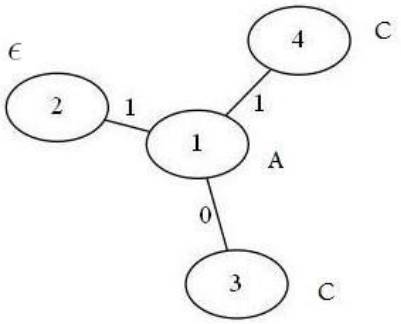


Figura 2.1: Grafo com atributos: os vértices são identificados por 1,2,3 e 4

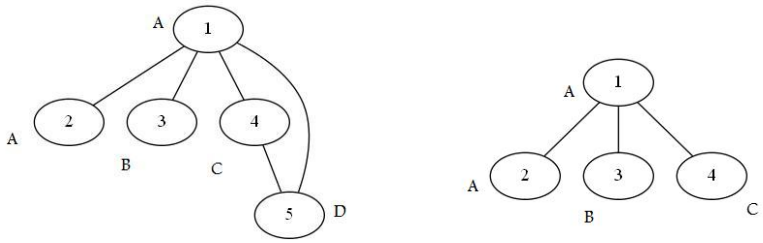
A Figura 2.1 exemplifica um grafo com atributos. Neste caso, $L = \{0, 1, A, B, C, \epsilon\}$, $V = \{1, 2, 3, 4\}$, $E = \{\{1, 2\}, \{1, 3\}, \{1, 4\}\}$, $\mu(1) = A$, $\mu(2) = \epsilon$, $\mu(3) = \mu(4) = C$, $\upsilon(1, 2) = \upsilon(1, 4) = 1$ e $\upsilon(1, 3) = 0$.

A definição usual de um grafo ponderado com pesos nas arestas é equivalente a associar o atributo "nulo" aos vértices ($\mu(u) = \epsilon, \forall u \in V$) e valores reais às arestas. Nesse caso, $L = \mathbb{R} \cup \{\epsilon\}$.

Definição 2.2 *Sejam* $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ *e* $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ *dois grafos rotulados. Diz-se que* G_1 *é um subgrafo de* G_2*, o que é denotado por* $G_1 \subseteq G_2$*, se* $V_1 \subseteq V_2$*,* $E_1 \subseteq E_2$*,* $\mu_1(u) = \mu_2(u), \forall u \in V_1$ *e* $\upsilon_1(e) = \upsilon_2(e), \forall e \in E_1$

Quando $E_1 = E_2 \cap (V_1 \times V_1)$, diz-se que G_1 é o subgrafo induzido em G_2 por V_1. Dado um grafo rotulado $G = (V, E, \mu, \upsilon)$, um subconjunto $V' \subseteq V$ determina unicamente um subgrafo G' induzido em G, normalmente representado por $G' = G(V')$. Deste ponto em

diante, exceto quando indicado o contrário, o termo *subgrafo* será utilizado com o mesmo sentido de *subgrafo induzido*.



(a) Grafo G (b) Grafo G'

Figura 2.2: Grafo G e grafo G' induzido em G por $V' = \{1, 2, 3, 4\}$

2.3 Correspondência entre grafos

Em muitos campos, como a química ou a biologia molecular, existem aplicações em que é necessário processar imagens e localizar regiões nessas imagens. Quando esse processamento é feito automaticamente por um computador, sem o auxílio de um especialista, uma maneira eficiente de representar essas imagens é por meio de grafos [1].

Nessas aplicações, há frequentemente duas imagens: uma imagem modelo e uma imagem teste. Deseja-se comparar essas duas imagens para verificar sua semelhança, de modo que seja possível classificar a imagem teste corretamente. Precisa-se, portanto, avaliar a semelhança entre as representações por grafos da imagem modelo e da imagem teste.

2.3.1 Correspondência exata entre grafos

O objetivo da correspondência exata entre grafos é determinar se dois grafos, ou partes desses dois grafos, sâo idênticas em termos de suas estruturas e atributos [13].

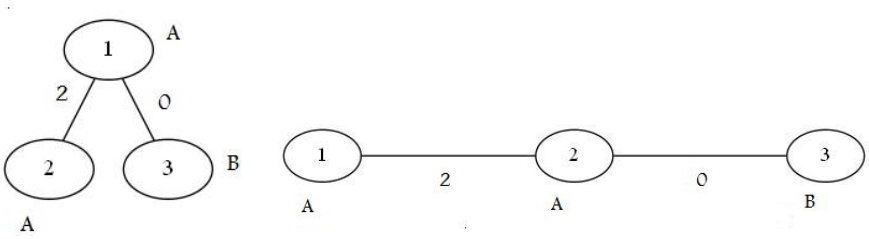
Definição 2.3 *Sejam $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ dois grafos rotulados. Diz-se que G_1 e G_2 são isomorfos se existe uma função bijetiva $f : V_1 \to V_2$ tal que*

- $\mu_1(u) = \mu_2(f(u))$, $\forall u \in V_1$;
- *para cada aresta $e_1 = \{u, v\} \in E_1$, existe uma aresta $e_2 = \{f(u), f(v)\} \in E_2$ tal que $\upsilon_1(e_1) = \upsilon_2(e_2)$; e*

- *para cada aresta* $e_2 = \{u, v\} \in E_2$, *existe uma aresta* $e_1 = \{f^{-1}(u), f^{-1}(v)\} \in E_1$ *tal que* $\upsilon_1(e_1) = \upsilon_2(e_2)$.

Portanto, no problema de isomorfismo de grafos rotulados é preciso verificar não somente se as adjacências são preservadas, como também se os rótulos dos vértices e arestas foram mantidos.

A Figura 2.3 mostra dois grafos rotulados G e H isomorfos entre si. A função $f : V_G \to V_H$ que define o isomorfismo é definida por $f(1) = 2$, $f(2) = 1$ e $f(3) = 3$, onde V_G e V_H denotam, respectivamente, os conjuntos de vértices de G e H.



(a) Grafo G (b) Grafo H

Figura 2.3: Isomorfismo de grafos

Relacionado ao problema de verificar se existe isomorfismo entre dois grafos, está o problema de isomorfismo de subgrafos. Dados dois grafos G_1 e G_2, o problema do isomorfismo de subgrafos consiste em verificar se existe um subgrafo de G_2 que seja isomorfo a G_1.

Definição 2.4 *Sejam* $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ *e* $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ *dois grafos rotulados. Uma função injetiva* $f : V_1 \to V_2$ *é chamada de isomorfismo de subgrafo de* G_1 *para* G_2 *se existe um subgrafo* $G \subseteq G_2$ *tal que* f *define um isomorfismo de grafo de* G_1 *para* G.

Em relação à complexidade desses dois problemas, o problema do isomorfismo de grafos pertence à classe NP [14]. Existem algoritmos polinomiais para o problema do isomorfismo para classes especiais de grafos (grafos planares e árvores, por exemplo), mas a NP-completude do caso geral ainda não foi provada.

O problema de isomorfismo de subgrafos é NP-completo (a demonstração é baseada numa redução do problema da clique [15]). A transformação polinomial de uma instância do problema de isomorfismo de subgrafos em uma instância do problema de isomorfismo de subgrafos rotulados é direta, bastando considerar que todos os vértices possuem o

mesmo rótulo na instância do problema rotulado. Assim, o problema do isomorfismo de subgrafos rotulados também é NP-completo.

Definição 2.5 *Sejam $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ dois grafos rotulados. Um grafo G é dito ser um subgrafo comum de G_1 e G_2 se existe um isomorfismo de subgrafo de G para G_1 e outro de G para G_2.*

Definição 2.6 *Um subgrafo comum G de dois grafos G_1 e G_2 é dito ser um máximo subgrafo comum de G_1 e G_2 ($MaxSub(G_1, G_2)$) se nenhum outro subgrafo comum de G_1 e G_2 possui mais vértices do que G.*

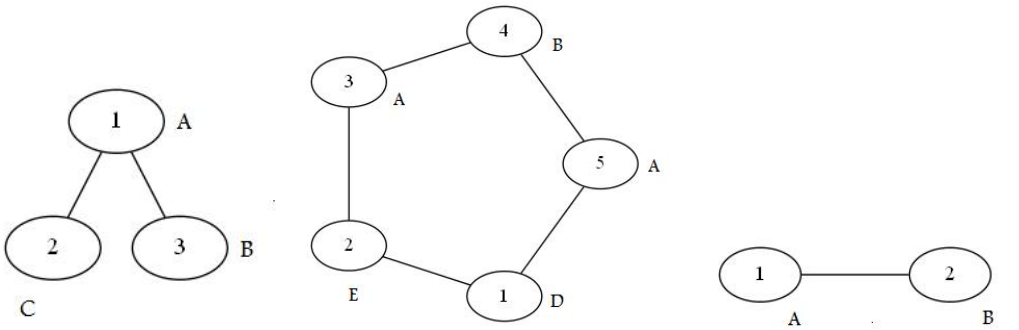
Verificar se existe um isomorfismo de subgrafos entre dois grafos G_1 e G_2 é verificar se existe um subgrafo de G_2 isomorfo a G_1. Encontrar um subgrafo comum de G_1 e G_2 corresponde a verificar se um subgrafo de G_1 é isomorfo a um subgrafo de G_2.

A versão de decisão do problema de determinar o máximo subgrafo comum entre dois grafos é um problema NP-completo [15]. Dentre as estratégias para computar o máximo subgrafo comum de dois grafos, duas abordagens são utilizadas com maior frequência. Uma é usar uma estratégia de *backtracking* para explorar o conjunto de soluções viáveis. A outra consiste em, a partir dos dois grafos originais, criar um grafo auxiliar e buscar a clique máxima desse grafo. A clique máxima encontrada é então transformada no máximo subgrafo comum. Comparações entre alguns desses métodos aparecem em [6, 10]. Três algoritmos exatos para o problema são apresentados em [10]: algoritmo de McGregor, algoritmo de Durand-Pasari e o algoritmo de Balas-Yu. Os resultados experimentais mostraram que nenhum dos três é definitivamente melhor do que os outros e que, dependendo da estrutura dos grafos, cada algoritmo pode ser consideravelmente mais eficiente do que os outros. A Figura 2.4 mostra dois grafos rotulados G_1 e G_2 e seu máximo subgrafo comum.

O conceito de mínimo supergrafo comum de dois grafos é análogo ao conceito de máximo subgrafo comum.

Definição 2.7 *Sejam $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ dois grafos rotulados. Um grafo G é dito ser um supergrafo comum de G_1 e G_2 se existe um isomorfismo de subgrafo de G_1 para G e outro de G_2 para G.*

Definição 2.8 *Um supergrafo comum G de dois grafos G_1 e G_2 é dito ser um mínimo supergrafo comum de G_1 e G_2 ($MinSup(G_1, G_2)$) se nenhum outro supergrafo comum de G_1 e G_2 possui menos vértices do que G.*



(a) Grafo G_1 (b) Grafo G_2 (c) Máximo subgrafo comum G

Figura 2.4: Em (c) está representado o máximo subgrafo comum G de G_1 e G_2. O isomorfismo de subgrafo de G para G_1 é dado pela função $f : V_G \to V_{G_1}$ tal que $f(1) = 1$ e $f(2) = 3$. O isomorfismo de subgrafo de G para G_2 é dado pela função $g : V_G \to V_{G_2}$ tal que $g(1) = 3$ e $g(2) = 4$.

2.3.2 Correspondência inexata entre grafos

Em aplicações do mundo real, quando um objeto ou imagem é representado por meio de um grafo, pode ocorrer de algum ruído ser introduzido na representação do objeto ou imagem (devido a algum erro no processo de aquisição da imagem, por exemplo).

Dessa forma, devido aos ruídos, um mesmo objeto pode ter representações diferentes. Em casos como este, o conceito de isomorfismo nos indicaria que os objetos são diferentes, apesar de os grafos possuírem grande semelhança. Os conceitos da correspondência exata entre grafos (isomorfismo, máximo subgrafo comum, mínimo supergrafo comum) não são adequados para esse tipo de situação. É preciso, portanto, introduzir alguma tolerância a erros no processo de correspondência de grafos.

A correspondência inexata consiste em associar os vértices e arestas de dois grafos, e utilizar uma função objetivo para medir a qualidade das correspondências obtidas. A distância de edição de dois grafos é uma das medidas mais utilizadas para medir a similaridade entre eles no caso da correspondência inexata de grafos. Como exemplo de problema de correspondência inexata, Boeres [2] e Boeres *et al.* [3] propõem heurísticas para o problema de reconhecer cenas em imagens complexas.

2.3.3 Distância baseada em edições

A distância baseada na edição é um dos principais conceitos em correspondência inexata entre grafos. Essa distância é uma medida da dissimilaridade entre dois grafos por meio da quantidade mínima de distorção (edição) exigida para transformar um grafo no outro

[13].

As operações de edição utilizadas para realizar a transformação são: inserção ou remoção de vértices e inserção ou remoção de arestas. Assim, dado um par de grafos G_1 e G_2, existe uma sequência de operações (ou caminho) de edição $p(G_1, G_2) = (o_1, o_2, ..., o_m)$ que transforma G_1 em um grafo isomorfo a G_2, onde cada o_i é uma operação de edição, $i = 1, \ldots, m$.

Dado um par de grafos G_1 e G_2, podem existir vários caminhos de edição que transformam G_1 em G_2. O conjunto formado por esses caminhos será denotado por $P(G_1, G_2)$. Para que possa ser feita uma avaliação quantitativa desses caminhos, é preciso associar custos às operações de edição, isto é, é preciso definir uma função de custo para a inserção e remoção de vértices e arestas. No Capítulo 3 será definida a função de custo que será utilizada.

A distância de edição entre dois grafos é definida como o custo do caminho de edição mínimo. Formalmente tem-se:

Definição 2.9 *Dados dois grafos rotulados $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$, a distância de edição entre G_1 e G_2 é dada pela soma dos custos de edição das operações do caminho mais curto de G_1 a G_2.*

Um exemplo do cálculo da distância de edição entre dois grafos é dado na Seção 3.2, página 15. Devido à complexidade do problema, algoritmos exatos para encontrar a distância de edição só podem ser usados para grafos pequenos. Um exemplo de algoritmo aproximado para computar essa distância pode ser encontrado em [21].

2.4 Grafo mediano

Dado um conjunto de grafos, seu grafo mediano é definido como aquele que possui a menor soma das distâncias aos grafos do conjunto. Na área de reconhecimento de padrões, o grafo mediano é muitas vezes usado como o representante de um conjunto de grafos.

Definição 2.10 *Seja U o conjunto de todos os grafos rotulados que podem ser construídos usando os atributos de um alfabeto L. Dado um conjunto de grafos $S = \{G_1, G_2, ..., G_n\} \subseteq U$, seu grafo mediano $\hat{G}$ e seu grafo mediano generalizado $\bar{G}$ são definidos como:*

$$\hat{G} = \text{argmin}_{G' \in S} \sum_{G \in S} d(G', G)$$

$$\bar{G} = \text{argmin}_{G' \in U} \sum_{G \in S} d(G', G).$$

O grafo mediano de S é um grafo $\hat{G} \in S$ que possui menor soma das distâncias para os grafos de S. O grafo mediano generalizado de S é um grafo $\bar{G} \in U$ que possui menor soma das distâncias para os grafos de S. Como $S \subseteq U$, o conjunto de soluções viáveis do problema do grafo mediano generalizado contém o conjunto de soluções do problema do grafo mediano. Assim, um grafo mediano generalizado é, em geral, um melhor representante do conjunto S do que um grafo mediano do conjunto.

2.4.1 Propriedades do grafo mediano

Nesta seção serão descritas propriedades do grafo mediano generalizado de um conjunto. Especificamente, serão mostrados limites para o número de vértices do grafo mediano e para a soma das distâncias (SOD) do grafo mediano.

Proposição 2.1 *Seja $S = \{G_1, G_2, ..., G_n\}$ um conjunto de grafos rotulados e seu grafo mediano generalizado $\bar{G}$, então:*

$$|\bar{G}| \leq \sum_{i=1}^{n} |G_i|$$

O grafo vazio $G_e = (V_e, E_e, \mu_e, \upsilon_e)$ é tal que $V_e = E_e = \emptyset$. O grafo união de um conjunto S de grafos rotulados é o grafo $G_u = (V_u, E_u, \mu_u, \upsilon_u)$ tal que $V_u = \bigcup_{i=1}^{n} V_i$, $E_u = \bigcup_{i=1}^{n} E_i$, e os rótulos dos vértices e arestas de cada grafo G_i são mantidos.

Em relação à soma das distâncias do grafo mediano tem-se a seguinte proposição [17]:

Proposição 2.2 *Sejam $S = \{G_1, G_2, ..., G_n\}$ um conjunto de grafos rotulados, $\bar{G}$ seu grafo mediano generalizado, G_e um grafo vazio e G_u o grafo união dos grafos de S. Então, a soma das distâncias do grafo mediano generalizado satisfaz à relação abaixo:*

$$SOD(\bar{G}) \leq \min\{SOD(G_e), SOD(G_u)\}.$$

Dado um conjunto M, uma função $d : M \times M \rightarrow \mathbb{R}$ é dita uma métrica quando possui as seguintes propriedades: $d(x, y) \geq 0$, $\forall x, y \in M$; $d(x, y) = d(y, x)$, $\forall x, y \in M$; $d(x, z) \leq d(x, y) + d(y, z)$, $\forall x, y, z \in M$; e $d(x, y) = 0 \Leftrightarrow x = y$.

Bunke, Jiang e Munger [17] ressaltam que o limite superior apresentado na Proposição 2.2 pode ser fraco e de pouca utilidade prática e encontram um limite inferior mais forte quando a distância utilizada é uma métrica.

2.4.2 Complexidade computacional e algoritmos

Quando se deseja encontrar o grafo mediano $\hat{G}$ de um conjunto $S = \{G_1, ..., G_n\}$ de grafos rotulados, é preciso computar as distância de cada grafo G_i para cada grafo de S, para $i = 1, \ldots, n$. É preciso, portanto, calcular $n(n-1)/2$ distâncias entre pares de grafos para obter aquele que possui a menor soma das distâncias. Assim, o grafo mediano pode ser computado em $O(n^2\delta)$, onde δ é a complexidade do cálculo da função de distância. Como a versão de decisão do problema de encontrar a distância de edição entre dois grafos é um problema NP-completo [25], a computação do grafo mediano ($\hat{G}$) de um conjunto de grafos segundo esse princípio só pode ser feita em tempo exponencial, se P $\neq$ NP.

O caso do grafo mediano generalizado é ainda mais complexo, uma vez que a computação é exponencial tanto na quantidade de grafos de entrada quanto no tamanho dos grafos. Como será visto mais adiante, o aumento no número de grafos aumenta o número de soluções viáveis e o aumento no tamanho dos grafos dificulta a computação das distâncias. Mesmo no caso especial de cadeias (que são uma classe particular de grafos), o tempo requerido na computação do grafo mediano é exponencial na quantidade de cadeias de entrada [11].

A maior parte dos algoritmos existentes na literatura tem o objetivo de encontrar o grafo mediano generalizado. A computação do grafo mediano $\hat{G}$ pode ser feita calculando-se todas as distâncias entre todos os pares de grafos que formam o conjunto. Entretanto, como o cálculo da distância de edição é uma tarefa difícil, é conveniente que seja adotada uma estratégia que possa evitar ou simplificar o cálculo de todas as $n(n-1)/2$ distâncias. O algoritmo *Fast Search Median Algorithm* [18] utiliza uma estratégia que evita a avaliação de todos os candidatos do conjunto.

Em relação ao grafo mediano generalizado, Ferrer [13] descreve os algoritmos existentes para o problema: um algoritmo exato (*MultiMatch*) e dois algoritmos aproximados. Além disso, propõe dois novos algoritmos exatos para o problema, um deles usando pro-

priedades para diminuir o espaço de busca e outro utilizando a técnica de transformar grafos em vetores e buscar o vetor mediano nesse espaço para posteriormente transformar esse vetor no grafo mediano. Ambos algoritmos, entretanto, lidam apenas com grafos de tamanho relativamente pequeno, com não mais de 20 vértices.

Capítulo 3

Função de custo e algoritmos

3.1 Introdução

No capítulo anterior foi visto que é necessário calcular a distância entre cada candidato a grafo mediano generalizado e todos os grafos do conjunto $S = \{G_1, \ldots, G_n\}$ para se obter aquele que apresenta a menor soma das distâncias. Neste capítulo é definida a função de custo das operações de edição e são apresentados os algoritmos implementados.

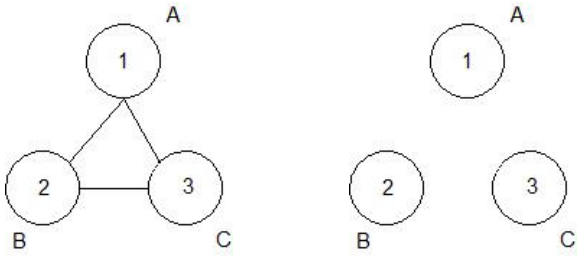
O capítulo está organizado da seguinte forma. Na Seção 3.2 são definidos os custos associados às operações de edição. Na Seção 3.3 são apresentadas novas propriedades do grafo mediano. Na Seção 3.4 é apresentado o espaço de busca no qual os algoritmos propostos procurarão a melhor solução para o problema do grafo mediano generalizado. Na Seção 3.5 são apresentados os algoritmos utilizados para computar o máximo subgrafo comum e o mínimo supergrafo comum. Na Seção 3.6 é apresentada a técnica utilizada para acelerar a avaliação de um candidato a grafo mediano generalizado. Na Seção 3.7 são apresentados os algoritmos propostos para encontrar o grafo mediano generalizado.

3.2 Função de custo

Nesta seção será apresentada a função de custo que será usada nesse trabalho, utilizada pela primeira vez em [4] e também por Ferrer [13].

As operações de edição consistem em inserir vértices e arestas ou remover vértices e arestas. A inserção de arestas ocorre somente quando um vértice é inserido, e a remoção de arestas ocorre somente quando um vértice é removido (as arestas adjacentes a esse vértice removido são removidas). Assume-se que os custos de inserir ou remover vértices

são unitários, enquanto os custos de inserir ou remover arestas são nulos. A Figura 3.1 ilustra o cálculo da distância de edição entre dois grafos.



(a) Grafo G (b) Grafo H

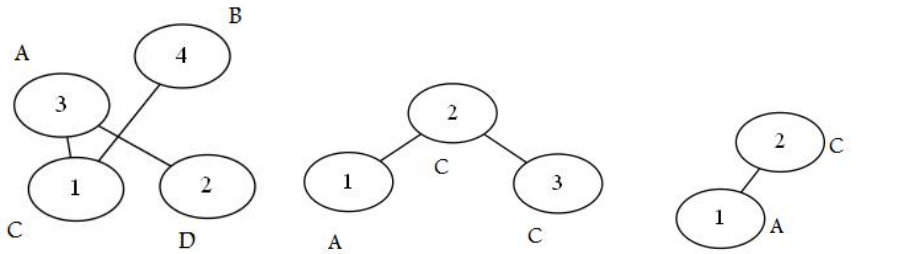
Figura 3.1: A seguinte sequência de operações de edição transforma o grafo G em um grafo isomorfo ao grafo H: retira-se o vértice 1 de G (automaticamente suas arestas são removidas), e insere-se o vértice 1 novamente (sem inserir nenhuma aresta adjacente a ele). Retira-se o vértice 2 e insere-se o novamente. Remover e inserir o vértice 1 tem custo 2, remover e inserir o vértice 2 tem custo 2. Assim, $d(G, H) = 2 + 2 = 4$

Foi demonstrado [4] que, utilizando essa função de custo, o cálculo da distância entre dois grafos G_1 e G_2 pode ser feito utilizando a seguinte fórmula:

$$d(G_1, G_2) = |G_1| + |G_2| - 2 \cdot |MaxSub(G_1, G_2)|$$

Conforme apresentado no Capítulo 2, $MaxSub(G_1, G_2)$ indica o máximo subgrafo comum de G_1 e G_2. Essa fórmula para a distância ilustra a ideia de que, quanto mais similares forem os grafos (quanto maior for o máximo subgrafo comum entre eles), menor será a distância entre eles [13]. Além das consequências teóricas desse resultado, existem também implicações práticas. Por exemplo, uma consequência imediata desse resultado é que qualquer algoritmo que compute a distância de edição entre dois grafos pode ser usado para a computação de seu máximo subgrafo comum [13].

Como exemplo, considerem-se os grafos G e H na Figura 3.2. O grafo G possui quatro vértices e o grafo H possui três vértices. O máximo subgrafo comum entre G e H é o grafo composto por um vértice com rótulo A conectado a outro vértice com rótulo C. Portanto, $|MaxSub(G, H)| = 2$. Logo, a distância entre G e H é dada por $d(G, H) = 4 + 3 - 2 \times 2 = 3$. O mesmo resultado pode ser obtido utilizando as operações de edição. O caminho de custo mínimo para transformar o grafo G no grafo H consiste em remover os vértices 2 e 4 pertencentes a V_G, assim como as arestas incidentes nos mesmos, adicionar um vértice com rótulo C, e adicionar uma aresta conectando esse novo vértice ao vértice 1 de V_G. O custo total é 3, mesmo resultado obtido utilizando a fórmula da distância.



(a) Grafo G (b) Grafo H (c) $MaxSub(G, H)$

Figura 3.2: O isomorfismo de subgrafo de $MaxSub(G, H)$ para G é definido pela função $f : V_{MaxSub} \rightarrow V_G$ tal que $f(1) = 3$ e $f(2) = 1$. O isomorfismo de $MaxSub(G, H)$ para H é definido pela função $g : V_{MaxSub} \rightarrow V_H$ tal que $g(1) = 1$ e $g(2) = 2$.

3.3 Propriedades

No Capítulo 2 foram apresentadas propriedades que determinavam limites para o tamanho e SOD do grafo mediano generalizado. Essas propriedades podem ser utilizadas para reduzir o espaço de busca. Entretanto, esses limites algumas vezes são fracos, sendo de pouca utilidade prática. Nesta seção serão apresentadas novas propriedades do grafo mediano, descritas pela primeira vez por Ferrer [13] e que fornecem limites de maior utilidade prática.

As definições a seguir generalizam o conceito de máximo subgrafo comum e mínimo supergrafo comum.

Definição 3.1 *Seja $S = \{G_1, ..., G_n\}$ um conjunto de grafos. $MaxSub(S)$ é um máximo subgrafo comum de S se for um subgrafo comum de $G_1, ..., G_n$ e nenhum outro subgrafo comum de $G_1, ..., G_n$ possui mais vértices do que $MaxSub(S)$.*

Definição 3.2 *Seja $S = \{G_1, ..., G_n\}$ um conjunto de grafos. $MinSup(S)$ é um mínimo supergrafo comum de S se for um supergrafo comum de $G_1, ..., G_n$ e nenhum outro supergrafo comum de $G_1, ..., G_n$ possui menos vértices do que $MinSup(S)$.*

Os limites apresentados no Capítulo 2 são válidos para qualquer função de distância. Utilizando a função de custo definida neste capítulo, Ferrer [13] demonstrou que esses limitantes podem ser melhorados:

Proposição 3.1 *Seja $S = \{G_1, G_2, ..., G_n\}$ um conjunto de grafos e $\bar{G}$ um grafo mediano generalizado de S. Considerando-se a função de distância definida na Seção 3.2, então:*

$$|MaxSub(S)| \leq |\bar{G}| \leq |MinSup(S)|$$

Esta proposição é importante, uma vez que ela naturalmente sugere um ponto de partida para a busca do grafo mediano generalizado no espaço de soluções. No próximo capítulo esta proposição servirá de base para a limitação do espaço de busca dos algoritmos propostos.

Um limitante superior para a SOD do grafo mediano generalizado de um conjunto de grafos é dado por $\min\{SOD(G_e), SOD(G_u)\}$. A proposição abaixo mostra que é possível melhorá-lo:

Proposição 3.2 *Seja $S = \{G_1, G_2, ..., G_n\}$ um conjunto de grafos e $\bar{G}$ um grafo mediano generalizado de S. Considerando a função de distância definida na Seção 3.2, então:*

$$SOD(\bar{G}) \leq SOD(MaxSub(S)) \leq \min\{SOD(G_e), SOD(G_u)\}$$

A proposição seguinte é uma contribuição dessa dissertação e pode ser utilizada para aumentar a eficiência de algoritmos destinados ao problema:

Proposição 3.3 *Seja $S = \{G_1, G_2, ..., G_n\}$ um conjunto de grafos tal que $|MaxSub(G_i, G_j)| = a \in \mathbb{Z}$, $\forall i, j = 1, \ldots, n, i \neq j$ e $n \geq 3$. Se $\hat{G}$ é o grafo mediano do conjunto S, então $|\hat{G}| = min\{|G_i|; i = 1, \ldots, n\}$.*

Prova: Seja $\hat{G} = G_k \in S$ o grafo mediano do conjunto S. Da definição de grafo mediano do conjunto tem-se:

$$d(G_1, G_k) + \ldots + d(G_n, G_k) \leq d(G_1, G_t) + \ldots + d(G_n, G_t)$$

para $t = 1, \ldots, n$. Daí, tem-se que:

$$|G_1| + |G_k| - 2 \cdot |MaxSub(G_1, G_k)| + \ldots + |G_n| + |G_k| - 2 \cdot |MaxSub(G_n, G_k)| \leq$$
$$|G_1| + |G_t| - 2 \cdot |MaxSub(G_1, G_t)| + \ldots + |G_n| + |G_t| - 2 \cdot |MaxSub(G_n, G_t)|$$

Como $|MaxSub(G_i, G_j)| = a$ para todo $i, j = 1, \ldots, n$, $i \neq j$, $|MaxSub(G_k, G_k)| = |G_k|$ e $|MaxSub(G_t, G_t)| = |G_t|$ segue que:

$$|G_1| + \ldots + |G_n| + n|G_k| - 2(n-1)a - 2 \cdot |G_k| \leq |G_1| + \ldots + |G_n| + n|G_t| - 2(n-1)a - 2 \cdot |G_t|$$

Logo, $(n-2) \cdot |G_k| \leq (n-2) \cdot |G_t|$. Portanto, $|G_k| \leq |G_t|$, $t = 1, \ldots, n$. Ou seja, $|\hat{G}| = |G_k| = min\{|G_i|; i = 1, \ldots, n\}$.

A proposição acima pode ser usada para encontrar os candidatos a grafo mediano do conjunto ($\hat{G}$) quando o máximo subgrafo comum de qualquer par de grafos do conjunto S possui o mesmo tamanho.

3.4 Espaço de busca

Conforme visto na Proposição 3.1, o espaço de busca para encontrar o grafo mediano de um conjunto S de grafos consiste de todos os grafos cujo número de vértices está entre $|MaxSub(S)|$ e $|MinSup(S)|$.

Portanto, uma vez que o grafo mediano generalizado satisfaz $|MaxSub(S)| \leq |\bar{G}| \leq |MinSup(S)|$, pode-se considerar o grafo $MinSup(S)$ e seus subgrafos como candidatos naturais a serem um grafo mediano generalizado do conjunto. Os grafos induzíveis em $MinSup(S)$ por algum subconjunto de vértices do mínimo supergrafo comum serão referenciados apenas pelo termo "subgrafos induzíveis" no restante desse texto.

Sejam S o conjunto de grafos para o qual se quer encontrar o grafo mediano e G' o subgrafo induzido no mínimo supergrafo comum $MinSup(S)$ por um subconjunto V' dos seus vértices. Seja ainda G'' um subgrafo de G' com o mesmo conjunto de vértices V'. Em [13] é demonstrado que $SOD(G') \leq SOD(G'')$. Assim, somente os subgrafos induzíveis de $MinSup(S)$ devem ser considerados na busca pelo grafo mediano generalizado.

O espaço de busca do problema é, portanto, formado pelo conjunto de todos os subgrafos induzíveis de $MinSup(S)$ cujo número de vértices está entre $|MaxSub(S)|$ e $|MinSup(S)|$. Portanto, um algoritmo de busca exaustivo consistiria em verificar, para cada subgrafo induzível $G \subseteq MinSup(S)$, satisfazendo $|MaxSub(S)| \leq |G| \leq |MinSup(S)|$, o valor de $SOD(G) = d(G_1, G) + d(G_2, G) + \ldots + d(G_n, G)$ e selecionar como grafo mediano aquele que apresenta o menor valor para a SOD.

Ferrer [13] propôs um algoritmo exato que explora o espaço de busca apresentado acima, utilizando técnicas de "look-ahead" para evitar a avaliação de grafos que não podem melhorar a solução já obtida. Este algoritmo exato foi aplicado em instâncias com $|MinSup(S)| \leq 14$, retiradas do mesmo conjunto de instâncias utilizadas neste trabalho.

3.5 Algoritmos auxiliares

O cálculo da distância entre dois grafos G_1 e G_2 usa o número de vértices $|MaxSub(G_1, G_2)|$ do seu máximo subgrafo comum. Da mesma forma, o espaço de busca para o grafo mediano generalizado consiste nos subgrafos induzíveis do mínimo supergrafo comum $MinSup(S)$. Precisa-se portanto, de algoritmos para encontrar esses grafos. Os algoritmos utilizados nesse trabalho para resolver esses problemas são o de Durand-Pasari [10] (para calcular o máximo subgrafo comum entre dois grafos) e o algoritmo para encontrar o mínimo supergrafo comum entre dois ou mais grafos [7] proposto por Bunke.

3.5.1 Algoritmo de Durand-Pasari

Neste trabalho é utilizada como medida de similaridade de dois grafos a distância de edição entre eles. Utilizando-se a função de custo apresentada anteriormente, a distância entre dois grafos G_1 e G_2 é dada por:

$$d(G_1, G_2) = |G_1| + |G_2| - 2 \cdot |MaxSub(G_1, G_2)|$$

.

Para o cálculo de $d(G_1, G_2)$ é necessário então computar o máximo subgrafo comum entre G_1 e G_2. Em [10] é feita uma revisão dos principais algoritmos existentes para esse problema.

Durand e Pasari [12] propuseram um algoritmo exato que utiliza a redução do problema do máximo subgrafo comum entre dois grafos ao problema de encontrar a clique máxima em um grafo.

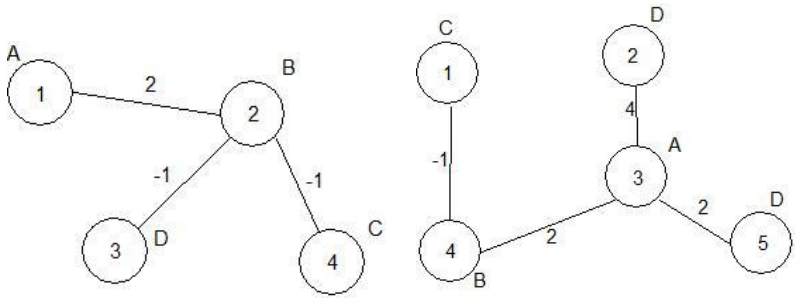
O algoritmo tem como entrada dois grafos rotulados G_1 e G_2. O primeiro passo desse algoritmo é a construção do *grafo auxiliar*, cujos vértices correspondem a pares de vértices dos grafos G_1 e G_2 que possuem o mesmo atributo. Ou seja, os vértices do grafo auxiliar são pares ordenados (n_1, n_2) onde $n_1 \in V_1$, $n_2 \in V_2$ e $\mu_1(n_1) = \mu_2(n_2)$.

As arestas do grafo auxiliar representam a compatibilidade entre pares de vértices. Isto é, um vértice (n_1, n_2) do grafo auxiliar estará conectado ao vértice (m_1, m_2) se e somente a aresta $\{n_1, m_1\}$ do primeiro grafo e a aresta $\{n_2, m_2\}$ do segundo grafo tiverem o mesmo atributo.

Cada clique no grafo auxiliar pode ser transformada em um subgrafo comum de G_1 e G_2 e vice-versa. Portanto, o máximo subgrafo comum pode ser obtido encontrando-se a

clique máxima do grafo auxiliar.

O exemplo abaixo ilustra o funcionamento do algoritmo. Sejam os grafos G_1 e G_2 representados na Figura 3.3, que levam ao grafo auxiliar a G_1 e G_2 apresentado na Figura 3.4.



(a) G_1 (b) G_2

Figura 3.3: Grafos G_1 e G_2

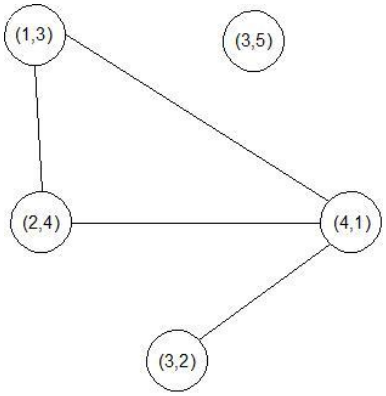


Figura 3.4: Grafo auxiliar a G_1 e G_2

Os vértices desse grafo auxiliar são pares (n_1, n_2) onde $n_1 \in V_1$, $n_2 \in V_2$ e $\mu_1(n_1) = \mu_2(n_2)$. Assim, o par (1,3) está presente no grafo auxiliar pois o vértice 1 pertence a V_1, o vértice 3 pertence a V_2 e $\mu_1(1) = \mu_2(3) = A$. Existe uma aresta entre os pares (1,3) e (2,4) porque as arestas $\{1, 2\} \in E_1$ e $\{3, 4\} \in E_2$ têm o mesmo atributo. Da mesma forma, existe uma aresta entre os pares (3,2) e (4,1), uma vez que $\{3, 4\} \notin E_1$ e $\{2, 1\} \notin E_2$ (para o algoritmo de Durand-Pasari, nos casos em que não existe uma aresta, considera-se que ambas arestas possuem um atributo com valor nulo).

Cada clique nesse grafo auxiliar corresponde a um subgrafo comum de G_1 e G_2. A clique máxima desse grafo auxiliar é formada pelos vértices correspondentes aos pares (1,3), (2,4) e (4,1). Para converter essa clique no máximo subgrafo comum de G_1 e G_2 basta separar os pares em suas coordenadas. Selecionando as primeiras coordenadas de cada par, tem-se que o máximo subgrafo comum entre G_1 e G_2 está representado em

G_1 pelos vértices $\{1, 2, 4\} \subset V_1$. Da mesma forma, esse máximo subgrafo comum está representado em G_2 pelos vértices $\{3, 4, 1\} \subset V_2$.

3.5.2 Algoritmo de Bunke para mínimo supergrafo comum de dois grafos

Dado um conjunto $S = \{G_1, \ldots, G_n\}$ de grafos, o espaço de busca do problema consiste dos subgrafos induzíveis no mínimo supergrafo comum $MinSup(S)$ de S. É necessário, portanto, um algoritmo para encontrar o mínimo supergrafo comum de um conjunto de grafos. Desse ponto em diante nessa dissertação, não será feita distinção entre grafos isomorfos, isto é, dados dois grafos isomorfos G_1 e G_2 assume-se que $G_1 = G_2$. Da mesma forma, se existe um isomorfismo de subgrafo de G_1 para G_2, assume-se que G_1 é um subgrafo de G_2, isto é, $G_1 \subseteq G_2$.

O algoritmo para encontrar o mínimo supergrafo comum de um conjunto de grafos se baseia na aplicação repetitiva do algoritmo exato de Bunke [8] para determinar o mínimo supergrafo comum de dois grafos (apresentado mais adiante), o que se constitui em um algoritmo aproximado para o problema geral com diversos grafos. Pode garantir-se apenas que o algoritmo encontra um supergrafo do mínimo supergrafo comum. O grafo encontrado por este algoritmo será tratado como sendo o mínimo supergrafo comum, embora não o seja necessariamente, isto é, o algoritmo eventualmente pode encontrar uma solução subótima. Antes de apresentá-lo, é preciso conhecer as seguintes definições:

Definição 3.3 *Sejam $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ dois grafos rotulados com $G_1 \subseteq G_2$. A diferença entre eles é o grafo $G_2 - G_1 = (V, E, \mu, \upsilon)$ onde $V = V_2 - V_1$, $E = E_2 \cap (V \times V)$, $\mu(v) = \mu_2(v)$ para qualquer $v \in V$, e $\upsilon(e) = \upsilon_2(e)$ para qualquer $e \in E$.*

Definição 3.4 *Sejam $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ dois grafos rotulados com $G_1 \subseteq G_2$. A imersão, $emb(G_1, G_2)$, de G_1 em G_2, é o conjunto de arestas que ligam vértices de G_1 a vértices de $G_2 - G_1$, isto é, $emb(G_1, G_2) = E_2 \cap [V_1 \times (V_2 - V_1)]$, onde $\upsilon(e) = \upsilon_2(e)$, para qualquer $e \in emb(G_1, G_2)$*

No Capítulo 2 foi utilizada uma definição de união de um conjunto de grafos. A definição a seguir define a *união de dois grafos G_1 e G_2 incluindo um conjunto de arestas* $\overline{E}$, que será simbolizada por $G_1 \bigcup_{\overline{E}} G_2$.

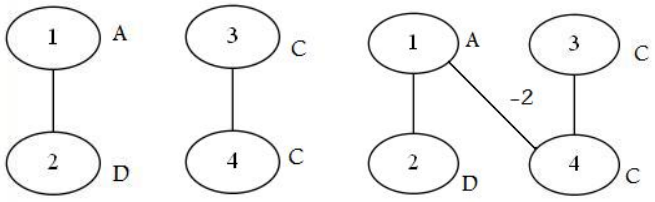
Definição 3.5 *Sejam $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ e $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ dois grafos rotulados com $V_1 \cap V_2 = \emptyset$. Seja $\overline{E} \subseteq V_1 \times V_2$ um conjunto de arestas com uma função $\overline{\upsilon} : \overline{E} \rightarrow L$*

que atribui rótulos às arestas. A união de G_1 *e* G_2 *incluindo* $\overline{E}$, $G_1 \bigcup_{\overline{E}} G_2$ *é um grafo* $G = (V, E, \mu, \upsilon)$ *onde* $V = V_1 \cup V_2$, $E = E_1 \cup E_2 \cup \overline{E}$ *e*

- $\mu(v) = \begin{cases} \mu_1(v), \ se\ v \in V_1 \\ \mu_2(v), \ se\ v \in V_2 \end{cases}$

- $\upsilon(e) = \begin{cases} \upsilon_1(e), \text{ se } e \in E_1 \\ \upsilon_2(v), \text{ se } e \in E_2 \\ \overline{\upsilon}(e), \text{ se } e \in \overline{E} \end{cases}$

A Figura 3.5 mostra dois grafos rotulados G_1 e G_2, o conjunto de arestas $\overline{E} = \{\{1, 4\}\}$ com $\beta(\{1, 4\}) = -2$ e sua união $G_1 \bigcup_{\overline{E}} G_2$.



(a) G_1 (b) G_2 (c) $G_1 \bigcup_{\overline{E}} G_2$

Figura 3.5: Grafos G_1 e G_2 e sua união

Bunke [8] mostrou que o mínimo supergrafo comum de dois grafos pode ser obtido da seguinte forma:

Proposição 3.4 *Sejam* $G_1 = (V_1, E_1, \mu_1, \upsilon_1)$ *e* $G_2 = (V_2, E_2, \mu_2, \upsilon_2)$ *dois grafos rotulados. Então:*

$$MinSup(G_1, G_2) = MaxSub(G_1, G_2) \cup_{A_1} (G_1 - MaxSub(G_1, G_2)) \cup_{A_2} (G_2 - MaxSub(G_1, G_2)),$$

onde $A_1 = emb(MaxSub(G_1, G_2), G_1)$ *e* $A_2 = emb(MaxSub(G_1, G_2), G_2)$.

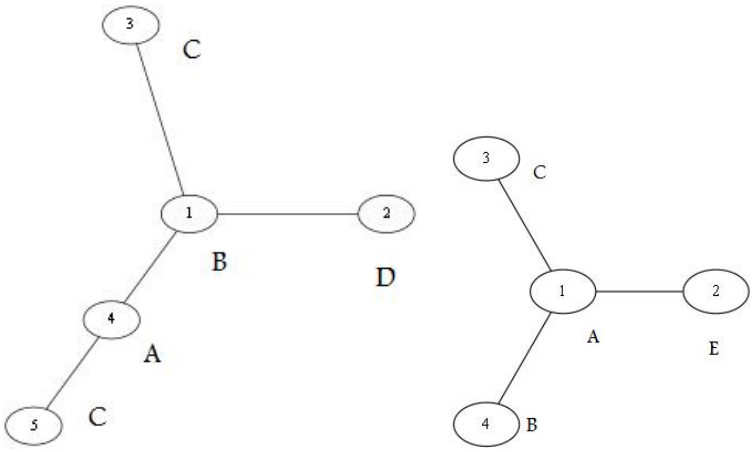
Para se obter o mínimo supergrafo comum de um conjunto S formado por $m \geq 2$ grafos, $MinSup(S)$, aplica-se sucessivamente a Proposição 3.4, como proposto por Bunke [7].

3.6 Cálculo acelerado das distâncias

O cálculo da distância de edição $d(G, H) = |G| + |H| - 2 \cdot |MaxSub(G, H)|$ entre dois grafos G e H é custoso, uma vez que o problema de computar o máximo subgrafo comum de dois

grafos é NP-difícil. Portanto, na busca pelo grafo mediano generalizado de um conjunto é conveniente evitar a computação do máximo subgrafo comum sempre que possível. Nesta seção é apresentada uma estratégia que permite evitar esse cálculo em certas situações, que posteriormente será usada para aumentar a eficiência dos algoritmos gulosos.

Considere os dois grafos rotulados G e H da Figura 3.6. O máximo subgrafo comum entre G e H é o grafo formado pelos vértices de rótulos A, B e C (correspondentes aos vértices 4, 1 e 5 no grafo G e aos vértices 1, 4 e 3 no grafo H). A distância entre esses grafos é, portanto:



(a) Grafo G (b) Grafo H

Figura 3.6: Cálculo de distância

$$d(G,H) = |G| + |H| - 2 \cdot |MaxSub(G,H)| = 5 + 4 - 2 \times 3 = 3$$

Seja agora o grafo G', obtido retirando-se o vértice 2 do grafo G. Esse vértice não faz parte do máximo subgrafo comum entre G e H, logo

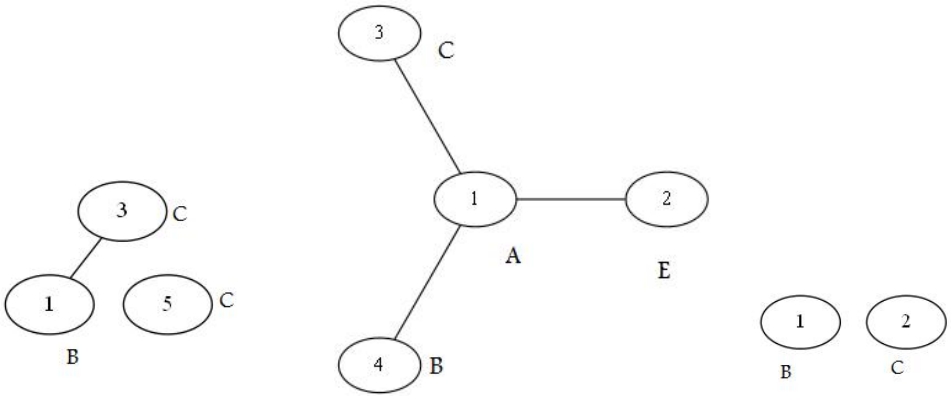
$d(G',H) = |G| - 1 + |H| - 2 \cdot |MaxSub(G,H)| = |G| + |H| - 2 \cdot |MaxSub(G,H)| - 1$, isto é, $d(G',H) = d(G,H) - 1 = 2$.

Neste caso, pode-se notar que a distância entre G' e H foi obtida a partir da distância entre G e H, sem que fosse necessário computar $MaxSub(G',H)$.

Seja agora o grafo G'' obtido retirando-se o vértice 4 de G'. Nesse caso, esse vértice faz parte do máximo subgrafo comum entre G' e H. Tem-se que $|G''| = 3$, $|H| = 4$ e $|MaxSub(G'',H)| = 2$. Portanto,

$$d(G'',H) = |G'| - 1 + |H| - 2 \cdot (|MaxSub(G',H)| - 1) = 3.$$

Ou seja, $d(G'', H) = d(G', H) + 1$.

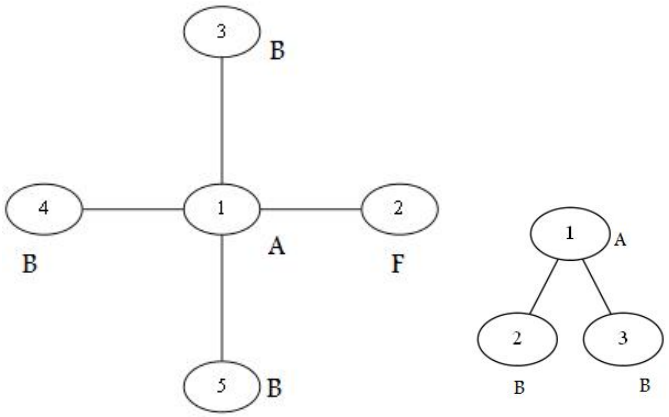


(a) Grafo G'' (b) Grafo H (c) $MaxSub(G'', H)$

Figura 3.7: $V_{G''} = V_{G'} - \{4\}$

Nem sempre retirar um vértice que faz parte do máximo subgrafo comum entre dois grafos irá resultar em um aumento no valor da distância. Por exemplo, considerem-se os grafos G e H da Figura 3.8. Se for retirado de G um vértice de rótulo B (vértice 3, por exemplo), obter-se-á um novo grafo G' tal que $|G'| = |G| - 1$ e $|MaxSub(G', H)| = |MaxSub(G, H)|$, uma vez que o máximo subgrafo comum entre G e H (que neste caso é o próprio grafo H) pode ser recomposto utilizando qualquer um dos outros dois vértices que possuem rótulo B. Tem-se então que:

$$\begin{aligned} d(G', H) &= |G'| + |H| - 2 \cdot |MaxSub(G', H)| = \\ &= |G| - 1 + |H| - 2 \cdot |MaxSub(G, H)| = d(G, H) - 1 \end{aligned}$$



(a) Grafo G (b) Grafo H

Figura 3.8: Cálculo de distância

Em resumo, se a distância $d(G, H)$ entre dois grafos G e H é conhecida e se G' é o grafo obtido pela eliminação de um vértice de G, é possível concluir que apenas dois casos

são possíveis:

- Se o vértice removido não faz parte do máximo subgrafo comum entre G e H, então: $d(G', H) = d(G, H) - 1$.
- Se o vértice removido faz parte do máximo subgrafo comum entre G e H então: $d(G', H) = d(G, H) \pm 1$.

A propriedade acima é contribuição deste trabalho e será utilizada nos algoritmos gulosos de forma a acelerar a computação da SOD na busca do grafo mediano generalizado. Adicionalmente, essa propriedade servirá de base para definir um critério guloso nos algoritmos propostos.

3.7 Algoritmos propostos

Nesta seção serão apresentados os algoritmos propostos para encontrar grafos medianos generalizados aproximados. Os dois primeiros utilizam escolhas gulosas e possuem estruturas similares. O terceiro se baseia no princípio do algoritmo A*.

3.7.1 Algoritmo guloso com tabela estática (A1)

Conforme foi visto na Seção 3.4, os grafos candidatos a grafo mediano generalizado de um conjunto S são os subgrafos induzíveis do mínimo supergrafo comum de S. O princípio deste algoritmo consiste em retirar sucessivamente vértices do mínimo supergrafo comum de forma a obter um subgrafo com a menor SOD possível. Para determinar quais vértices retirar será retomada a idéia apresentada na Seção 3.6.

Dado um conjunto $S = \{G_1, \ldots, G_n\}$ de grafos, seja $MinSup(S)$ seu mínimo supergrafo comum. Como $MinSup(S)$ é supergrafo de cada G_i, tem-se que $MaxSub(MinSup(S), G_i) = G_i$, $1 \leq i \leq n$. Considere agora a SOD inicial do mínimo supergrafo comum $MinSup(S)$:

$$SOD(MinSup(S)) = d(MinSup(S), G_1) + \ldots + d(MinSup(S), G_n)$$

Considere o vértice $k \in V_{MinSup(S)}$. Esse vértice faz parte do máximo subgrafo comum de alguns grafos G_i's com $MinSup(S)$, $i = 1, \ldots, n$ e não faz parte de outros. Suponha que k faça parte de apenas um máximo subgrafo comum, por exemplo $k \in MaxSub(MinSup(S), G_r)$. Considere o grafo $MinSup(S)'$, obtido retirando

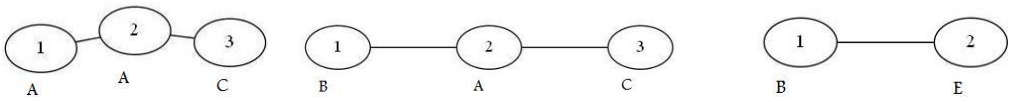
o vértice k do grafo $MinSup(S)$. Como visto na seção anterior, é razoável estimar que $d(MinSup(S)', G_r) = d(MinSup(S), G_r) + 1$ (uma vez que, quando se retira um vértice que pertence ao máximo subgrafo comum de dois grafos a distância somente diminui uma unidade só quando é possível recompor o máximo subgrafo comum novamente novamente, como visto na Seção 3.6). Para os demais grafos, sabe-se que $d(MinSup(S)', G_j) = d(MinSup(S), G_j) - 1,\ j = 1, \ldots, n, j \neq r$. Portanto, estima-se a SOD do grafo $MinSup(S)'$ por:

$$SOD(MinSup(S)') =$$
$$= d(MinSup(S), G_1) - 1 + \ldots + d(MinSup(S), G_r) + 1 + \ldots + d(MinSup(S), G_n) - 1 =$$
$$= SOD(MinSup(S)) - (n-1) + 1 = SOD(MinSup(S)) - n + 2$$

Ou seja, estima-se que haja uma variação de $2 - n$ unidades em relação à SOD inicial, que foi então reduzida.

O algoritmo guloso se baseia em avaliar, para cada vértice do mínimo supergrafo comum, uma estimativa de quanto será reduzida a SOD caso esse vértice seja retirado. Para que seja possível estimar a redução na SOD de $MinSup(S)$ decorrente da retirada de um vértice, é preciso saber, para cada vértice $i \in V_{MinSup(S)}$, se i faz parte ou não de $MaxSub(MinSup(S), G_j)$, $j = 1, \ldots, n$.

Considere-se como exemplo o conjunto de grafos $S = \{G_1, G_2, G_3\}$ na Figura 3.9, cujo mínimo supergrafo comum está representado na Figura 3.10.



(a) G_1 (b) G_2 (c) G_3

Figura 3.9: Conjunto S

Para cada um dos grafos $G_j \in S, j = 1, 2, 3$, existe um isomorfismo de subgrafo de G_i para $MinSup(S)$. Portanto, tem-se que $MaxSub(MinSup(S), G_j) = G_j,\ j = 1, 2, 3$. Nesse caso, os isomorfismos são:

- $f_1 : V_{G_1} \to V_{MinSup(S)}$ com $f_1(1) = 1,\ f_1(2) = 2,\ f_1(3) = 3$.
- $f_2 : V_{G_2} \to V_{MinSup(S)}$ com $f_2(1) = 4,\ f_2(2) = 2,\ f_2(3) = 3$.
- $f_3 : V_{G_3} \to V_{MinSup(S)}$ com $f_3(1) = 4,\ f_3(2) = 5$.

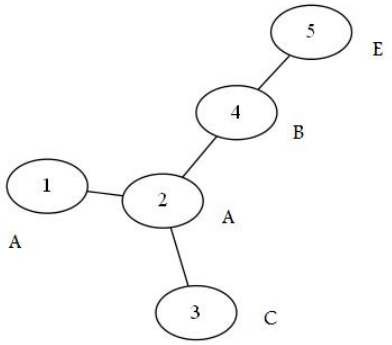


Figura 3.10: Mínimo Supergrafo Comum

Cada uma dessas funções é um "mapeamento" dos vértices do grafo G_j nos vértices de $MinSup(S)$. A Tabela 3.1 resume o mapeamento de cada vértice de $MinSup(S)$. Os símbolos $\exists$ e $\nexists$ são utilizados apenas para indicar se existe ou não mapeamento de um vértice de G_j em algum vértice de $MinSup(S)$. Quando existe mapeamento utiliza-se o símbolo $\exists$ e isso implica que o vértice faz parte do máximo subgrafo comum de G_j e $MinSup(S)$. Por exemplo, como $f_3(2) = 5$, na linha do vértice 5, na coluna do grafo G_3 está o símbolo $\exists$. Isso indica que, nesse momento, o vértice 5 de $V_{MinSup(S)}$ faz parte de $MaxSub(MinSup(S), G_3)$.

Tabela 3.1: Vértices de $MinSup(S)$

V	G_1	G_2	G_3
1	$\exists$	$\nexists$	$\nexists$
2	$\exists$	$\exists$	$\nexists$
3	$\exists$	$\exists$	$\nexists$
4	$\nexists$	$\exists$	$\exists$
5	$\nexists$	$\nexists$	$\exists$

A partir dessa tabela pode-se estimar em quanto será reduzida a SOD atual se for retirado cada vértice de $MinSup(S)$. Seja H_1 o grafo obtido retirando-se o vértice 1 do mínimo supergrafo comum, por exemplo. Existe um vértice do grafo G_1 que está mapeado em 1 ($f_1(1) = 1$), mas nenhum vértice dos grafos G_2 e G_3 está mapeado em 1. Assim, como visto anteriormente, ao se retirar o vértice 1 do mínimo supergrafo comum espera-se que a SOD do novo candidato H_1 tenha um aumento de 1 unidade (devido a estimar-se que $d(H_1, G_1) = d(MinSup, G_1) + 1$, uma vez que o vértice 1 faz parte de $MaxSub(MinSup(S), G_1)$) e uma redução de 2 unidades (devido a estimar-se que $d(H_1, G_2) = d(MinSup(S), G_2) - 1$ e $d(H_1, G_3) = d(MinSup(S), G_3) - 1$). Ou seja, espera-se uma variação de (+1 - 2) = - 1 unidade na SOD do novo candidato; isto é, uma

redução na SOD:

$$\begin{aligned} SOD(H_1) &= d(MinSup(S), G_1) + 1 + d(MinSup(S), G_2) - 1 + d(MinSup(S), G_3) - 1 \\ &= SOD(MinSup(S)) - 1 \end{aligned}$$

Portanto, a ideia é ordenar os vértices de acordo com a estimativa de redução da SOD obtida com a sua retirada. Nesse exemplo, utilizando a Tabela 3.1, teria-se que:

Tabela 3.2: Tabela de estimativas

V	Est_i
5	-1
1	-1
2	+1
3	+1
4	+1

A estratégia deste algoritmo guloso é, portanto, retirar sucessivamente do mínimo supergrafo comum os vértices na ordem decrescente de suas estimativas de redução.

O mapeamento entre o novo candidato H_1 e os grafos $G_j, j = 1, 2, 3$, está representado na Tabela 3.3.

Tabela 3.3: Vértices de H_1

V	G_1	G_2	G_3
2	$\exists$	$\exists$	$\nexists$
3	$\exists$	$\exists$	$\nexists$
4	$\nexists$	$\exists$	$\exists$
5	$\nexists$	$\nexists$	$\exists$

A discussão acima em relação a estimativas de redução de SOD de $MinSup(S)$ continua válida para os grafos obtidos retirando-se vértices de H_1. Assim, neste algoritmo, a partir do grafo $MinSup(S)$, a cada iteração um vértice é retirado (de acordo com a tabela de estimativas) e o grafo obtido tem sua SOD calculada. O algoritmo termina quando o grafo vazio é avaliado, já que esse é o último candidato. Adicionalmente, a estratégia de evitar o cálculo da distância entre dois grafos (Seção 3.6) também é utilizada para aumentar a eficiência do algoritmo. Para que isso seja possível, a tabela que mostra os vértices que compõem $MaxSub(Candidato, G_j)$ é recalculada a cada novo candidato, como ilustrado pelas Tabelas 3.1 e 3.3.

O pseudo-código deste algoritmo é apresentado a seguir: Nas linhas 1-4 é calculado o mínimo supergrafo comum $MinSup$ do conjunto de grafos de entrada por meio da função

MSup, que consiste do procedimento de Bunke para calcular o mínimo supergrafo comum de dois grafos. Nesse momento também é realizado o mapeamento inicial dos vértices dos grafos G_j nos vértices do mínimo supergrafo comum (como ilustrado pela Tabela 3.1). Nas linhas 5 a 8 a SOD de $MinSup$ é calculada e a linha 9 armazena $MinSup$ em *Cand*. Nas linhas 10-11 os resultados obtidos são armazenados como os melhores até o momento. Nas linhas 12-22 são calculadas as estimativas de redução provenientes da retirada de cada vértice de $MinSup$. Nas linhas 23-38 são gerados os grafos candidatos e suas SOD's são avaliadas. A linha 23 indica que o processo continua enquanto houver vértice não selecionado. Na linha 24, k armazena o vértice que possui a menor estimativa Est_i, $i = 1, \ldots, |V_{MinSup}|$, e na linha 25 o vértice é marcado como selecionado. Na linha 26, *Cand_ant* armazena o candidato atual e na linha 27 o vértice k é retirado do grafo *Cand*. Nas linhas 28-34 a distância de *Cand* para cada grafo $G_j \in S$ é calculada. A linha 29 realiza o teste que verifica se o vértice retirado k fazia parte de $MaxSub(Cand\_ant, G_j)$. Em caso afirmativo, na linha 30 a distância entre *Cand* e G_j é calculada (isto é, é preciso computar $MaxSub(Cand, G_j)$). Se o vértice não fazia parte de $MaxSub(Cand\_ant, G_j)$, a linha 32 atribui à distância entre *Cand* e G_j o valor da distância entre *Cand_ant* e G_j menos uma unidade. A cada passagem do bloco 23-34 uma tabela de mapeamento (como as Tabelas 3.1 e 3.3) é montada, de forma que se possa saber o mapeamento dos vértices dos grafos G_j em *Cand* (isto é, para que se conheça $MaxSub(Cand, G_j)$). Nas linhas 35-38 a SOD do candidato é calculada e nas linhas 39-41 é testado se o candidato é o melhor obtido até o momento. No apêndice A é ilustrado o funcionamento deste algoritmo.

3.7.2 Algoritmo guloso com tabela dinâmica (A2)

O algoritmo guloso com tabela dinâmica é muito similar ao algoritmo guloso com tabela estática. Conforme visto, o algoritmo estático começa com o mínimo supergrafo comum, verifica o mapeamento dos grafos $G_i \in S$ no mínimo supergrafo comum, cria uma tabela de estimativas baseada nesse primeiro mapeamento e retira os vértices sempre de acordo com essa tabela inicial (ilustrada pela Tabela 3.2). Entretanto, a medida que vértices vão sendo retirados dos candidatos a grafo mediano generalizado, o mapeamento dos grafos G_i's no candidato pode mudar. Assim, as estimativas Est_i's dos vértices que ainda não foram retirados podem mudar ao longo das iterações.

Para ilustrar essa mudança de mapeamento, a Figura 3.7 apresentada anteriomente mostra dois grafos G e H. Se $S = \{G, H\}$ então $G = MinSup(S)$. Um mapeamento do grafo H no mínimo supergrafo comum é dado por $f : V_H \rightarrow V_{MinSup}$ tal que $f(1) = 1$,

Algoritmo 1 Algoritmo guloso com tabela estática

1: $MinSup \leftarrow \textbf{MSup}(G_1, G_2)$
2: **para** $(j = 3 \ldots n)$ **faça**
3: $\quad MinSup \leftarrow \textbf{MSup}(MinSup, G_j)$
4: **fim para**
5: $SOD \leftarrow 0$
6: **para** $(j = 1 \ldots n)$ **faça**
7: $\quad SOD \leftarrow SOD + d(MinSup, G_j)$
8: **fim para**
9: $Cand \leftarrow MinSup$
10: $SOD^* \leftarrow SOD$
11: $GrafoMedianoGen \leftarrow Cand$
12: **para todo** $(i \in V_{Minsup})$ **faça**
13: $\quad Est_i \leftarrow 0$
14: $\quad Selecionado_i \leftarrow$ `Falso`
15: $\quad$ **para** $j = 1 \ldots n$ **faça**
16: $\quad\quad$ **se** $(i \in MaxSub(MinSup, G_j))$ **então**
17: $\quad\quad\quad Est_i \leftarrow Est_i + 1$
18: $\quad\quad$ **senão**
19: $\quad\quad\quad Est_i \leftarrow Est_i - 1$
20: $\quad\quad$ **fim se**
21: $\quad$ **fim para**
22: **fim para**
23: **enquanto** $(\exists\ i \in V_{Minsup} : Selecionado_i =$ `Falso`$)$ **faça**
24: $\quad k \leftarrow \text{argmin}\{Est_i : Selecionado_i =$ `Falso` e $i \in V_{Minsup}\}$
25: $\quad Selecionado_k \leftarrow$ `Verdadeiro`
26: $\quad Cand\_ant \leftarrow Cand$
27: $\quad V_{Cand} \leftarrow V_{Cand\_ant} - \{k\}$
28: $\quad$ **para** $(j = 1 \ldots n)$ **faça**
29: $\quad\quad$ **se** $(k \in MaxSub(Cand\_ant, G_j))$ **então**
30: $\quad\quad\quad d(Cand, G_j) \leftarrow |Cand| + |G_j| - 2 \cdot |MaxSub(Cand, G_j)|$
31: $\quad\quad$ **senão**
32: $\quad\quad\quad d(Cand, G_j) \leftarrow d(Cand\_ant, G_j) - 1$
33: $\quad\quad$ **fim se**
34: $\quad$ **fim para**
35: $\quad SOD \leftarrow 0$
36: $\quad$ **para** $(j = 1 \ldots n)$ **faça**
37: $\quad\quad SOD \leftarrow SOD + d(Cand, G_j)$
38: $\quad$ **fim para**
39: $\quad$ **se** $(SOD < SOD^*)$ **então**
40: $\quad\quad SOD^* \leftarrow SOD$
41: $\quad\quad GrafoMedianoGen \leftarrow Cand$
42: $\quad$ **fim se**
43: **fim enquanto**

$f(2) = 4$ e $f(3) = 5$. Retirando-se do mínimo supergrafo comum um dos vértices de rótulo B, o vértice 4 por exemplo, o grafo H pode ser remapeado no grafo $MinSup(S)$ através da função $f(1) = 1$, $f(2) = 3$ e $f(3) = 5$.

O algoritmo com tabela dinâmica leva em conta as mudanças nas estimativas. Como visto no algoritmo estático, a cada novo candidato uma tabela de mapeamento entre os grafos $G_j, j = 1, \ldots, n$, e os vértices do candidato é montada (de modo que se conheça $MaxSub(Cand, G_j)$ e seja possível acelerar a avaliação da SOD do próximo candidato). No algoritmo dinâmico, em cada iteração essa tabela de mapeamento será utilizada também para recomputar as estimativas de redução obtidas com a retirada de cada vértice do candidato. Assim, no algoritmo dinâmico, a cada iteração uma tabela similar à Tabela 3.2 é montada e os vértices são retirados de acordo com essa nova tabela de estimativas. É razoável supor que o uso da lista dinâmica seja melhor do que o da estática, uma vez que a primeira é atualizada a cada iteração.

3.7.3 Algoritmo A*

O algoritmo A^* foi proposto por Nilsson, Raphael e Hart [16], ver também [22, 23]. Em linhas gerais, o algoritmo A^* tem como objetivo encontrar o caminho de custo mínimo de um vértice inicial (s) até um vértice alvo (t) de um grafo. Para encontrar tal caminho o algoritmo mantém dois conjuntos: $ABERTOS$, contendo os vértices que ainda podem ser explorados, e $FECHADOS$, contendo os vértices que já foram explorados e até os quais já se conhece o caminho mais curto.

O algoritmo A^* utiliza uma função de custo para estimar o custo dos caminhos de s até t passando por x, dada pela soma de duas funções: $g(x)$, que guarda o custo do caminho do vértice inicial até o vértice x, e $\hat{h}(x)$ que estima o custo do caminho de x até o vértice alvo. Inicialmente, são colocados em $ABERTOS$ os vértices s' que podem ser atingidos partindo de s (filhos de s), e cada um deles é avaliado através da função de custo $\hat{f}(s') = g(s') + \hat{h}(s')$. O vértice de menor estimativa $ABERTOS$ é escolhido, seus filhos são colocados em $ABERTOS$ e ele é colocado em $FECHADOS$. Em seguida, escolhe-se o vértice de $ABERTOS$ que possui a melhor estimativa e repete-se o processo. O algoritmo termina quando o vértice alvo t é selecionado.

O próximo algoritmo para o problema do grafo mediano generalizado é baseado no princípio do algoritmo A^*. No caso, o objetivo é obter um caminho entre o mínimo supergrafo comum (vértice inicial s) e o grafo mediano generalizado do conjunto (vértice alvo t). Os filhos de um grafo são os grafos que podem ser obtidos a partir desse grafo

retirando-se cada um de seus vértices. Como não se conhece de antemão o grafo mediano generalizado (nem sua SOD), não é possível usar a definição original da função de custo. Assim, para avaliar quem é o candidato do conjunto dos ABERTOS com melhor estimativa, será usada a SOD estimada, explicada mais adiante.

O algoritmo funciona da seguinte maneira. Nas linhas 1-4 é computado o mínimo supergrafo comum do conjunto. Nas linhas 5-8 a SOD de $MinSup$ é calculada. A linha 9 inicializa *Cand* com o mínimo supergrafo comum. Nas linhas 10-11, *Cand* e sua SOD são armazenados como o melhor candidato obtido até o momento. Nas linhas 12-22 são calculadas as estimativas de redução da SOD obtidas ao se retirar cada vértice de $MinSup$. A linha 23 inicializa o conjunto $ABERTOS$ como vazio e na linha 24 o conjunto dos $FECHADOS$ tem $MinSup$ como primeiro elemento. Na linha 25 o conjunto $ABERTOS$ recebe os filhos de $MinSup$. Cada filho de $MinSup$ possui uma SOD estimada, obtida a partir da SOD de $MinSup$ atualizando-a. Indicando por V_{MinSup} o conjunto dos vértices de $MinSup$, a SOD estimada de cada filho obtido pela retirada de um vértice k é $SOD_{MinSup} + Est_k$.

A linha 26 indica que o algoritmo termina quando o conjunto ABERTOS fica vazio. Em seguida, na linha 27 é escolhido como candidato o grafo *Cand* que possui a melhor SOD estimada. Escolhido esse novo candidato, sua SOD, da qual se conhecia apenas uma estimativa, é efetivamente calculada nas linhas 28-31. A linha 32 faz o cálculo das estimativas de redução da SOD desse candidato, usando o mesmo procedimento das linhas 12-22 quando as estimativas foram calculadas a partir de $MinSup$. Na linha 33, *Cand* é retirado do conjunto $ABERTOS$. Os filhos de *Cand* são adicionados em $ABERTOS$ e na linha 35 *Cand* é posto no conjunto dos $FECHADOS$. Na linha 36 é verificado se *Cand* é melhor do que a melhor solução obtida até o momento e, caso afirmativo, a melhor solução é atualizada nas linhas 37-38.

Algoritmo 2 Algoritmo A^*

1: $MinSup \leftarrow \textbf{MSup}(G_1, G_2)$
2: **para** $(i = 3 \ldots n)$ **faça**
3: $\quad MinSup \leftarrow \textbf{MSup}(MinSup, G_i)$
4: **fim para**
5: $SOD \leftarrow 0$
6: **para** $(j = 1 \ldots n)$ **faça**
7: $\quad SOD \leftarrow SOD + d(MinSup, G_j)$
8: **fim para**
9: $Cand \leftarrow MinSup$
10: $SOD^* \leftarrow SOD$
11: $GrafoMedianoGen \leftarrow Cand$
12: **para todo** $(i \in V_{Minsup})$ **faça**
13: $\quad Est_i \leftarrow 0$
14: $\quad Selecionado_i \leftarrow$ `Falso`
15: $\quad$ **para** $j = 1 \ldots n$ **faça**
16: $\quad\quad$ **se** $(i \in MaxSub(MinSup, G_j))$ **então**
17: $\quad\quad\quad Est_i \leftarrow Est_i + 1$
18: $\quad\quad$ **senão**
19: $\quad\quad\quad Est_i \leftarrow Est_i - 1$
20: $\quad\quad$ **fim se**
21: $\quad$ **fim para**
22: **fim para**
23: $ABERTOS \leftarrow \emptyset$
24: $FECHADOS \leftarrow \{MinSup\}$
25: $ABERTOS \leftarrow ABERTOS \cup \textbf{Filhos}(MinSup)$
26: **enquanto** $ABERTOS \neq \emptyset$ **faça**
27: $\quad Cand \leftarrow \textbf{SELECMELHORCAND}(ABERTOS)$
28: $\quad SOD \leftarrow 0$
29: $\quad$ **para** $(j = 1 \ldots n)$ **faça**
30: $\quad\quad SOD \leftarrow SOD + d(Cand, G_j)$
31: $\quad$ **fim para**
32: $\quad$ Calcula estimativa de redução de cada vértice do candidato selecionado (*Cand*)
33: $\quad ABERTOS \leftarrow ABERTOS - \{Cand\}$
34: $\quad ABERTOS \leftarrow ABERTOS \cup \textbf{Filhos}(Cand)$
35: $\quad FECHADOS \leftarrow FECHADOS \cup \{Cand\}$
36: $\quad$ **se** $(SOD < SOD^*)$ **então**
37: $\quad\quad SOD^* \leftarrow SOD$
38: $\quad\quad GrafoMedianoGen \leftarrow Cand$
39: $\quad$ **fim se**
40: **fim enquanto**

Capítulo 4

Resultados Computacionais

Este capítulo apresenta os resultados dos experimentos realizados para avaliação dos algoritmos propostos.

Os algoritmos apresentados nos capítulos anteriores foram implementados na linguagem C++ e compilados com o compilador gcc (TDM-2 mingw32) 4.4.1. As execuções foram feitas em um computador Intel i5 com quatro núcleos de 2,8 GHz cada, com quatro Gigabytes de RAM, executado sob o sistema operacional *Windows 7 Home*.

Na Seção 4.1 são apresentadas as instâncias utilizadas e na Seção 4.2 são apresentados os testes feitos e os resultados obtidos.

4.1 Instâncias

As instâncias utilizadas neste trabalho foram obtidas do banco de dados *IAM Graph Database Repository* [5]. Este banco consiste de dez grupos de grafos, sendo que para essa dissertação foi escolhido o grupo *AIDS*.

Este grupo é formado por grafos que representam moléculas e está dividido em duas classes: moléculas ativas e moléculas inativas, de acordo com a sua relação com o vírus da *AIDS*. Este grupo é composto por 2000 grafos e está dividido em três conjuntos:

- um conjunto de treinamento (250 grafos)
- um conjunto de testes (1500 grafos)
- um conjunto de validação (250 grafos)

Os testes realizados neste trabalho foram feitos sobre os grafos do conjunto de testes. Dos 1500 grafos que compõem o conjunto de testes, 300 são ativos e 1200 são inativos em relação ao vírus da *AIDS*.

Os vértices dos grafos possuem elementos químicos como rótulos e suas arestas possuem valores númericos representado o valor da ligação entre os elementos na molécula. O tamanho dos grafos inativos considerados variou de 2 até 12 vértices e dos ativos de 11 até 95 vértices.

4.2 Experimentos e Resultados

Para realizar a avaliação dos algoritmos, foram escolhidos aleatoriamente conjuntos de grafos cujo *número total de vértices* era 20, 40, 60, ..., 180. Foram selecionados aleatoriamente dez conjuntos de grafos para cada quantidade total de vértices. Como exemplo, a instância i01.20 é formada pelos grafos 2128, 6497 e 13072. O grafo 2128 possui nove vértices, o grafo 6497 possui seis vértices e o 13072 é composto por cinco vértices. Assim, o número total de vértices nessa instância é $9 + 6 + 5 = 20$ vértices.

As Tabelas 4.1 a 4.5 mostram os resultados obtidos nos experimentos. Para cada instância é fornecido o número de vértices do mínimo supergrafo comum do conjunto, obtido pelo procedimento aproximado que generaliza o algoritmo de Bunke para dois grafos, e sua *SOD*. A *SOD* do grafo mediano $\hat{G}$ é apresentada para servir de critério de comparação com os resultados obtidos pelos algoritmos. São apresentadas as *SOD*'s e os tempos obtidos pelos dois algoritmos gulosos e as *SOD*'s obtidas pelo algoritmo A^* para os tempos de um, dois e quatro minutos. A descrição das instâncias encontra-se no Apêndice B.

Tabela 4.1: Instâncias formadas por grafos com 20 e 40 vértices no total

				Guloso A1		**Guloso A2**		**A***		
Inst	|MinSup|	SOD MinSup	SOD($\hat{G}$)	SOD	tempo(s)	SOD	tempo(s)	SOD(1')	SOD(2')	SOD(4')
i01.20	13	19	15	13	0.19	13	0.20	12	12	12
i02.20	15	40	22	20	0.18	20	0.18	20	20	20
i03.20	18	52	22	20	0.20	20	0.21	20	20	20
i04.20	13	32	14	14	0.14	14	0.14	14	14	14
i05.20	11	24	10	10	0.19	10	0.19	10	10	10
i06.20	13	32	18	18	0.21	18	0.20	18	18	18
i07.20	11	24	16	16	0.17	16	0.17	16	16	16
i08.20	16	60	23	20	0.17	20	0.17	20	20	20
i09.20	15	55	24	20	0.19	20	0.19	20	20	20
i10.20	14	50	22	19	0.20	19	0.19	19	19	19
i01.40	22	92	38	36	0.55	36	0.53	34	34	34
i02.40	23	98	38	32	0.47	32	0.46	30	30	30
i03.40	23	98	40	38	0.58	38	0.54	38	38	38
i04.40	22	92	36	30	0.46	34	0.42	30	30	30
i05.40	22	92	36	32	0.42	32	0.43	30	30	30
i06.40	21	107	39	34	0.39	34	0.40	33	33	33
i07.40	19	93	38	34	0.41	34	0.41	34	34	34
i08.40	18	86	34	31	0.47	31	0.45	31	31	31
i09.40	24	152	50	40	0.38	40	0.40	40	40	40
i10.40	22	136	38	36	0.46	36	0.45	36	36	36

Tabela 4.2: Instâncias formadas por grafos com 60 e 80 vértices no total

				Guloso A1		**Guloso A2**		**A***		
Inst	|MinSup|	SOD MinSup	SOD($\hat{G}$)	SOD	tempo(s)	SOD	tempo(s)	SOD(1')	SOD(2')	SOD(4')
i01.60	30	150	64	52	1.41	49	1.40	48	48	48
i02.60	27	129	50	42	1.11	43	1.08	41	39	39
i03.60	28	164	48	46	1.18	46	1.15	44	44	44
i04.60	29	172	54	46	0.93	48	0.91	46	46	46
i05.60	21	213	46	46	0.74	45	0.75	45	45	45
i06.60	27	183	58	53	0.63	51	0.64	51	51	51
i07.60	22	160	52	48	0.78	48	0.75	48	48	48
i08.60	22	182	52	47	0.72	46	0.72	45	45	45
i09.60	22	182	50	50	0.75	48	0.71	47	47	47
i10.60	32	324	76	60	0.65	60	0.66	60	60	60
i01.80	33	217	65	60	2.28	61	2.13	56	56	56
i02.80	32	240	70	58	1.58	58	1.54	58	58	58
i03.80	33	283	71	65	1.76	67	1.76	62	62	62
i04.80	32	272	63	58	1.36	55	1.37	55	55	55
i05.80	32	272	75	63	1.21	63	1.21	62	62	62
i06.80	34	294	78	68	2.00	65	1.94	63	63	63
i07.80	33	283	74	69	2.02	71	1.99	66	66	66
i08.80	33	283	74	69	1.49	66	1.47	60	60	60
i09.80	36	316	72	67	1.32	67	1.33	67	67	67
i10.80	35	305	72	69	1.50	67	1.43	67	67	67

Tabela 4.3: Instâncias formadas por grafos com 100 e 120 vértices no total

				Guloso A1		**Guloso A2**		**A***		
Inst	|MinSup|	SOD MinSup	SOD($\hat{G}$)	SOD	tempo(s)	SOD	tempo(s)	SOD(1')	SOD(2')	SOD(4')
i01.100	40	340	89	76	3.89	78	3.51	72	72	72
i02.100	38	356	86	76	2.37	78	2.26	74	74	74
i03.100	38	394	101	83	1.74	85	1.78	82	82	82
i04.100	38	394	96	90	3.18	87	3.10	84	84	84
i05.100	36	368	86	76	2.40	72	2.36	70	70	70
i06.100	38	394	80	72	1.79	73	1.84	70	70	70
i07.100	34	342	81	76	2.19	78	2.17	73	73	73
i08.100	37	418	88	82	2.07	76	2.03	74	74	74
i09.100	37	418	100	88	1.95	84	2.11	82	82	82
i10.100	38	432	100	86	2.80	86	2.69	86	86	86
i01.120	42	426	113	93	5.87	92	5.65	92	92	92
i02.120	41	495	105	97	4.66	93	4.02	94	94	94
i03.120	44	540	126	102	3.22	104	3.18	100	100	100
i04.120	41	495	102	99	4.05	95	4.38	94	94	94
i05.120	41	495	106	105	3.03	99	3.11	94	94	94
i06.120	41	536	124	100	2.53	100	2.54	96	96	96
i07.120	40	520	112	102	2.64	102	2.99	102	102	102
i08.120	37	472	102	86	2.65	94	2.58	86	86	86
i09.120	42	552	106	100	3.14	100	3.33	98	98	98
i10.120	38	488	98	86	2.98	90	2.75	84	84	84

Tabela 4.4: Instâncias formadas por grafos com 140 e 160 vértices no total

				Guloso A1		**Guloso A2**		**A***		
Inst	|MinSup|	SOD MinSup	SOD($\hat{G}$)	SOD	tempo(s)	SOD	tempo(s)	SOD(1')	SOD(2')	SOD(4')
i01.140	50	710	130	112	7.02	117	7.07	115	110	110
i02.140	44	652	122	112	5.32	114	5.63	106	104	104
i03.140	44	696	132	122	3.68	114	3.42	114	114	114
i04.140	44	696	119	109	3.19	107	3.09	101	101	101
i05.140	37	563	112	113	2.97	107	3.00	104	104	104
i06.140	45	760	124	120	3.73	110	3.68	110	110	110
i07.140	49	889	151	124	3.17	124	3.04	124	124	124
i08.140	46	826	148	123	3.11	122	3.07	117	117	117
i09.140	47	847	135	125	3.68	118	3.62	118	115	115
i10.140	42	742	130	118	3.74	118	3.69	117	117	117
i01.160	53	741	139	123	16.60	118	14.67	409	116	116
i02.160	54	812	170	134	8.42	138	8.31	432	128	128
i03.160	50	740	140	130	8.15	134	7.92	148	124	124
i04.160	49	771	145	131	9.17	129	9.93	189	123	123
i05.160	49	771	156	130	11.90	142	11.42	480	130	130
i06.160	55	885	151	129	12.84	136	14.02	681	129	129
i07.160	47	733	143	130	6.50	127	6.93	119	119	119
i08.160	47	733	140	138	6.92	128	6.56	124	124	124
i09.160	39	581	128	125	4.14	115	3.86	115	115	115
i10.160	47	733	134	121	14.91	130	16.28	195	114	113

Tabela 4.5: Instâncias formadas por grafos com 180 vértices no total

				Guloso A1		**Guloso A2**		**A***		
Inst	|MinSup|	SOD MinSup	SOD($\hat{G}$)	SOD	tempo(s)	SOD	tempo(s)	SOD(1')	SOD(2')	SOD(4')
i01.180	56	996	169	153	9.66	149	9.70	595	149	149
i02.180	47	807	152	132	8.20	139	8.26	123	123	123
i03.180	50	870	153	124	11.29	127	16.64	458	124	124
i04.180	47	807	164	144	10.56	155	10.63	403	138	138
i05.180	54	1008	162	144	11.18	138	11.11	586	134	134
i06.180	61	1162	174	152	8.83	152	8.44	578	150	150
i07.180	56	1052	166	144	12.55	148	10.74	772	142	142
i08.180	51	942	160	146	9.99	150	10.27	544	146	146
i09.180	50	920	148	132	31.03	130	46.44	840	740	124
i10.180	51	942	158	144	8.73	144	8.45	136	136	136

Do total de 90 instâncias, o algoritmo A^* apresentou estritamente o melhor resultado em 52 instâncias, o algoritmo A2 (tabela dinâmica) foi estritamente melhor em uma única instância. Ocorreram empates em 37 instâncias (foram considerados empates os casos em que pelo menos um dos algoritmos gulosos conseguiu igualar o valor obtido pelo algoritmo A^*). Ou seja, o algoritmo A^* obteve a melhor solução em 89 das 90 instâncias.

Na comparação entre os algoritmos gulosos A1 e A2, o primeiro foi estritamente melhor em 26 casos e o segundo em 31 casos. Houve empate em 33 instâncias. Em 71 casos o algoritmo A^* não apresentou melhora com o aumento do tempo, isto é, não foi possível em quatro minutos melhorar a solução que havia sido encontrada no primeiro minuto. Com exceção de dois casos (i02.60 e i09.160), as instâncias em que houve melhoria após o primeiro minuto apresentavam mais de 40 vértices no primeiro mínimo supergrafo comum calculado.

A Tabela 4.6 mostra as reduções médias obtidas na SOD do mínimo supergrafo comum para a da melhor solução obtida pelo algoritmo guloso A2. Para cada uma das instâncias foi realizado o cálculo $\frac{SOD_{MinSup}-SOD_{melhorsol}}{SOD_{MinSup}}$ e calculou-se a média dos percentuais encontrados. A mesma comparação é feita entre a SOD do grafo mediano e a SOD da melhor solução.

Tabela 4.6: Percentual médio de redução SOD (algoritmo A2)

Instâncias	$SOD_{Minsup} \rightarrow SOD_{melhorsolA2}$	$SOD_{\hat{G}} \rightarrow SOD_{melhorsolA2}$
20 vértices	52,3%	12,5%
40 vértices	65,6%	16,2%
60 vértices	72,4%	18,1%
80 vértices	76,0%	17,4%
100 vértices	78,8%	19,4%
120 vértices	80,2%	18,5%
140 vértices	83,8%	19,3%
160 vértices	82,3%	17,8%
180 vértices	84,2%	18,6%
Média	75,1%	17,53%

Em média, do valor obtido para a SOD do mínimo supergrafo comum até a melhor solução houve um redução de 75,1%, e do grafo mediano $\hat{G}$ à melhor solução houve uma redução média de 17,53%. Da tabela pode-se perceber que as diferenças entre as SOD do grafo mediano e do grafo mediano aproximado generalizado aumentam a medida que as instâncias se tornam maiores. Com o aumento da quantidade de informações presentes nos grafos, a capacidade do grafo mediano $\hat{G}$ de resumir as características do conjunto se torna cada vez menor, ilustrando a melhoria que pode ser trazida pelo grafo mediano

generalizado. Em todas as tabelas pode-se observar que os algoritmos propostos podem ser utilizados para encontrar grafos melhores do que o grafo mediano do conjunto $\hat{G}$ para representar o conjunto de grafos. A Tabela 4.7 realiza a mesma comparação da Tabela 4.6 para o resultados obtidos pelo algoritmo A^* com 4 minutos.

Tabela 4.7: Percentual médio de redução SOD (algoritmo A^* com 4 minutos)

Instâncias	$SOD_{Minsup} \rightarrow SOD_{melhorsolA^*}$	$SOD_{\hat{G}} \rightarrow SOD_{melhorsolA^*}$
20 vértices	52,8%	13,3%
40 vértices	66,8%	19,3%
60 vértices	73,3%	20,3%
80 vértices	77,1%	20,7%
100 vértices	79,6%	22,8%
120 vértices	80,9%	21,3%
140 vértices	84,3%	22,0%
160 vértices	83,3%	23,1%
180 vértices	85,0%	23,0%
Média	75,9%	20,6%

No caso do algoritmo A^*, a redução média para a SOD do mínimo supergrafo comum até a melhor solução foi de 75,9% e do grafo mediano $\hat{G}$ à melhor solução houve uma redução média de 20,6%. Assim como no caso do algoritmo guloso A2, a SOD obtida pelo algoritmo A^* distância-se da SOD do grafo mediano a medida que o número total de vértices aumenta.

A Figura 4.1 mostra a SOD média de cada um dos algoritmos para cada valor do número total de vértices.

É possível notar que a medida que o número total de vértices aumenta, o algoritmo A^* vai apresentando um desempenho superior aos algoritmos gulosos. Uma possível explicação é o fato de que a medida que os grafos aumentam, os algoritmos gulosos exploram proporcionalmente menos grafos, o que piora o seu desempenho em relação ao A^*.

Na Figura 4.2 é comparada a SOD média dos algoritmos em função do tamanho do mínimo supergrafo comum. Foram calculados os tamanhos de mínimo supergrafo comum que apareceram com maior frequência e computou-se a média dos valores de SOD de cada algoritmo para as instâncias selecionadas (por exemplo, em quatro instâncias o tamanho do mínimo supergrafo comum foi 22, assim calculou-se a SOD média dessas instâncias para cada algoritmo).

Assim como no gráfico da Figura 4.1, o algoritmo A^* mostra um desempenho superior ao dos gulosos. A medida que aumenta o tamanho do mínimo supergrafo comum, o

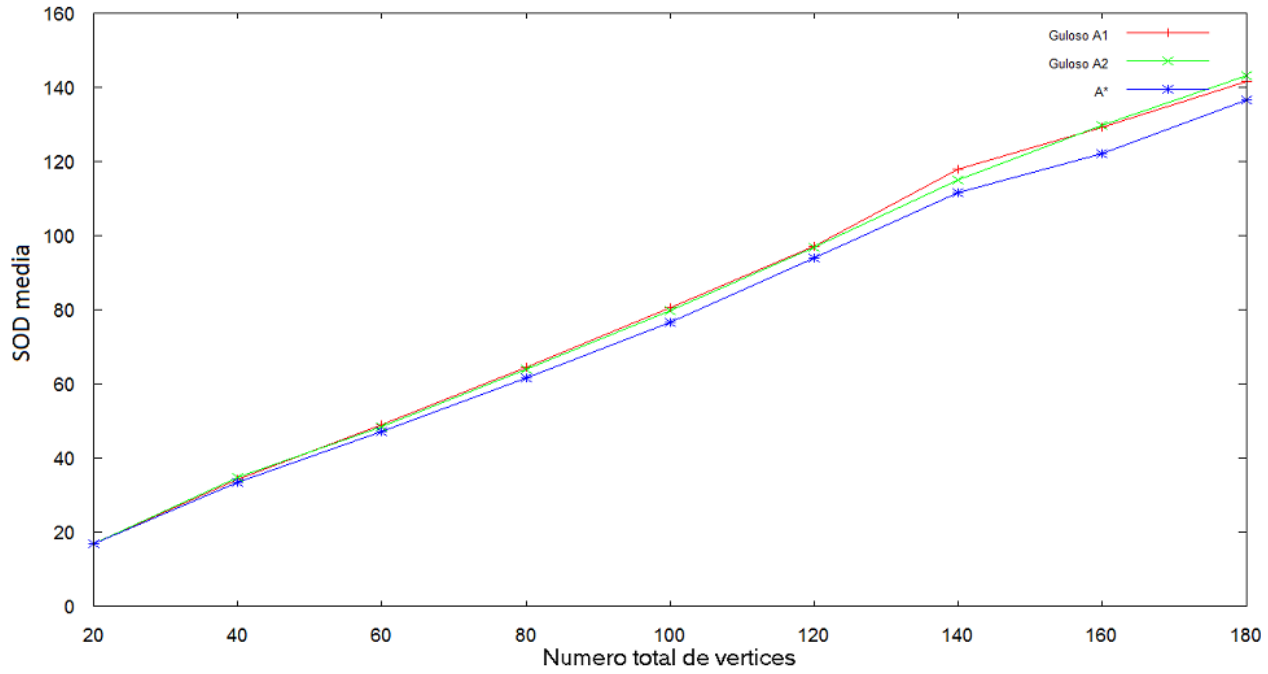


Figura 4.1: SOD média sobre os algoritmos para diferentes números totais de vértices

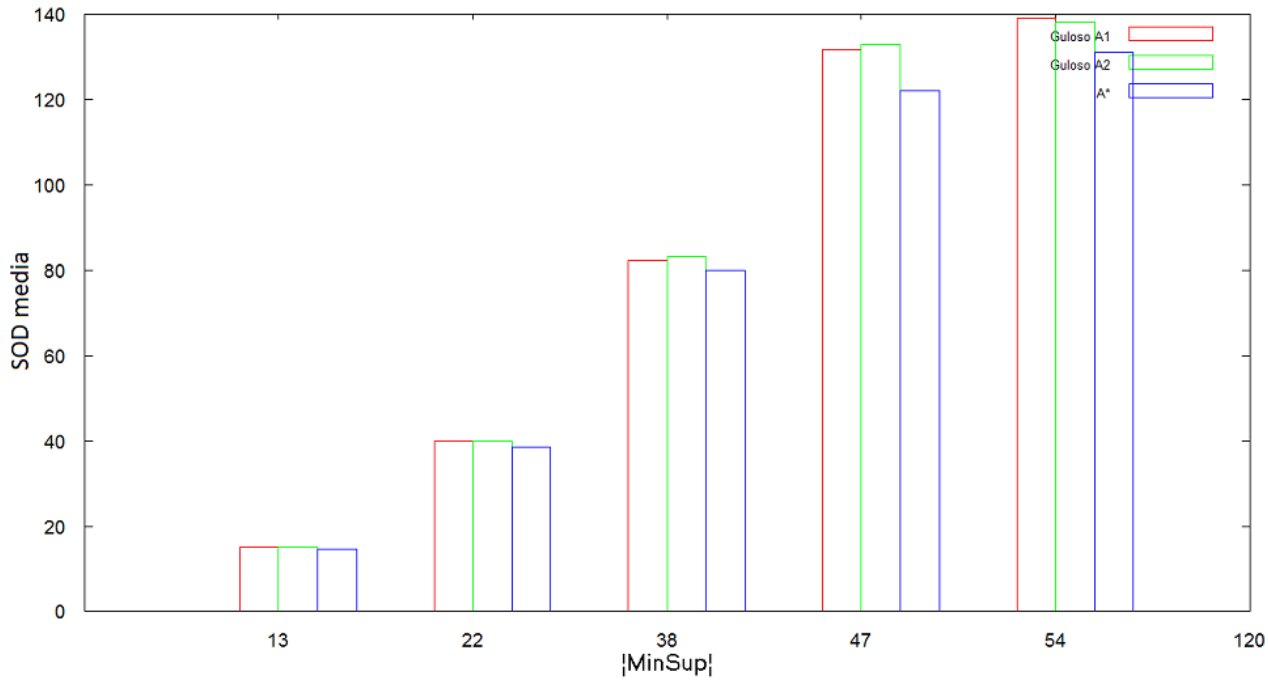


Figura 4.2: SOD média em função do Mínimo supergrafo comum

algoritmo A^* se distancia favoravelmente dos resultados obtidos pelos outros algoritmos.

O objetivo dessas heurísticas era obter soluções aproximadas melhores que o grafo mediano. Considerando-se o algoritmo guloso A1, houve melhora em relação ao grafo mediano em 85 casos, tendo havido 4 empates e um caso em que o grafo mediano é melhor que o candidato aproximado obtido. Em relação ao algoritmo A2, esse foi melhor

que o grafo mediano em 86 casos, ocorrendo apenas quatro empates.

O algoritmo A^* conseguiu obter grafos medianos generalizados melhores do que o grafo mediano em 86 instâncias, tendo havido empates nos outros quatro casos. Embora tanto A2 como A^* tenham obtido grafos medianos generalizados melhores do que o grafo mediano em 86 casos, a redução proporcionada pelo algoritmo A^* é melhor do aquela obtida por A2.

Capítulo 5

Conclusões e trabalhos futuros

Neste trabalho foram apresentadas três heurísticas para o problema do grafo mediano generalizado. Dois dos algoritmos são baseados na retirada sucessiva de vértices, de acordo com estratégias gulosas, enquanto o outro é baseado no princípio do algoritmo A^*. As instâncias utilizadas nos testes computacionais consistem de moléculas relacionadas ao vírus da AIDS. Os resultados obtidos mostraram que as heurísticas foram capazes de obter grafos melhores do que o grafo mediano, em termos da função objetivo. Assim, os algoritmos propostos podem ser utilizados para obter bons representantes para um conjunto de grafos em baixo tempo computacional. Em relação à qualidade das soluções, o algoritmo A^* obteve um desempenho melhor do que os gulosos, mas ao custo de tempos computacionais maiores. Deve-se mencionar que os algoritmos exatos disponíveis na literatura [13] tratam de problemas com instâncias de no máximo 25 vértices no total, enquanto os algoritmos aproximados apresentados nessa dissertação tratam de problemas com até 180 vértices no total.

Como trabalhos futuros, as heurísticas apresentadas podem ser tornadas mais eficientes ou mais eficazes. Conforme visto, os algoritmos gulosos utilizam técnicas para evitar que as distâncias sejam recalculadas (sempre que isso for possível). Devido às estruturas de dados utilizadas na implementação, essa ideia não foi implementada no algoritmo A^* devido a problemas de espaço de memória. Como trabalho futuro, a inclusão dessas técnicas no algoritmo permitirá a avaliação de um número muito maior de candidatos, melhorando ainda mais os grafos aproximados encontrados. Os algoritmos gulosos não terminam necessariamente em ótimos locais. O desenvolvimento de técnicas de busca local poderia melhorar a qualidade dos resultados obtidos. Como próxima etapa os algoritmos gulosos e de busca local podem ser combinados em metaheurísticas como GRASP, busca tabu e outros. De fato, um algoritmo construtivo guloso randomizado para a fase

inicial de uma heurística GRASP pode ser facilmente obtido randomizando-se a escolha do vértice a ser retirado a cada iteração dos algoritmos A1 ou A2, de acordo com a tabela de estimativas.

Os algoritmos exatos existentes para o problema são capazes de lidar apenas com grafos relativamente pequenos. Um trabalho futuro consiste em uma abordagem exata baseada em programação inteira para o problema, para possivelmente tratar instâncias maiores em tempos aceitáveis do ponto de vista prático. Outra possibilidade é utilizar métodos híbridos, combinando métodos exatos com metaheurísticas.

Ainda como trabalho futuro, os algoritmos exatos e híbridos podem ser utilizados em diferentes aplicações. Uma dessas aplicações, por exemplo, seria na área de *prototype learning*, na qual várias imagens de um mesmo objeto são feitas, mas pequenas diferenças são introduzidas devido a ruídos no processo. O objetivo é encontrar a imagem mediana, isto é, encontrar um modelo que melhor represente esse conjunto de imagens [24].

Referências

[1] Bengoetxea, E. *Inexact Graph Matching Using Estimation of Distribution Algorithms*. Tese de Doutorado, Ecole Nationale Supérieure des Télécommunications, Paris, 2002.

[2] Boeres, M. *Heurísticas para reconhecimento de cenas por correspondência entre grafos*. Tese de Doutorado, COPPE, Universidade Federal do Rio de Janeiro, 2002.

[3] Boeres, M.; Ribeiro, C.; Bloch, I. A randomized heuristic for scene recognition by graph matching. *Lecture Notes in Computer Science 3059* (2004), 101–114.

[4] Bunke, H. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters 18* (1997), 689–694.

[5] Bunke, H. Graph representation for intelligent information processing - fundamentals and algorithms for classification and clustering, 2011. Referência online em http://cvpr-ss-2010.cecs.anu.edu.au/pdfs/HorstBunke.pdf, última visita em 02/01/2013.

[6] Bunke, H.; Foggia, P.; Guidobaldi, C.; Sansone, C.; Vento, M. A comparison of algorithms for maximum common subgraph on randomly connected graphs. *Lecture Notes in Computer Science 2396* (2002), 123–132.

[7] Bunke, H.; Foggia, P.; Guidobaldi, C.; Vento, M. Graph clustering using the weighted minimum common supergraph. In *Proceedings of the 4th IAPR International Conference on Graph Based Representations in Pattern Recognition* (Berlin, Heidelberg, 2003), Graph Based Representations in Pattern Recognition'03, Springer-Verlag, pp. 235–246.

[8] Bunke, H.; Jiang, X.; Kandel, A. On the minimum common supergraph of two graphs. *Computing 65* (2000), 13–25.

[9] Bunke, H.; Riesen, K. Towards the unification of structural and statistical pattern recognition. *Pattern Recognition Letters 33* (2012), 811–825.

[10] Conte, D.; Foggia, P.; Vento, M. Challenging complexity of maximum common subgraph detection algorithms: A performance analysis of three algorithms on a wide database of graphs. *Journal of Graph Algorithms and Applications 11* (2007), 99–143.

[11] de la Higuera, C.; Casacuberta, F. Topology of strings: Median string is NP-complete. *Theoretical Computer Science 230* (2000), 39–48.

[12] Durand, P.; Pasari, R.; Baker, J.; Tai, C. An efficient algorithm for similarity analysis of molecules, 1999. Referência online em http://www.cs.kent.edu/∼jbaker/paper/, última visita em 03/01/2013.

[13] Ferrer, M. *Theory and Algorithms on the Median Graph. Application to Graph-based Classification and Clustering.* Tese de Doutorado, Universitat Autonoma de Barcelona, Belaterra, 2008.

[14] Fortin, S. The graph isomorphism problem, 1996. Referência online em http://www.iocd.unam.mx/organica/seminario/graph.pdf, última visita em 03/01/2013.

[15] Garey, M.; Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, 1979.

[16] Hart, P.; Nilsson, N.; Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics SSC-4* (1968), 100–107.

[17] Jiang, X.; Munger, A.; Bunke, H. On median graphs: Properties, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence 23* (2001), 1144–1151.

[18] Juan, A.; Vidal, E. Fast median search in metric spaces. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition* (London, 1998), Springer-Verlag, pp. 905–912.

[19] Mukherjee, L.; Singh, V.; Peng, J.; Xu, J.; Zeitz, M. J.; Berezney, R. Generalized median graphs: Theory and applications. In *Proceedings of the IEEE 11th International Conference on Computer Vision* (Buffalo, 2007), IEEE, pp. 1–8.

[20] Mukherjee, L.; Singh, V.; Peng, J.; Xu, J.; Zeitz, M. J.; Berezney, R. Generalized median graphs and applications. *Journal of Combinatorial Optimization 17*, 1 (2009), 21–44.

[21] Neuhaus, M.; Riesen, K.; Bunke., H. Fast suboptimal algorithms for the computation of graph edit distance. *Lecture Notes in Computer Science 4109* (2006), 163–172.

[22] Nilsson, N. *Problem-Solving Methods in Artificial Intelligence.* McGraw-Hill, 1971.

[23] Nilsson, N. *Principles of Artificial Intelligence.* Springer-Verlag, 1982.

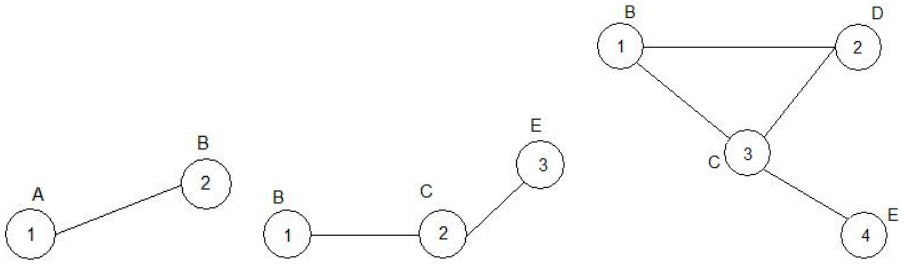
[24] Raveaux, R.; Adam, S.; Héroux, P.; Trupin, I. Learning graph prototypes for shape recognition. *Computer Vision and Image Understanding 115* (2011), 905–918.

[25] Voigt, K. Semi-automatic matching of heterogeneous model-based specifications. *Lecture Notes in Informatics - Software Engineering (Workshops) P-160* (2010), 537–542.

APÊNDICE A – Exemplo de funcionamento do algoritmo guloso A1

Neste apêndice é ilustrado o funcionamento do algoritmo guloso A1 (tabela estática).

Considere-se como exemplo o conjunto S formado pelos grafos G_1, G_2 e G_3, ilustrados na Figura A.1. O mínimo supergrafo comum desse conjunto está representado na Figura A.2.



(a) Grafo G_1 (b) Grafo G_2 (c) G_3

Figura A.1: Conjunto S

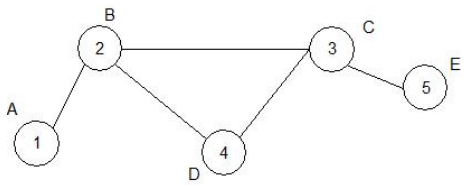


Figura A.2: $MinSup(S)$

O valor da SOD de $MinSup(S)$ é:

$$\begin{aligned} SOD(MinSup(S)) &= d(MinSup(S), G_1) + d(MinSup(S), G_2) + d(MinSup(S), G_3) \\ &= 3 + 2 + 1 = 6 \end{aligned}$$

A Tabela A.1 mostra os mapeamentos dos grafos G_i nos vértices de $MinSup(S)$, informando se um vértice $i \in V_{MinSup}$ está ou não mapeado em algum vértice do grafo

$MaxSub(MinSup(S), G_i)$, $i = 1, 2, 3$. A partir da Tabela A.1 é possível montar a Tabela A.2, que mostra as estimativas de redução.

Tabela A.1: Vértices $MinSup(S)$

V_{MinSup}	G_1	G_2	G_3
1	$\exists$	$\nexists$	$\nexists$
2	$\exists$	$\exists$	$\exists$
3	$\nexists$	$\exists$	$\exists$
4	$\nexists$	$\nexists$	$\exists$
5	$\nexists$	$\exists$	$\exists$

Tabela A.2: Tabela de estimativas

V_{MinSup}	Est_i
4	-1
1	-1
3	+1
5	+1
2	+3

Como este é o algoritmo de tabela estática, os vértices serão retirados na ordem da Tabela A.2. O vértice 4 é o primeiro a ser retirado. O novo candidato $MinSup(S)^{(1)}$ está representado na Figura A.3:

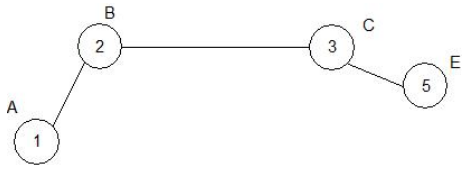


Figura A.3: $MinSup(S)^{(1)}$

Como o vértice 4 não faz parte de $MaxSub(MinSup(S), G_1)$ e nem de $MaxSub(MinSup(S), G_2)$, sabe-se que:

$$d(MinSup(S)^{(1)}, G_1) = d(MinSup(S), G_1) - 1$$

e

$$d(MinSup(S)^{(1)}, G_2) = d(MinSup(S), G_2) - 1.$$

Assim, apenas $d(MinSup(S)^{(1)}, G_3)$ precisa ser calculada. A SOD desse novo candidato é:

$$SOD(MinSup(S)^{(1)}) = d(MinSup(S)^{(1)}, G_1) + d(MinSup(S)^{(1)}, G_2) + d(MinSup(S)^{(1)}, G_3)$$
$$= 2 + 1 + 2 = 5.$$

É preciso encontrar a nova tabela de mapeamento. Como não foram recalculadas as distâncias de $MinSup(S)^{(1)}$ para G_1 e G_2 (mais precisamente, não foi necessário computar $MaxSub(MinSup(S)^{(1)}, G_i)$, i=1,2), as colunas de G_1 e G_2 não irão sofrer alterações. Como foi recalculada a distância de $MinSup(S)^{(1)}$ para G_3, existe a possibilidade de a coluna associada a G_3 mudar. A Tabela A.3 indica o novo mapeamento dos grafos G_i em $MinSup(S)^{(1)}$.

Tabela A.3: Vértices $MinSup(S)^{(1)}$

V_{MinSup^1}	G_1	G_2	G_3
1	$\exists$	$\nexists$	$\nexists$
2	$\exists$	$\exists$	$\exists$
3	$\nexists$	$\exists$	$\exists$
5	$\nexists$	$\exists$	$\exists$

Aqui reside a diferença entre o algoritmo de tabela estática e o de tabela dinâmica. Neste algoritmo de tabela estática já está definido (pela Tabela A.2) que o próximo vértice a ser retirado é o vértice 1. No algoritmo de tabela dinâmica, a Tabela A.3 é utilizada para gerar uma nova tabela de estimativas, que determinaria uma nova ordem de retirada dos vértices. Para este algoritmo de tabela estática, a Tabela A.3 serve apenas para saber se um vértice de $MinSup(S)^{(1)}$ faz parte ou não de $MaxSub(MinSup(S)^{(1)}, G_i)$ (pois essa informação é necessária para evitar o recálculo das distâncias para o próximo candidato).

O vértice 1 é o próximo a ser retirado. A Figura A.4 mostra o novo candidato $MinSup(S)^{(2)}$.

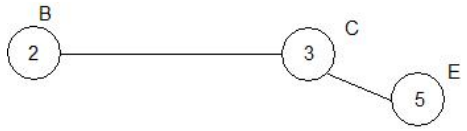


Figura A.4: $MinSup(S)^{(2)}$

Da mesma forma, como o vértice 1 não faz parte de $MaxSub(MinSup(S)^{(1)}, G_2)$ e $MaxSub(MinSup(S)^{(1)}, G_3)$ (Tabela A.3), não é preciso recalcular a distância entre o novo candidato $MinSup(S)^{(2)}$ e os grafos G_2 e G_3.

A SOD do novo candidato é, portanto:

$$SOD(MinSup(S)^2) = d(MinSup(S)^{(2)}, G_1) + d(MinSup(S)^{(2)}, G_2) + d(MinSup(S)^{(2)}, G_3)$$
$$= 3 + 0 + 1 = 4.$$

A Tabela A.4 mostra os mapeamentos dos vértices do grafo $MinSup(S)^{(2)}$.

Tabela A.4: Vértices $MinSup(S)^{(2)}$

V_{MinSup^2}	G_1	G_2	G_3
2	$\exists$	$\exists$	$\exists$
3	$\nexists$	$\exists$	$\exists$
5	$\nexists$	$\exists$	$\exists$

O vértice 3 é o próximo a ser retirado. Na Tabela A.2 a estimativa do vértice 3 é $+1$. Assim, estima-se que a SOD do novo candidato seja aumentada em uma unidade. A Figura A.5 ilustra o novo candidato:

Figura A.5: $MinSup(S)^{(3)}$

A SOD desse candidato é:

$$SOD(MinSup(S)^{(3)}) = d(MinSup(S)^{(3)}, G_1) + d(MinSup(S)^{(3)}, G_2) + d(MinSup(S)^{(3)}, G_3)$$
$$= 2 + 1 + 2 = 5.$$

Pela Tabela A.4, o vértice 3 não fazia parte de $MaxSub(MinSup(S)^{(2)}, G_1)$ e fazia parte dos máximos subgrafos comuns de $MinSup(S)^{(2)}$ com G_2 e G_3. Assim, não foi necessário recalcular a distância $d(MinSup(S)^{(3)}, G_1)$, bastou tomar $d(MinSup(S)^{(3)}, G_1) = d(MinSup(S)^{(2)}, G_1) - 1$. Para G_2 e G_3 foi preciso recalcular as distâncias (isto é, foi preciso computar o máximo subgrafo comum). Como dito anteriormente, quando há a computação do máximo subgrafo comum é possível que o mapeamento de um grafo G_i nos vértices do candidato mude, conforme ilustrado na Seção 3.6. No pseudo-código do algoritmo A1, quando é feito o cálculo da distância $d(Cand, G_j)$ o algoritmo monta a coluna relativa ao grafo G_j na tabela de mapeamento. Se foi necessário computar o máximo subgrafo comum entre $Cand$ e G_j (linha 30) então a própria computação desse máximo

subgrafo comum fornece o preenchimento da coluna G_j. Se a distância $d(Cand, G_j)$ foi calculada pela linha 32 então a coluna G_j da tabela de mapeamento é a repetição da coluna G_j da tabela de mapeamento anterior.

A Tabela A.5 mostra o mapeamento do grafo $MinSup(S)^{(3)}$.

Tabela A.5: Vértices $MinSup(S)^{(3)}$

V_{MinSup^3}	G_1	G_2	G_3
2	$\exists$	$\exists$	$\exists$
5	$\nexists$	$\exists$	$\exists$

O vértice 5 é o próximo a ser retirado. A Figura A.6 ilustra o novo candidato $MinSup(S)^{(4)}$.

Figura A.6: $MinSup(S)^{(4)}$

A SOD desse candidato é:

$$\begin{aligned} SOD(MinSup(S)^{(4)}) &= d(MinSup(S)^{(4)}, G_1) + d(MinSup(S)^{(4)}, G_2) + d(MinSup(S)^{(4)}, G_3) \\ &= 1 + 2 + 3 = 6 \end{aligned}$$

O mapeamento dos vértices de $MinSup(S)^{(4)}$ aparece na Tabela A.6.

Tabela A.6: Vértices $MinSup(S)^{(4)}$

V_{MinSup^4}	G_1	G_2	G_3
2	$\exists$	$\exists$	$\exists$

Retirando-se o vértice 2 obtém-se o grafo vazio G_e, cuja SOD é:

$$\begin{aligned} SOD(G_e) &= d(G_e, G_1) + d(G_e, G_2) + d(G_e, G_3) \\ &= 2 + 3 + 4 = 9. \end{aligned}$$

Assim, a melhor solução encontrada pelo algoritmo foi o grafo $MinSup(S)^{(2)}$, com $SOD(MinSup(S)^{(2)}) = 4$.

APÊNDICE B – Descrição das instâncias

As instâncias foram retiradas do banco de dados *IAM Graph Database Repository*. Foram usados grafos inativos do conjunto de testes.

Tabela B.1: Instâncias com número total de vértices igual a 20

Instância	Grafos
i1.20	2128, 6497, 13072
i2.20	153, 973, 4493, 8117
i3.20	4901, 6075, 6614, 8117
i4.20	813, 1541, 8117, 22446
i5.20	153, 718, 4901, 14734
i6.20	41, 262, 646, 5461
i7.20	646, 718, 1540, 21840
i8.20	153, 2325, 5727, 7910, 8117
i9.20	41, 2325, 5727, 6073, 8117
i10.20	914, 5727, 6075, 8117, 13072

Tabela B.2: Instâncias com número total de vértices igual a 40

Instância	Grafos
i1.40	8117, 6075, 1540, 4446, 37148, 31114
i2.40	8117, 1540, 165, 1079, 23533, 29886
i3.40	153, 5727, 7910, 31574, 2205, 12741
i4.40	153, 13072, 1540, 7796, 36308, 36531
i5.40	6075, 1540, 973, 607, 6801, 1561
i6.40	8117, 13072, 7910, 1540, 21840, 2803, 3181
i7.40	41, 718, 5461, 14787, 607, 2199, 10567
i8.40	5461, 718, 13072, 7910, 1540, 770, 25925
i9.40	8117, 5727, 153, 1540, 7910, 348, 7411, 5497
i10.40	8117, 153, 22446, 4901, 150, 41, 90, 2528

Tabela B.3: Instâncias com número total de vértices igual a 60

Instância	Grafos
i1.60	8117, 5727, 4142, 28329, 14689, 21760, 2454
i2.60	5461, 31, 22, 6266, 5048, 9939, 41363
i3.60	22446, 4901, 6073, 646, 486, 1322, 1812, 18026
i4.60	13072, 7910, 11016, 4446, 10567, 5067, 1026, 12427
i5.60	8117, 153, 4901, 22446, 646, 718, 150, 6073, 5461, 7910, 5268, 9901, 21825
i6.60	153, 5727, 718, 1137, 1744, 2632, 231, 3442, 7513
i7.60	8117, 41, 5461, 150, 1540, 426, 107, 4257, 2291, 11563
i8.60	153, 4901, 22446, 718, 19440, 11016, 973, 31, 97, 6497, 2728
i9.60	718, 150, 6073, 4901, 22446, 973, 31, 6497, 11216, 1079, 4464
i10.60	8117, 5727, 153, 6073, 6075, 13072, 1540, 1541, 426, 348, 14912, 5067

Tabela B.4: Instâncias com número total de vértices igual a 80

Instância	Grafos
i1.80	7910, 13072, 1540, 1089, 28915, 975, 11095, 23414, 16664
i2.80	153, 11216, 56, 7411, 21840, 34256, 986, 38951, 23129, 6191
i3.80	8117, 153, 7910, 1540, 13072, 4250, 1486, 25316, 6081, 32791, 11958
i4.80	8117, 1540, 18105, 7834, 97, 7575, 6262, 16370, 10338, 14562, 38971
i5.80	8117, 13072, 348, 21840, 261, 7899, 3344, 1259, 30821, 38, 2029
i6.80	5727, 153, 5461, 22446, 18105, 19440, 6239, 14286, 10498, 16999, 1200
i7.80	5727, 6073, 22446, 13072, 7910, 11216, 5581, 28437, 8478, 16013, 23125
i8.80	153, 914, 426, 246, 7834, 3488, 31, 20691, 21829, 25951, 373
i9.80	6073, 5461, 718, 150, 6075, 1540, 40760, 1154, 32167, 12203, 17167
i10.80	6073, 22446, 7910, 1540, 14930, 9901, 486, 22331, 5057, 1481, 2648

Tabela B.5: Instâncias com número total de vértices igual a 100

Instância	Grafos
i1.100	6075, 1540, 11216, 1079, 2291, 36894, 28336, 3058, 1640, 11958, 2075
i2.100	8117, 607, 3344, 31574, 1323, 2694, 35059, 20681, 1861, 5811, 3056, 1422
i3.100	8117, 5727, 13072, 1540, 7910, 7796, 21825, 9834, 1974, 24418, 24420, 133, 22695
i4.100	153, 5727, 646, 718, 1540, 97, 426, 23558, 25929, 16703, 14029, 2817, 1455
i5.100	153, 5727, 22446, 1540, 214, 7899, 266, 18907, 6614, 2818, 27893, 10646, 34221
i6.100	6075, 22446, 1540, 973, 19440, 983, 4257, 2051, 30003, 11064, 3010, 427, 3616
i7.100	150, 7910, 13072, 1540, 214, 803, 7281, 6801, 1079, 1498, 1152, 1917, 22696
i8.100	8117, 5727, 153, 1540, 13072, 11216, 1079, 214, 15698, 2029, 6643, 1422, 17570, 1014
i9.100	8117, 153, 41, 2097, 14930, 7281, 8652, 13480, 1119, 403, 4446, 3182, 26336, 2445
i10.100	5727, 153, 6075, 4901, 6073, 13072, 1540, 1885, 3182, 17256, 10936, 31998, 10858, 18928

Tabela B.6: Instâncias com número total de vértices igual a 120

Instância	Grafos
i1.120	8117, 261, 12326, 813, 8960, 11425, 2356, 2238, 25358, 37328, 700, 17958, 18682
i2.120	8117, 6073, 646, 718, 11016, 13480, 8652, 2097, 5015, 10320, 21831, 1322, 23600, 10706, 14511
i3.120	8117, 6073, 12431, 214, 90, 41423, 14974, 11041, 4257, 3762, 20702, 7870, 696, 24628, 16694
i4.120	153, 41, 718, 646, 2097, 262, 3759, 21825, 34210, 9009, 28389, 10674, 16402, 11788, 2222
i5.120	22446, 718, 6075, 41, 646, 15698, 22448, 252, 38639, 38974, 30112, 14313, 11369, 15841, 8331
i6.120	8117, 153, 5727, 6073, 34878, 2803, 18931, 31569, 3442, 1089, 1622, 1694, 1259, 1391, 2488, 1085
i7.120	8117, 4901, 6073, 150, 1540, 348, 261, 22448, 1573, 377, 1017, 9335, 1640, 20685, 14920, 1315
i8.120	153, 13072, 7910, 1540, 5330, 1137, 6801, 347, 6757, 35332, 27861, 20691, 1732, 52, 36308, 537
i9.120	22446, 646, 5461, 41, 18105, 2856, 6801, 262, 12116, 3182, 22767, 14899, 1854, 38982, 10982, 1979
i10.120	6073, 150, 5461, 1540, 11216, 13480, 1079, 5268, 7575, 1573, 8652, 1974, 7078, 8165, 25930, 11215

Tabela B.7: Instâncias com número total de vértices igual a 140

Instância	Grafos
i1.140	153, 5727, 22446, 973, 14787, 7834, 2694, 2688, 1316, 25935, 2263, 10320, 7870, 16381, 29598, 16392, 2099
i2.140	5727, 150, 1540, 13072, 7910, 14787, 914, 90, 102, 34412, 2128, 2728, 1437, 2275, 20708, 13307, 27162, 18330
i3.140	153, 5727, 6073, 646, 1540, 7910, 56, 12431, 347, 3940, 45, 20702, 36307, 10962, 42366, 13468, 15283, 23496, 34258
i4.140	153, 1540, 7910, 973, 3488, 19440, 246, 97, 7834, 983, 214, 2418, 4464, 6036, 32867, 9642, 2967, 337, 18868
i5.140	6073, 150, 22446, 1540, 6497, 56, 1541, 973, 348, 4250, 2291, 28389, 3761, 10151, 20702, 38312, 16370, 23527, 35328
i6.140	153, 5727, 41, 6073, 646, 13072, 6497, 56, 914, 607, 11216, 12326, 377, 22636, 5307, 13195, 10301, 11426, 14337, 1822
i7.140	8117, 153, 5727, 6075, 646, 13072, 7910, 1540, 6153, 1573, 7411, 90, 635, 770, 1026, 22331, 6104, 243, 23612, 10979, 11215
i8.140	8117, 5727, 6073, 718, 6075, 11216, 607, 56, 97, 246, 12116, 948, 16598, 486, 27584, 12326, 6104, 39570, 14899, 118, 20682
i9.140	8117, 5727, 6073, 13072, 7910, 14787, 97, 56, 1541, 19440, 6497, 11016, 3488, 90, 12116, 2028, 3940, 3578, 28597, 2742, 258
i10.140	8117, 41, 6073, 6075, 4901, 150, 5461, 718, 7910, 13072, 1540, 7834, 10567, 39570, 6787, 27870, 25530, 3578, 27735, 22214, 10245

Tabela B.8: Instâncias com número total de vértices igual a 160

Instância	Grafos
i1.160	1540, 246, 5330, 813, 31574, 3759, 2025, 45, 6266, 34256, 2408, 1738, 2709, 16013, 11819, 1570, 26420
i2.160	8117, 153, 5727, 9600, 4493, 2144, 1769, 28915, 27870, 6036, 5811, 34945, 2151, 18018, 13418, 865, 12459, 16999
i3.160	41, 150, 11216, 97, 426, 14787, 3344, 14931, 8130, 4479, 10247, 2078, 23035, 11606, 1455, 10674, 16260, 23984
i4.160	8117, 646, 5461, 13072, 7796, 335, 27584, 8960, 27321, 1437, 6002, 2013, 2728, 1981, 11036, 1722, 1561, 1315, 25360
i5.160	8117, 6075, 7910, 6497, 914, 7411, 7575, 5330, 36116, 4257, 9834, 10094, 21875, 5934, 10498, 8104, 15286, 19619, 14338
i6.160	8117, 7910, 1540, 13072, 107, 973, 11216, 261, 1970, 20478, 3940, 2630, 2244, 16616, 19619, 18037, 18040, 18637, 12761
i7.160	5727, 153, 2856, 9901, 6104, 27584, 5067, 22636, 1744, 20680, 2487, 8394, 1585, 3762, 28597, 14227, 1178, 12340, 11819
i8.160	22446, 6073, 6075, 13072, 1540, 2856, 1431, 1137, 3759, 4493, 35405, 7767, 34224, 16872, 12494, 12360, 26420, 25929, 14920
i9.160	718, 5461, 22448, 2739, 5330, 7899, 4250, 486, 1704, 1440, 52, 20478, 5595, 3363, 22, 7711, 15286, 12283, 16758
i10.160	646, 7834, 11016, 348, 426, 14787, 1541, 18105, 19265, 811, 6266, 6261, 463, 2968, 35329, 1246, 1776, 27970, 12360

Tabela B.9: Instâncias com número total de vértices igual a 180

Instância	Grafos
i1.180	8117, 7910, 1540, 14787, 3488, 90, 6801, 214, 15698, 30, 23612, 21734, 25256, 18018, 8582, 23614, 37, 15590, 39692, 14944, 1765
i2.180	8117, 7910, 1540, 2856, 7125, 14974, 10301, 3363, 30, 6787, 3064, 18878, 1641, 5581, 33085, 38312, 42366, 34218, 18364, 1210, 12313
i3.180	5727, 13072, 7411, 1079, 2739, 1885, 347, 7796, 15698, 6104, 2694, 62, 2632, 991, 857, 6204, 22695, 27162, 230, 21866, 10180
i4.180	6073, 6075, 41, 4901, 7910, 973, 56, 33085, 4151, 6614, 857, 34218, 2818, 1861, 1889, 34416, 35264, 11390, 13888, 1994, 12744
i5.180	8117, 5727, 5461, 13072, 1540, 7834, 1454, 34878, 23493, 30809, 69, 13239, 12262, 3010, 1481, 2263, 8894, 21332, 34846, 1051, 38, 7444
i6.180	8117, 153, 7910, 13072, 246, 348, 18105, 11016, 11216, 56, 2630, 9600, 30, 6964, 3895, 32873, 19353, 133, 10670, 26693, 11841, 116
i7.180	5727, 153, 6075, 4901, 5461, 41, 6073, 1885, 2487, 20680, 6396, 28389, 2052, 24934, 9915, 6119, 1496, 19622, 19595, 16716, 23802, 18954
i8.180	5727, 153, 4901, 41, 1540, 7910, 13072, 973, 3759, 37660, 21825, 4962, 4550, 21044, 3892, 4897, 36531, 1854, 10963, 19353, 2175, 14338
i9.180	22446, 6073, 5268, 2856, 137, 90, 7575, 15698, 7411, 3759, 14973, 486, 2418, 1089, 28915, 36307, 475, 8394, 2275, 10979, 10670, 10256
i10.180	1540, 7910, 246, 56, 973, 3488, 14930, 7899, 3344, 7796, 2097, 261, 1431, 7125, 335, 34217, 10858, 19085, 11777, 14406, 21594, 28074