HEURÍSTICAS PARA RECONHECIMENTO DE CENAS
POR CORRESPONDÊNCIA DE GRAFOS

Maria Claudia Silva Boeres

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE PRODUÇÃO.

Aprovada por:

Prof. Celso Carneiro Ribeiro, Dr. Habil.

Profa. Nair Maria Maia de Abreu, D.Sc.

Profa. Isabelle Bloch, Ph.D

Profa. Lilian Markenzon, D.Sc.

Prof. Roberto Diéguez Galvão, Ph.D

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2002

BOERES, MARIA CLAUDIA SILVA

Heurísticas para reconhecimento de cenas por correspondência de grafos [Rio de Janeiro] 2002

XIII, 108 p., 29.7 cm, (COPPE/UFRJ, D.Sc., Engenharia de Produção, 2002)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1 - Correspondência de grafos

2 - Otimização combinatória

3 - Heurísticas

I. COPPE/UFRJ II. Título (Série)

Ao meu pai

Agradecimentos

Agradeço a todos que me permitiram um grande aprendizado com a realização deste trabalho. Em especial, agradeço:

Ao Prof. Celso Carneiro Ribeiro, pela amizade, compreensão e excelente orientação recebida.

À Profa. Nair Maria Maia de Abreu, pelo apoio, amizade e constante orientação em minha "vida de pesquisa".

À Profa. Isabelle Bloch, pela orientação no problema de aplicação deste trabalho, pela simplicidade e riqueza de aprendizado em nossas diversas interações.

À CAPES pelo apoio financeiro através da bolsa de doutorado.

Ao Departamento de Informática da Universidade Federal do Espírito Santo, que concedeu o afastamento para realização deste doutorado.

Aos professores e funcionários do Programa de Engenharia de Produção da COPPE-UFRJ pela ajuda recebida para a realização deste curso. Agradeço em especial a Andréia Lima da Silva, secretária da PO.

Aos meus queridos amigos da COPPE, UFES, PUC-Rio, ENST-Paris e UFF, cujo carinho e apoio recebidos durante todo esse período se constituem em um dos melhores "resultados"deste trabalho.

À minha família, cujo apoio está implícito em nossas relações. Em especial a meu pai, que deve estar vibrando junto comigo neste momento.

À Deus, por esta oportunidade e por estes amigos.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

HEURÍSTICAS PARA RECONHECIMENTO DE CENAS POR CORRESPONDÊNCIA DE GRAFOS

Maria Claudia Silva Boeres

Setembro/2002

Orientadores: Celso Carneiro Ribeiro
Nair Maria Maia de Abreu
Programa: Engenharia de Produção

Considera-se nesta tese o problema de reconhecimento de cenas em imagens complexas. A motivação para o estudo deste problema surgiu de uma aplicação na área de reconhecimento de imagens médicas. Nesta aplicação, pretende-se efetuar o reconhecimento de estruturas tridimensionais do cérebro humano, a partir de imagens obtidas por ressonância magnética. O problema de reconhecimento é tratado como a determinação da correspondência ótima entre grafos que representam a imagem a ser reconhecida e um modelo conhecido (atlas).

Propõe-se uma formulação original em variáveis 0-1 para o problema de correspondência ótima de grafos. Diferentes funções objetivo são propostas e investigadas.

Devido à complexidade do problema, diversos algoritmos aproximados são propostos para sua resolução. O primeiro é uma heurística construtiva randomizada que gera soluções viáveis. A segunda heurística é um procedimento de busca local utilizando dois tipos distintos de vizinhanças, aplicado às soluções construídas pelo algoritmo randomizado. A terceira e mais elaborada heurística é uma implementação da metaheurística GRASP, combinando os dois procedimentos anteriores de construção e de busca local. São descritos experimentos computacionais sobre 36 problemas teste, definidos por 12 pares de grafos imagem-atlas e três variantes da função objetivo escolhida. Mostra-se que a heurística GRASP encontra boas soluções subótimas para problemas associados a grafos com centenas de nós e milhares de arestas.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

HEURISTICS FOR SCENE RECOGNITION BY GRAPH MATCHING

Maria Claudia Silva Boeres

September/2002

Advisors: Celso Carneiro Ribeiro

Nair Maria Maia de Abreu

Department: Production Engineering

We consider in this thesis the problem of scene recognition in complex images. The motivation for studying this problem comes from an application in the analysis of medical images. In this application, we want to recognize 3-dimensional structures in the human brain, using images obtained by magnetic resonance. The recognition problem is treated as the determination of the best correspondence between two graphs representing the image to be recognized and a known model (i.e., an atlas).

We propose an original formulation in 0-1 variables for optimal graph matching. Several objective functions are also proposed and evaluated. Due to the computational complexity of the graph matching problem, a sequence of approximation algorithms is proposed for its solution. The first is a randomized construction heuristic which generates feasible solutions. The second heuristic is a local search procedure using two different neighborhood structures, which is applied to the solutions built by the randomized construction algorithm. The third and more elaborate heuristic is an implementation of the GRASP metaheuristic, combining the previously mentioned construction and local search procedures.

We report the results of computational experiments for 36 test problems made up of 12 pairs of image-atlas graphs and three variants of the selected objective function. We show that the GRASP heuristic finds good suboptimal solutions for problems associated with graphs of up to a few hundreds of nodes and a few thousands of edges.

Índice

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Muitas das questões práticas que aparecem em várias áreas das ciências exatas correspondem a problemas combinatórios difíceis em termos computacionais. Diante da dificuldade de obtenção de soluções exatas para problemas dessas classes, tem se intensificado bastante o estudo de vários tipos de algoritmos aproximados para sua resolução.

Apesar do sucesso crescente desses métodos na solução de problemas do mundo real, tornou-se notória a dificuldade de se criar heurísticas de caráter geral que sejam eficientes na solução de classes mais amplas de problemas. Visando a diminuir essa dificuldade, as metaheurísticas consistem em abordagens genéricas que permitem a utilização de informações específicas do problema tratado no sentido de melhorar a qualidade da solução encontrada.

As metaheurísticas têm sido uma ferramenta amplamente utilizada para resolver vários problemas modelados a partir de aplicações do mundo real, como por exemplo, o problema do caixeiro viajante, *bin packing*, roteamento de veículos, alocação quadrática, escalonamento de tarefas e aplicações na área de reconhecimento de imagens, como a análise de cenas complexas (aquelas nas quais é necessária a identificação de muitos objetos).

Problemas de reconhecimento de cenas complexas requerem normalmente uma boa caracterização não apenas dos objetos envolvidos, mas também das relações espaciais existentes entre eles. Em geral, esse tipo de problema é difícil de resolver, devido à grande variabilidade de formas de um mesmo objeto que aparece em diferentes instanciações de uma cena e também devido às similaridades entre diferentes objetos existentes em uma mesma cena, dificultando a sua discriminação. Portanto,

são necessários uma ferramenta apropriada para descrever características relevantes de uma cena e um formalismo preciso para auxiliar o processo de reconhecimento [37, 42].

Modelos baseados em grafos são frequentemente utilizados para representar cenas em processamento de imagens [13, 19, 43, 68, 69]. Em geral, vértices desses grafos representam objetos e arestas (ou arcos) representam relações existentes entre esses objetos. A informação relevante para o processo de reconhecimento pode ser extraída da cena e representada através de grafos relacionais de atributos (GRA). A estrutura da cena é mapeada pelo grafo e os atributos representam características da mesma, importantes para o reconhecimento.

A motivação para estudar esse tipo de problema veio de uma aplicação na área de reconhecimento de imagens médicas. Nesta aplicação, pretende-se efetuar o reconhecimento de estruturas 3D do cérebro humano, a partir de imagens em ressonância magnética (IRM) que tenham sido previamente processadas por algum método de segmentação automática. O processo de reconhecimento se baseia na busca da exata correspondência estrutural entre a imagem e um modelo genérico, tipicamente definido como um atlas (por exemplo, uma IRM representando uma estrutura cerebral tridimensional, segmentada manualmente em regiões bem definidas por radiologistas).

O problema considerado neste trabalho é aquele do reconhecimento estrutural[1] de objetos de uma cena complexa usando um modelo complexo[2]. O esquema de especificação do problema foi originalmente proposto por Perchant [49] e é apresentado na Figura 1.1. Ambas imagens (o atlas e a imagem a ser reconhecida) são representadas através de grafos relacionais de atributos difusos (GRAD). Rótulos são utilizados na representação de regiões, tanto da imagem segmentada quanto do atlas. Cada região no modelo é identificada por exatamente um rótulo. Já na imagem segmentada, os rótulos devem ser encontrados. Neste caso, um mesmo rótulo pode ser associado a várias regiões. Imprecisões e incertezas gerados pelos métodos de segmentação podem ser descritos através do uso de conceitos de incerteza na representação. A interpretação da imagem com respeito ao modelo é realizada através

[1] O processo de reconhecimento tem seu enfoque na identificação da estrutura dos objetos da cena ao invés de seus detalhes, eventualmente presentes na imagem. Em imagens do cérebro, por exemplo, o objetivo é a classificação de cada região da imagem (de acordo com o atlas) e não a identificação de aspectos específicos de cada região.

[2] Cenas com mais de 30 objetos.

da correspondência entre os grafos. Problemas de isomorfismo entre grafos ou subgrafos são frequentemente utilizados para resolver problemas de reconhecimento de imagens [18, 19, 41]. No entanto, a interpretação de imagens que sofreram uma sub ou supersegmentação necessita de um formalismo que permita a correspondência de uma única região do modelo a várias regiões da imagem ou vice-versa. O formalismo proposto em [50] contempla esse aspecto. Os GRAD que representam o modelo e a imagem são comparados por meio desse formalismo, que permite várias formas de correspondência (*matching*).

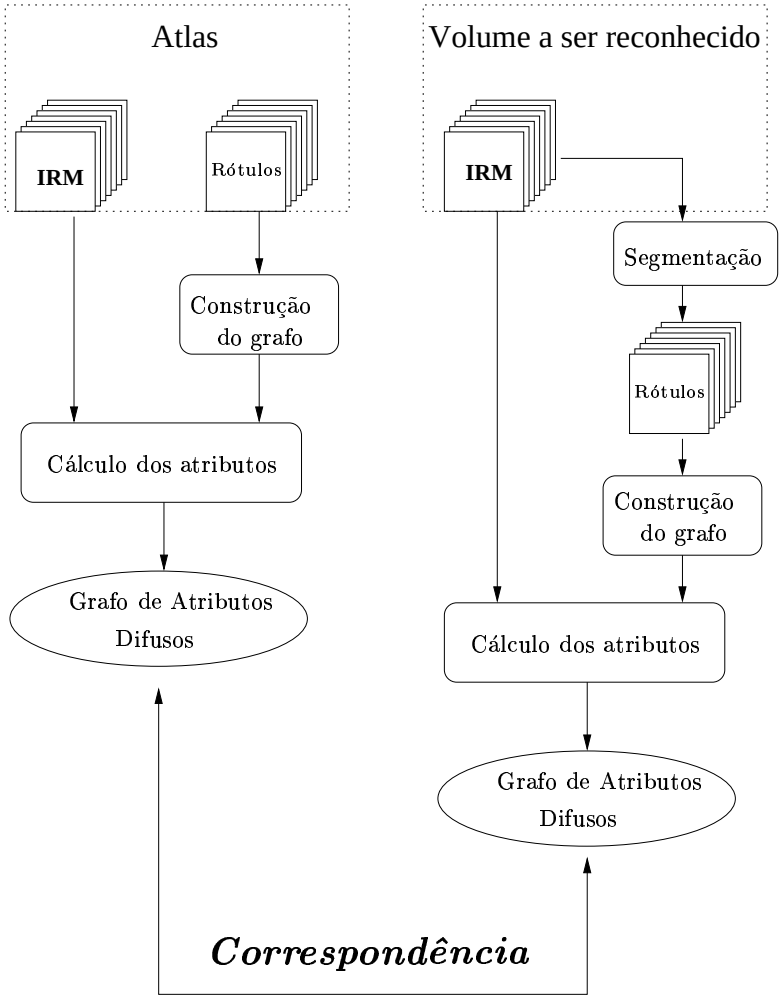


Figura 1.1: O esquema completo de especificação do problema

O problema tratado nesta tese é aquele da determinação da correspondência entre o modelo e a imagem que melhor represente o reconhecimento dessa última. Denota-se esse problema como o Problema de Correspondência de Grafos de Atributos Difusos (PCGAD), que é formulado como um problema de otimização combinatória. Propõe-se uma função objetivo que leva em consideração as similaridades existentes

entre o modelo e a imagem e um conjunto de restrições impostas sobre o espaço de soluções. São propostas e comparadas ferramentas para auxiliar a busca da correspondência mais adequada.

No próximo capítulo é apresentada uma revisão bibliográfica dos trabalhos relativos a problemas de reconhecimento de imagens modelados por grafos e das técnicas de correspondência utilizadas para efetuar esse reconhecimento. Também apresenta-se a descrição do problema considerado como motivação para essa tese, técnicas de construção dos grafos para representação das imagens e o formalismo para a correspondência entre grafos proposto em [50]. A formulação do Problema de Correspondência de Grafos de Atributos Difusos (PCGAD) como um problema de otimização combinatória é proposta no Capítulo 3. Com o objetivo de explorar o espaço de soluções do PCGAD na busca daquela que represente a melhor correspondência entre os grafos, foram propostos vários algoritmos. O algoritmo construtivo descrito no Capítulo 4 gera uma solução viável para o PCGAD, ou seja, uma solução que atenda ao conjunto de restrições. Algoritmos de busca local são apresentados no Capítulo 5. Duas vizinhanças distintas são definidas e utilizadas, comparando-se os resultados obtidos pela aplicação do algoritmo de busca local nestas vizinhanças. No Capítulo 6, os algoritmos construtivo e de busca local são combinados em uma heurística do tipo GRASP. Testes sobre doze problemas teste são realizados e os resultados obtidos são analisados. Conclusões são apresentadas no último capítulo.

Capítulo 2

Revisão bibliográfica

2.1 Introdução

Várias são as abordagens de interpretação de uma cena encontradas na literatura de reconhecimento de imagens. Todas, no entanto, utilizam uma estrutura de representação com a qual se possa descrever adequadamente as características da cena necessárias ao seu reconhecimento. A descrição minuciosa da cena possui maior ou menor relevância de acordo com a aplicação. Normalmente enfatiza-se sua descrição estrutural, abstraindo-se dos detalhes que podem eventualmente sobrecarregar o processo de reconhecimento. Grafos são estruturas matemáticas adequadas para a descrição estrutural de cenas. O problema de reconhecimento pode ser formulado como um problema de correspondência de grafos. A expressão *graph matching* (correspondência de grafos) se refere tipicamente à comparação de dois grafos entre si [9].

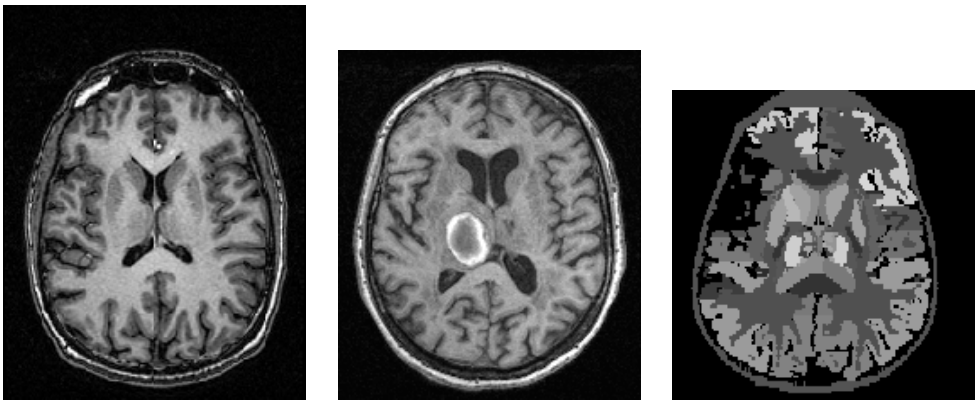
A estrutura dos objetos na cena pode ser primordial para o processo de reconhecimento de imagens. Entretanto, esta estrutura não é única e diversos modelos estruturais podem ser extraídos da cena. Uma escolha natural para representação de cenas é a utilização de grafos de atributos ou grafos de atributos difusos. Nestes grafos, cada vértice representa um objeto da cena e cada aresta representa as relações espaciais existentes entre eles. As características dos objetos e de suas relações são descritas por atributos. A escolha apropriada dos atributos é altamente dependente do problema. Grafos podem também ser utilizados para a representação individual de objetos extraídos de uma cena através de métodos de segmentação. Neste caso, o reconhecimento acontece através da comparação de grafos que representam cada objeto e seus vários modelos ou protótipos.

Imprecisões nas imagens a serem reconhecidas representam uma das principais dificuldades do problema. Os objetos na cena são segmentados e todas as dificuldades advindas dessa segmentação se refletem na correspondência com o modelo. Algumas destas dificuldades são a subsegmentação da imagem (o número de regiões da imagem segmentada automaticamente é menor do que o número de regiões do modelo) ou sua supersegmentação (o número de regiões da imagem segmentada é em geral bem maior do que o número de regiões do modelo), objetos inesperados encontrados na cena, objetos esperados mas não encontrados, grandes deformações dos objetos e o desaparecimento de objetos que estavam presentes na cena. Como trata-se do reconhecimento de cenas complexas, é impossível evitar todas estas dificuldades. Por exemplo, no reconhecimento de imagens aéreas, a identificação de estradas pode ser dificultada pela presença de nuvens ou árvores. A calibração inadequada do foco da câmera (imagem turva) pode facilitar a identificação errônea de objetos ou mesmo modificar atributos, como a largura da estrada. Além disso, com uma segmentação mal feita, é possível que algum segmento da estrada seja omitido da imagem [39]. Problemas similares ocorrem em outras aplicações.

O próprio modelo da cena é também freqüentemente impreciso. Ele pode estar incompleto ou os atributos dos objetos podem ser imprecisos. Além disso, dificuldades podem surgir da possível generalidade do modelo escolhido. Por exemplo, no processo de reconhecimento de imagens cerebrais, o primeiro passo consiste em segmentar estruturas anatômicas do cérebro a partir de uma imagem de ressonância magnética (IRM). Se todos os cérebros humanos fossem anatomicamente similares, um modelo genérico poderia ser escolhido. Entretanto, sabe-se que estruturas cerebrais possuem uma grande variabilidade entre indivíduos, de formas e de posicionamento. Por esse motivo, estes tipos de informação não são adequadamente descritos se forem definidos como atributos exatos. Se, além disso, a imagem a ser reconhecida é de um cérebro patológico, então a patologia é considerada como um objeto adicional na estrutura cerebral. Esta patologia pode gerar deformações das estruturas sãs ou mesmo aparecer no lugar de um outro objeto que deveria existir na imagem. Para ilustrar estas dificuldades, a Figura 2.1 apresenta um corte axial de três volumes cerebrais diferentes: (a) um cérebro normal, (b) um cérebro patológico (com um tumor), e (c) a representação de um atlas do cérebro onde cada tom de cinza corresponde a uma única estrutura conexa. Observa-se que as estruturas em tons mais escuros existentes no centro da figura (os ventrículos laterais) são muito

maiores em (b) do que em (a). A estrutura em tom branco (tumor) não está presente nem no atlas (c), nem no cérebro são (a).

O reconhecimento de imagens de ressonância magnética do cérebro humano se constitui na motivação desta tese. Alguns algoritmos de reconhecimento propostos são descritos em [24, 25, 36, 37, 50, 52, 55].



(a) (b) (c)

Figura 2.1: Exemplos de IRM: (a) corte axial de um cérebro são, (b) corte axial de um cérebro patológico com tumor, e (c) corte axial de um atlas do cérebro.

Outro exemplo de aplicação do problema de correspondência de grafos consiste na interpretação e descrição de imagens já conhecidas. Neste caso, o objetivo é fornecer uma descrição o mais fiel possível da realidade. Esse tipo de problema é estudado em [42, 43]. Deseja-se propor a melhor descrição possível da imagem aérea de uma região urbana de Paris, baseando-se no mapa da região. O processo de interpretação é composto de várias etapas. Em um primeiro momento, um mapa é comparado à imagem. A partir daí, uma estrutura de estradas é construída, obedecendo certos graus de confiança obtidos pelas informações provindas da imagem. Detecção de concentrações de prédios ou de áreas descampadas (com determinados graus de confiança) podem auxiliar bastante a descrição da imagem estudada. Nos dois casos citados acima, percebe-se que a representação adequada da imagem estudada é fundamental para sua análise. Outros exemplos de aplicações nesta área encontram-se em [5, 39].

Em um grande número de problemas de reconhecimento de imagens, observa-se que imprecisões decorrem tanto da forma como a representação da cena e do modelo são construídos, quanto da correspondência estabelecida entre essas duas estruturas, que normalmente não é um mapeamento biunívoco (isto é, um isomorfismo). O

primeiro tipo de imprecisões pode ser tratado por exemplo, através da definição de atributos difusos para descrever os objetos e suas relações, e da construção dos grafos baseada nas características estruturais do problema. O segundo tipo diz respeito ao mapeamento entre duas estruturas possivelmente diferentes, consistindo no problema de correspondência não exata de grafos. Neste caso, o problema pode ser resolvido permitindo-se a correspondência não biunívoca entre os grafos comparados [50] ou através da utilização de operações de edição dos mesmos a fim de transformar o problema em isomorfismo de grafos ou de subgrafos [46, 58, 62].

Desta forma, pode se encontrar várias abordagens de tratamento do problema de correspondência de grafos na literatura de reconhecimento de imagens. Dentre elas, pode-se distinguir alguns tipos principais de metodologia: (1) divisão do processo de reconhecimento em duas etapas, efetuando-se o reconhecimento global de toda a estrutura da cena e identificando-se posteriormente cada um dos objetos individualmente, ou (2) realizando-se correspondências locais e usando-as para o posterior reconhecimento da estrutura global. Pode-se também combinar a busca de correspondências entre objetos ao mesmo tempo em que se tenta efetuar o reconhecimento da estrutura global de uma imagem.

2.2 Tipos de grafos utilizados para a representação de imagens

Existem diversos tipos de grafos que podem ser utilizados para modelar imagens ou objetos presentes em imagens. Definições e propriedades desses tipos de grafos foram apresentadas em [49]. Algumas das definições mais utilizadas na literatura de reconhecimento de imagens modeladas por grafos são resumidas nesta seção.

Definição 2.1 *Um grafo é uma estrutura $G = (N, E)$ onde N é um conjunto finito de vértices e $E \subseteq N \times N$, um conjunto finito de arcos (se são pares ordenados) ou arestas (caso contrário). As cardinalidades dos conjuntos N e E são representadas respectivamente por $|N|$ e $|E|$.*

Definição 2.2 *Um grafo relacional (GR) é um grafo $G = (N, E)$ com um peso p_{ij} associado a cada arco (ou aresta) $(i, j) \in E$.*

Definição 2.3 *Um grafo difuso (GD)* $G = (\sigma, \mu)$ *sobre um conjunto* N *de vértices é um par de aplicações* $\sigma : N \to [0,1]$ *e* $\mu : N \times N \to [0,1]$, *tais que* $\mu(u,v) \leq min\{\sigma(u), \sigma(v)\}$, $\forall (u,v) \in N \times N$.

Definição 2.4 *Seja* $Z = \{z_i\}, i = 1, \ldots, I$ *um conjunto de atributos. Cada atributo* z_i *pode tomar um valor definido em* $S_i = \{s_{ij}, j = 1, \ldots, J_i\}$. *O conjunto* $L_v = \{(z_i, s_{ij}), j = 1, \ldots, J_i; i = 1, \ldots, I\}$ *representa todas as combinações atributo-valor possíveis. Uma configuração possível de atributos associados a um vértice é um subconjunto de* L_v *onde cada atributo aparece apenas uma vez. De maneira análoga, são definidos respectivamente os conjuntos* Z^a, S_i^a *e* L_a *para os arcos ou arestas do grafo. Define-se* Π *como o conjunto das configurações possíveis para os vértices e* Θ *como o conjunto das configurações possíveis para os arcos ou arestas do grafo.*

Um grafo relacional de atributos (GRA) definido sobre $L = (L_v, L_a)$ *com uma estrutura de grafo* $G = (N, E)$ *é um par ordenado* (V, A), *onde* $V = (N, \sigma)$ *é chamado de conjunto de atributos dos vértices e* $A = (E, \delta)$ *é chamado de conjunto de atributos das arestas ou arcos. A aplicação* $\sigma : N \to \Pi$ *é o interpretador de vértice e a aplicação* $\delta : E \to \Theta$ *é o interpretador de arco ou aresta.*

Definição 2.5 *Seja* $Z = \{z_i\}, i = 1, \ldots, I$ *um conjunto de atributos. Cada atributo* z_i *pode tomar um valor definido em* $S_i = \{s_{ij}, j = 1, \ldots, J_i\}$. *O conjunto* $\tilde{L}_v = \{(z_i, \tilde{A}_{S_i}), i = 1, \ldots, I\}$ *representa todas as combinações atributo-valor difuso possíveis.* $\tilde{A}_{S_i}$ *é um conjunto difuso sobre o conjunto de valores de atributos* S_i. *Uma configuração possível de atributos associados a um vértice é um subconjunto de* $\tilde{L}_v$ *onde cada atributo aparece apenas uma vez. De maneira análoga, são definidos respectivamente os conjuntos* Z^a, S_i^a, $\tilde{A}_{S_i^a}$ *e* $\tilde{L}_a$ *para os arcos ou arestas do grafo. Define-se* $\tilde{\Pi}$ *como o conjunto das configurações possíveis para os vértices e* $\tilde{\Theta}$ *como o conjunto das configurações possíveis para os arcos ou arestas do grafo.*

Um grafo relacional de atributos difusos (GRAD) definido sobre $\tilde{L} = (\tilde{L}_v, \tilde{L}_a)$ *com uma estrutura de grafo* $G = (N, E)$ *é um par ordenado* $(\tilde{V}, \tilde{A})$, *onde* $\tilde{V} = (N, \tilde{\sigma})$ *é chamado de conjunto de atributos difusos dos vértices e* $\tilde{A} = (E, \tilde{\delta})$ *é chamado de conjunto de atributos difusos das arestas ou arcos. A aplicação* $\tilde{\sigma} : N \to \tilde{\Pi}$ *é o interpretador de vértice e a aplicação* $\tilde{\delta} : E \to \tilde{\Theta}$ *é o interpretador de arco ou aresta.*

Todas essas famílias de grafos compartilham algumas propriedades, o que permite que relações possam ser definidas entre elas. Por exemplo, a definição de um grafo de atributos difusos pode ser obtida a partir das definições de grafo difuso e de grafo relacional de atributos. Relações entre famílias de grafos podem ser representadas pela própria estrutura de um grafo orientado. Esse tipo de informação é útil, por exemplo, para se determinar para quais famílias de grafos um determinado algoritmo pode ser aplicado ou, ainda, para definir novos tipos de grafos, eventualmente necessários para representar particularidades de uma determinada aplicação, criando-se novas relações a partir das existentes [49].

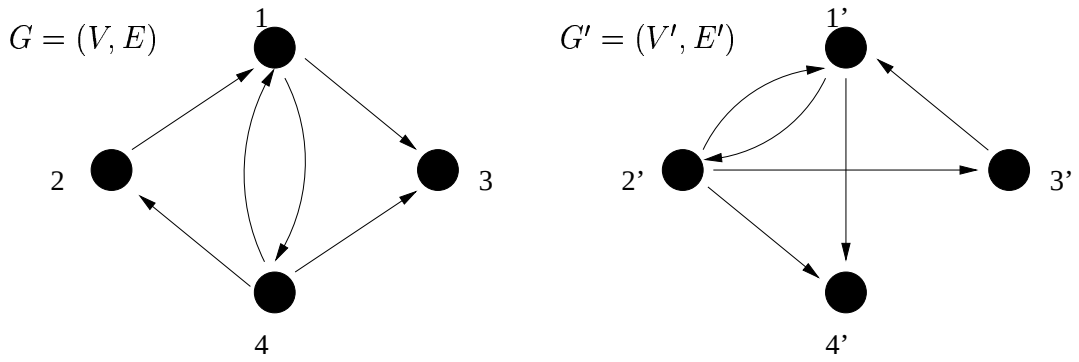
2.3 Correspondência de grafos

Além de se definir a metodologia de reconhecimento, dois tipos de correspondência de grafos podem ser estabelecidos. A correspondência exata de grafos ou de subgrafos trata aplicações onde as estruturas foram descritas precisamente. Esse tipo de problema pode ser modelado por isomorfismo de grafos ou de subgrafos. Um isomorfismo consiste em um mapeamento bijetivo entre vértices dos grafos comparados, de maneira que a estrutura das arestas seja preservada. Essa mesma idéia pode ser utilizada para resolver o problema de isomorfismo de subgrafos, considerando-se neste caso a comparação do tipo grafo-subgrafo. A noção de isomorfismo de grafos é frequentemente usada na literatura de reconhecimento de imagens, como uma forma de estabelecer uma correspondência bijetiva entre os componentes dos grafos que representam a imagem a ser reconhecida e o modelo ou modelos aos quais ela será comparada. As definições de isomorfismo de grafos ou de subgrafos [3, 49] são apresentadas neste capítulo.

Definição 2.6 *Os grafos não direcionados* $G = (V, E)$ *e* $G' = (V', E')$ *são isomorfos se e somente se* $|V| = |V'|$ *e existir uma função unívoca* $f : V \rightarrow V'$*, tal que* $\{u, v\} \in E$ *se e somente se* $\{f(u), f(v)\} \in E'$*. Se* G *e* G' *são direcionados, então eles são isomorfos se e somente se existir uma função unívoca* $f : V \rightarrow V'$*, tal que* $(u, v) \in E$ *se e somente se* $(f(u), f(v)) \in E'$*.*

Os grafos da Figura 2.2 são isomorfos.

Em situações frequentes, deseja-se identificar objetos presentes em uma imagem. Por exemplo, em uma imagem de um bairro de uma cidade, obtida via satélite,



$f(1) = 1',\ f(2) = 3',\ f(3) = 4',\ f(4) = 2'$

$|V| = |V'| = \{1, 2, 3, 4\}$

$E = \{(1,3), (1,4), (2,1), (4,1), (4,2), (4,3)\}$

$E' = \{(1',4'), (1',2'), (3',1'), (2',1'), (2',3'), (2',4')\}$

Figura 2.2: G é isomorfo a G'

deseja-se identificar um conjunto de ruas representadas através de um mapa. Se tanto a imagem quanto um modelo do objeto forem descritos por grafos, então o problema de identificação do objeto equivale a encontrar um subgrafo do grafo que representa a imagem e que coincida com o modelo.

Definição 2.7 *Dados os grafos não direcionados $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$, diz-se que G_1 contém um* subgrafo isomorfo *a G_2 se e somente se existem um subconjunto $V \subseteq V_1$ e um subconjunto $E \subseteq E_1$ satisfazendo $|V| = |V_2|$ e $|E| = |E_2|$ e uma função unívoca $f : V_2 \to V$ tal que $\{u, v\} \in E_2$ se e somente se $\{f(u), f(v)\} \in E$. Se G_1 e G_2 são direcionados, então a função unívoca $f : V_2 \to V$ é tal que $(u, v) \in E_2$ se e somente se $(f(u), f(v)) \in E$.*

O subgrafo G de G_1 é isomorfo ao grafo G_2, na Figura 2.3. Pode-se também estabelecer isomorfismos entre subgrafos de dois grafos.

Esse tipo de abordagem por isomorfismos é adequado para situações onde correspondências precisas entre os grafos são facilmente estabelecidas. No entanto, na maioria dos casos, um grande esforço de edição de grafos é necessário, devido aos erros advindos do processo de segmentação da imagem ou mesmo devido ao uso de vários modelos de grafo para representar cada aspecto relevante de um objeto. Se mais de um modelo é viável, então eles deveriam ser suficientemente precisos para serem detectados na imagem.

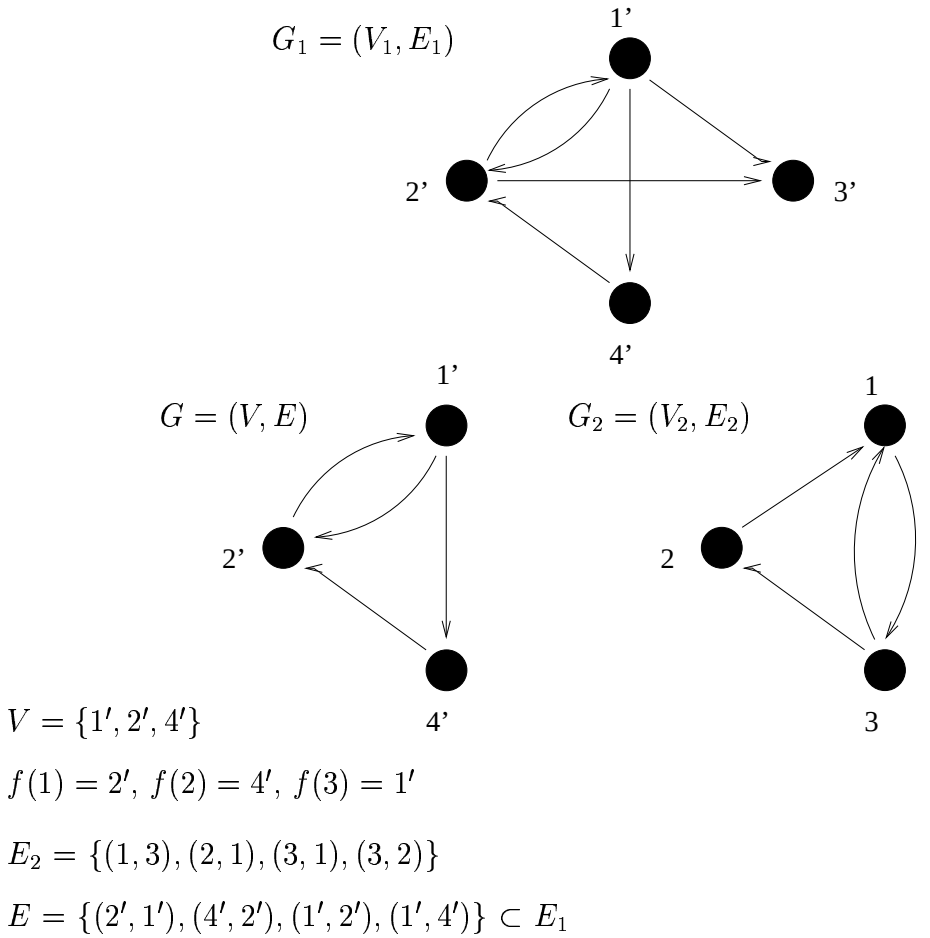


Figura 2.3: O subgrafo G de G_1 é isomorfo a G_2

A correspondência inexata de grafos modela aplicações de reconhecimento de imagens com distorções. Esse tipo de modelagem é o mais frequentemente encontrado na literatura de reconhecimento de imagens. O reconhecimento de imagens com distorções também pode ser modelado por isomorfismo de grafos ou de subgrafos. Neste caso, as distorções, geralmente dependentes do problema, são também modeladas através de operações de edição de grafos, como eliminação, inserção e substituição de nós ou arestas. A quantidade de operações de edição de grafos pode ser usada como uma medida de similaridade entre os grafos (quanto menor for a seqüência de operações de edição necessárias para a transformação de um grafo no outro, mais similares eles são).

2.4 Abordagens de solução do problema de correspondência de grafos

Basicamente, dois tipos de utilização do problema de correspondência de grafos são encontrados na literatura de reconhecimento de imagens. Um é o problema de se identificar a melhor correspondência entre o grafo $G_2 = (N_2, E_2)$ que representa uma imagem a ser reconhecida e o grafo $G_1 = (N_1, E_1)$ que representa um modelo dessa imagem. O outro é aquele em que se deseja determinar o melhor representante para um objeto a ser identificado, dentre todos os modelos disponíveis desse objeto. Tanto o objeto quanto os modelos são descritos por grafos relacionais de atributos (GRA), grafos relacionais de atributos difusos (GRAD) ou descrições relacionais [61]. As principais dificuldades para resolver o problema de correspondência de grafos são (1) como calcular a distância ou medida de similaridade entre descrições relacionais ou grafos, ou seja, como calcular o grau de semelhança existente entre os objetos representados por essas estruturas e (2) como procurar de forma eficiente a melhor correspondência entre eles.

Várias são as técnicas propostas na literatura para o cálculo de medidas de similaridade. Algumas prevêem a remoção ou a inserção de nós dos grafos (acarretando na atualização das arestas envolvidas). É certamente desejável que essas operações de edição do grafo sejam realizadas com o menor custo possível. Esses cálculos podem ser efetuados através da definição de funções que consideram apenas a manipulação de nós ou arestas, com as quais se possa definir o conceito de *distância* de um grafo ao outro [19, 58, 60, 61, 65, 67, 68, 69], ou utilizando-se modelos probabilísticos, que são capazes de representar e manipular os numerosos aspectos de imperfeição da informação. Estes modelos são normalmente utilizados no tratamento de baixo nível (*pixels*, por exemplo) para modelar a imperfeição dos dados, ou mesmo de alto nível (semântico), para controlar os processos de decisão [14, 18, 48].

O uso de sistemas difusos na modelagem do problema de correspondência de grafos é também muito utilizado. Uma das vantagens desse tipo de abordagem é a maior facilidade em se descrever as imperfeições e distorções presentes nas imagens a serem reconhecidas. Além da descrição de características das imagens utilizando-se sistemas difusos, estes também são usados no cálculo de medidas de similaridade [12, 49, 50, 51, 53, 63].

2.4.1 Abordagens por manipulação de grafos ou descrições relacionais

Sanfeliu [58] apresenta um método para determinar uma medida de similaridade entre dois grafos relacionais de atributos, caracterizados por gramáticas descritivas. A medida é baseada no cálculo do número mínimo de operações de edição para transformar o grafo G_2 no grafo G_1 ao qual ele está sendo comparado.

Sonbaty e Ismail [19] propõem um algoritmo para resolução do problema de isomorfismo de subgrafos, cuja estratégia tem como principal característica a decomposição de grafos relacionais de atributos em subgrafos menores. O processo de correspondência ocorre entre os subgrafos gerados.

Blake [4] propõe uma estratégia de resolução para o problema de correspondência de grafos ou subgrafos através de seu particionamento em subproblemas. O autor propõe o uso da teoria dos reticulados como base de definição da metodologia proposta. Define-se uma ordenação parcial para três tipos diferentes de dados: grafo (definindo-se o conceito de dois grafos comparáveis), tabela de valores de associação (que contém elementos que definem uma associação entre dois grafos) e restrições (que representam propriedades dos objetos envolvidos). Desta forma, o autor define uma correspondência entre dois grafos como um algoritmo para determinar uma associação entre eles que seja compatível com as restrições definidas e com o menor custo possível. Um exemplo de aplicação do método é apresentado.

You, Chan e Wong [68] propõem a utilização da técnica de *branch-and-bound* [32] para resolver o problema de isomorfismo de subgrafos.

Um problema muito estudado na área de visão é o processo de reconhecimento de objetos por um robo. A estratégia usual é promover o reconhecimento desses objetos a partir de imagens digitalizadas ou de projeções bi-dimensionais dos mesmos. Várias são as propostas para reconhecimento de objetos 3D. Wong [69] descreve um método baseado no problema de correspondência grafo-subgrafo. Os objetos são modelados como grafos relacionais de atributos, através de uma correspondência direta: os vértices do grafo correspondem aos vértices da superfície e as arestas correspondem às arestas entre as faces do objeto. Os atributos associados a cada vértice do grafo correspondem a um conjunto de tipos de junções entre as faces de um objeto e as possíveis combinações a partir destes, referentes aos nós vizinhos. É

apresentado um conjunto de classes genéricas de junções entre faces de um objeto.

Em [10] objetos bidimensionais da cena a ser reconhecida são representados pelos seus contornos. O processo de reconhecimento é baseado na definição de árvores onde cada nó corresponde a um segmento do contorno do objeto (convexo, côncavo ou reto) e um arco conecta nós que estão em sucessivos níveis de resolução da imagem. Neste caso, o problema de comparar um objeto com o seu modelo consiste em determinar o melhor mapeamento entre nós em todos os níveis das árvores que representam o objeto e o modelo. O algoritmo é baseado em programação dinâmica [32]. Os autores afirmam que o método é robusto, pois a correspondência perfeita é obtida nos testes realizados, mesmo na presença de distorções da imagem.

Bunke e Messmer [9, 40, 41] propõem um método de decomposição de grafos para tratar o problema de isomorfismo de subgrafos. O método proposto é estendido para o problema de correspondência inexata de grafos. O problema considerado é o de encontrar isomorfismos entre grafos G_1 que representam modelos ou protótipos conhecidos *a priori*, de objetos que pertencem a uma imagem representada por um grafo G_2 dado como entrada. Normalmente o reconhecimento dos objetos é realizado sequencialmente, ou seja, verifica-se a correspondência de G_2 com cada um dos grafos G_1 que representam os modelos. O número de modelos existentes influenciará na eficiência dessa estratégia. Além disso, se existem subgrafos de G_2 que se encontram repetidamente em vários dos modelos, o processo de reconhecimento se torna ainda mais lento, porque esses subgrafos serão identificados em vários modelos desnecessariamente. Propõe-se um pré-processamento dos grafos modelos para reduzir o esforço computacional, decompondo-os em subgrafos menores.

A necessidade de estabelecimento de relações envolvendo mais de dois elementos, ou da definição de mais de uma relação entre dois elementos de um conjunto de componentes de um objeto, motivou Shapiro e Haralick a propor uma descrição para objetos mais geral do que os grafos relacionais, definida precisamente e denotada em [61] por uma descrição relacional (D). Intuitivamente, se X é um conjunto de componentes de um objeto, uma descrição relacional D_X permite a definição de vários tipos de relações R entre esses componentes. Esse tipo de descrição pode ser utilizado por exemplo para facilitar o reconhecimento de caracteres chineses. Os autores propõem uma medida de similaridade para descrições relacionais.

O problema de organização de grandes conjuntos de modelos de um mesmo

objeto é discutido em [60, 61]. As técnicas utilizadas se baseiam na organização do conjunto de modelos em grupos cujos elementos possuam similaridades entre si. Cada grupo possui um elemento representativo com o qual um objeto é comparado, reduzindo-se o número de comparações realizadas. Outro método de organização de conjuntos de modelos, através de árvores binárias, também é apresentado.

2.4.2 Abordagens por técnicas de inteligência artificial e de redes neuronais

Abordagens por técnicas de inteligência artificial e de redes neuronais são encontradas na literatura de reconhecimento de imagens. Uma rede neuronal pode "aprender" as estruturas dos modelos de imagens que se quer reconhecer e classificar as entradas segundo esse aprendizado. Através das técnicas de inteligência artificial, é possível desenvolver um modelo conceitual do que se quer reconhecer e propor o reconhecimento da cena utilizando-se sistemas especialistas [8]. Por exemplo, em [48] é proposto um ambiente que combina a utilização de grafos com um sistema especialista para aprender características importantes do modelo G_1 e produzir regras para calcular pesos para essas características de acordo com o número de suas ocorrências em G_2. Neste trabalho, o algoritmo A^* [47] é utilizado para encontrar a melhor correspondência entre G_1 e G_2.

2.4.3 Abordagens por modelos probabilísticos

Com o objetivo de escapar da convergência prematura para ótimos locais, característica das técnicas discretas de resolução do problema de correspondência de grafos, foram propostos novos algoritmos de resolução que podem ser classificados em duas grandes categorias: algoritmos de busca que incorporam elementos estocásticos no seu processo de atualização de solução a fim de superar ótimos locais, como os algoritmos genéticos [29, 31] e *simulated annealing* [33], e algoritmos que atuam em espaços contínuos de solução, como *mean-field annealing* [70].

Métodos de relaxação [26, 30], considerados boas estratégias para o tratamento de correspondência inexata de grafos devido à sua capacidade de identificar interpretações simbólicas consistentes a partir de dados de entrada com distorções, podem incorporar componentes probabilísticas. Em métodos de relaxação discreta, interpretações simbólicas consistentes são associadas aos arranjos de vértices de grafos

relacionais através de operações de troca de símbolos. Nos métodos de relaxação probabilística, as correspondências consistentes são localizadas através da atualização de probabilidades de associação. Esses métodos devem ser guiados por uma função que representa uma medida de consistência global que mede a qualidade da correspondência entre os grafos. O objetivo é encontrar a correspondência que otimize essa medida. A dificuldade de definição de uma medida de consistência suficientemente precisa consiste no principal obstáculo para a utilização de algoritmos de relaxação para aplicações reais.

Em [13, 15, 16, 23, 64, 65, 66] foi desenvolvido um extenso trabalho em correspondência de grafos usando modelos probabilísticos. Um ambiente bayesiano para o problema de correspondência de grafos foi proposto para estimar a medida de consistência através de uma distribuição de probabilidade exponencial que usa a distância de Hamming para medir as diferenças estruturais entre os grafos comparados. Essa medida [13, 65] é utilizada nas diferentes estratégias determinísticas de busca investigadas em [64]. Foi proposta também uma estratégia de busca tabu [27, 28], definindo-se um critério para estabelecer uma ordem de prioridade baseada na qualidade das correspondências estabelecidas entre os grafos.

A fim de superar falhas do processo de atualização determinístico, foram desenvolvidas duas estratégias de otimização estocástica, cujas funções objetivo se baseiam na medida de consistência relacional [13]. Os métodos propostos foram *simulated annealing* e algoritmos genéticos. Na primeira técnica, o processo de relaxação discreta é representado por uma distribuição de Boltzmann. Na segunda técnica utilizada, a medida de consistência global é usada como a função de adequação do algoritmo [14, 17, 18]. Os autores mostram que a técnica proposta consegue identificar precisamente modelos de vários objetos que ocorrem na imagem a ser reconhecida eventualmente com alguma interseção. Em [16] é apresentada uma análise de convergência do algoritmo apresentado em [18], mostrando-se ainda que o algoritmo genético tem sua convergência acelerada quando um algoritmo guloso determinístico é utilizado como um operador adicional aos clássicos. Exemplos de aplicação do método em imagens reais são apresentados.

Myers [44, 45] propõe ainda o uso de algoritmos genéticos para resolver o problema de correspondência de grafos com ambigüidades, estendendo a definição da medida de consistência global para tratar esse tipo de problema. A novidade consiste na utilização do princípio de Marr [38], que afirma a importância da manutenção

de várias hipóteses acerca da solução de um problema até que existem evidências suficientes para a escolha de apenas uma delas. Assim, várias interpretações de uma região localmente ambígua podem ser consideradas para enriquecer a informação necessária para a convergência do método de busca utilizado para aquela que será uma solução de boa qualidade. Os autores discutem a utilização de algoritmos genéticos para a resolução do problema de correspondência de grafos com ambigüidades, seguindo o princípio de Marr. No trabalho de Myers, Wilson e Hancock [46] é estudado o problema de como medir a similaridade de grafos com violações estruturais. É proposta uma medida de similaridade baseada naquela proposta em [58], que usa a distância de Levenshtein para modelar a distribuição de probabilidade que representa os erros estruturais no problema de correspondência de grafos. A distância de Levenshtein [35] calcula o menor número de operações de edição para transformar uma cadeia de símbolos em outra.

Outro exemplo de abordagem probabilística é aquele proposto por Bengoetxea et al [2]. Assim como os algoritmos genéticos, *Estimation Distribution Algorithms (EDAs)* são algoritmos de busca que manipulam populações de indivíduos. Cada indivíduo representa um mapeamento entre grafos que descrevem a imagem e o modelo ao qual a imagem é comparada. O indivíduo é descrito por um vetor de dimensão $|N_2|$ (número de vértices do grafo que representa a imagem) de variáveis aleatórias. Cada variável aleatória pode assumir valores inteiros no intervalo $[1, |N_1|]$, onde $|N_1|$ representa a cardinalidade do conjunto de vértices do grafo modelo. Uma população inicial é gerada, assumindo-se uma distribuição uniforme para cada variável. Cada indivíduo dessa população é avaliado de acordo com a função f_1 definida em [50], que mede o grau de similaridade da correspondência. Um conjunto de indivíduos dessa população é selecionado de acordo com algum critério estabelecido (no caso, os indivíduos com maiores valores da função objetivo são selecionados). A seguir, um modelo probabilístico que melhor represente as interdependências entre as variáveis de um individuo é definido. A próxima população não é gerada utilizando-se operadores de *crossover* ou mutação, como nos algoritmos genéticos, mas através da simulação do modelo probabilístico definido a partir dos indivíduos selecionados na iteração anterior. O passo mais importante desse processo é encontrar a interdependência entre as variáveis que representem uma solução. A idéia básica consiste em induzir modelos probabilísticos a partir dos melhores indivíduos selecionados na população gerada na iteração anterior. Uma vez estimado o modelo probabilístico,

ele é amostrado para gerar novas soluções que constituirão a nova população. Foram apresentados três algoritmos para estimar esse modelo: (a) UMDA (não considera interdependências entre variáveis); (b) MIMIC (são consideradas pares de dependências) e (c) EBNA (são consideradas múltiplas interdependências entre as variáveis). Com as informações adquiridas a partir desses algoritmos, uma rede bayesiana [34] é simulada e utilizada como ferramenta para geração da próxima população. Um indivíduo correto consiste em uma solução que satisfaz a três condições: (a) todos os vértices do grafo imagem deve ter pelo menos uma correspondência com um nó no grafo modelo; (b) cada nó no grafo imagem pode ter no máximo um nó casado no grafo modelo e (c) todos os nós no grafo modelo devem ter pelo menos um nó casado no grafo imagem. Técnicas para obtenção de indivíduos corretos são aplicadas. Essas técnicas consistem em controlar o passo de simulação (através da alteração de probabilidades com o objetivo de guiar a geração de indivíduos corretos), correção de indivíduos após o passo de simulação e penalização de indivíduos incorretos. As condições (a) e (b) são automaticamente respeitadas pelo tipo de representação escolhido para os indivíduos. Já a condição (c) precisa ser verificada. Experimentos com instâncias artificiais são realizados.

2.4.4 Abordagens por sistemas difusos

A teoria de conjuntos difusos oferece uma grande variedade de ferramentas que permitem a modelagem de imprecisões e incertezas inerentes ao problema de correspondência de grafos. Na abordagem adotada por Perchant [49], consideram-se aspectos estruturais de uma cena para efetuar o seu reconhecimento comparando-a com um modelo. Mecanismos de modelagem e manipulação das cenas são propostos. A imagem e o modelo são descritos por grafos relacionais de atributos difusos, que permitem construir uma estrutura de dados próxima da estrutura da cena, concentrando toda a informação pertinente nos elementos dos grafos (nós e arestas). A correspondência entre os grafos é realizada através do *morfismo difuso de grafos*, que consiste na associação não biunívoca dos vértices e arestas dos grafos. Um vértice (resp. aresta) de um dos grafos pode ser associado com vários vértices (resp. arestas) do outro grafo, com uma certa intensidade.

A teoria que fundamenta o morfismo difuso de grafos é apresentada em detalhes em [49]. Ela permite modelar os diversos variantes do problema de correspondência de grafos encontrados na literatura, uniformizando-os em um único formalismo que

possibilita de forma simples a incorporação das informações estruturais relevantes para a solução do problema.

Pelos aspectos de generalidade, simplicidade e riqueza de representação, o morfismo difuso de grafos proposto por Perchant foi utilizado nesta tese como base para a formulação do problema de correspondência de grafos de atributos difusos (PCGAD) como um problema de otimização combinatória, apresentado no Capítulo 3. Nas próximas seções serão apresentadas as estratégias de construção dos grafos (que modelam as imagens comparadas) e a descrição simplificada do morfismo difuso de grafos.

2.5 Construção dos grafos

Grafos Relacionais de Atributos Difusos (GRAD) são amplamente usados em problemas de reconhecimento de imagens. A escolha e a implementação dos atributos difusos são definidas de acordo com o problema tratado. Adjacência difusa [49], distância difusa [7] e posição relativa [6] são exemplos de atributos definidos para relações estabelecidas entre objetos presentes em estruturas cerebrais.

Em Perchant [49], os atributos difusos representam as características dos objetos, através de conjuntos difusos. De maneira geral, muitos atributos podem ser extraídos de objetos de uma cena e sua pertinência depende fortemente do problema considerado. Para a aplicação de imagens cerebrais estudada em [49], um atributo utilizado foi o nível de cinza em cada região de uma imagem cerebral. Cada classe de nível de cinza deve corresponder a uma matéria visível sobre a IRM. O número de classes varia em função do tipo de aquisição. Os níveis de cinza presentes em uma imagem são classificados segundo um determinado critério. A partir dessa classificação, foi imposto um modelo de conjuntos difusos para calcular as funções de pertinência a cada classe de nível de cinza (isto é, a cada matéria visível). Cada objeto é então caracterizado por um histograma de seus níveis de cinza, que são normalizados para construir-se uma função de pertinência à região. A etapa final de cálculo do atributo é seu grau de pertinência a cada classe de nível de cinza, calculado sobre o conjunto da imagem. Já os atributos relacionais, eles representam as relações espaciais entre os objetos. Dados dois objetos, deseja-se verificar se eles compartilham um conjunto de relações. Em reconhecimento de imagens, esse tratamento é feito para cada par de objetos. Os atributos de seus vértices e arestas

podem ser calculados em conjunto com a construção de um GRAD. Essa construção depende das imperfeições da cena ou do modelo de referência e dos atributos associados às relações existentes entre os objetos. Convenciona-se que cada região (tanto do modelo quanto da imagem segmentada) é representada por um vértice do grafo. Diferenças ocorrem na criação das arestas, devido aos atributos ou a defeitos na segmentação da cena. Cinco estratégias de construção do conjunto de arestas podem ser usadas quando a cena é complexa (mais de 30 objetos):

1. *Grafo completo*: uma estrutura de grafo completo é criada e os atributos das arestas entre cada par de objetos são computados. Esse tipo de construção pode ser útil em casos de subsegmentação, quando todas as relações entre objetos são relevantes.

2. *Simplificação de um grafo completo*: arestas com atributos não significantes são removidas do grafo. Esse tipo de construção é útil quando a subsegmentação é esparsa na imagem.

3. *Construção local*: é adequada quando algumas das relações entre objetos da cena não são relevantes, existem muitos objetos e a segmentação não gerou defeitos. Adjacência é um exemplo desse tipo de construção. No entanto, em situações como a da Figura 2.4, se a aresta (v_1, v_2) não existe, então a informação de proximidade entre v_1 e v_2 (eventualmente importante para o processo de reconhecimento) não seria representada. Em casos como esse, a noção de distância é útil para a construção do conjunto de arestas.

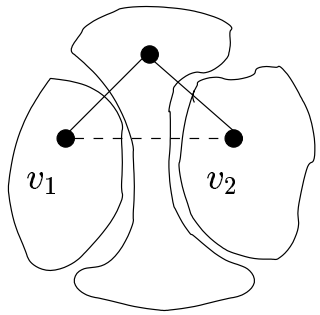


Figura 2.4: Construção de arestas usando distâncias

4. *Fecho transitivo k-local sobre construção local*: é adequado quando uma supersegmentação homogênea é realizada na imagem. O valor de k representa o número de níveis de vizinhança de um vértice do grafo considerados para a

criação de uma nova aresta. Para $k = 1$ essa operação é similar ao de fecho transitivo de um conjunto. Seja inicialmente o grafo G' construído como uma cópia de um dado grafo G. Se as arestas (u, v) e (v, w) pertencem a G, então a aresta (u, w) é adicionada ao grafo G'. As arestas adicionadas em G' a partir de G são representadas na Figura 2.5 como linhas tracejadas. O valor de k é escolhido em função do grau de supersegmentação na imagem.

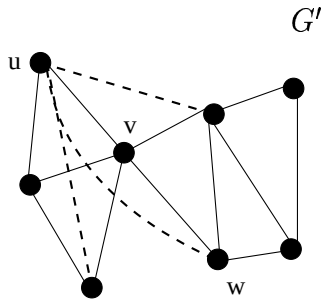


Figura 2.5: Fecho transitivo k-local sobre construção local com $k = 1$

5. *Fecho condicional transitivo k-local sobre construção local*: é adequado quando a supersegmentação realizada na imagem é heterogênea. Uma aresta é adicionada ao fecho transitivo somente se ela liga duas regiões que pertencem a um pedaço da imagem que possua um alto grau de supersegmentação. Esse critério só pode ser adotado se for possível estimar localmente o grau de supersegmentação das regiões da imagem.

No exemplo de reconhecimento de imagens cerebrais tratado em [50], o modelo de referência é um atlas (uma imagem 3D com rótulos), a partir da qual a estrutura do grafo é construída usando o método de construção local. A IRM a ser reconhecida é automaticamente supersegmentada de forma homogênea. Objetos na imagem podem apresentar alterações na sua forma, decorrentes do processo de segmentação. O grafo que representa essa imagem é construído usando o método do fecho transitivo k-local sobre construção local ($k = 2$), utilizando o conceito de adjacências difusas. Ao aplicar-se esse conceito, são consideradas como adjacentes além das regiões fisicamente situadas lado a lado na imagem, aquelas cuja distância não excede a um dado parâmetro ou ainda aquelas que possuem outras informações de proximidade representadas pelos atributos.

2.6 Morfismo difuso de grafos

O processo de reconhecimento de cenas baseado em grafos implica na representação por grafos das estruturas da cena e do modelo (resp. conjunto de modelos) ao qual (resp. aos quais) a cena será comparada. Além disso, é necessária a definição de uma estratégia de comparação entre os grafos que representam o modelo e a cena a ser reconhecida.

A definição de morfismo difuso de grafos é apresentada em [49, 50]. Ela se propõe a uniformizar as definições apresentadas na literatura de reconhecimento de cenas. A partir dela é possível representar tanto a correspondência entre grafos ou subgrafos através da noção de isomorfismo, quanto as correpondências com regras menos estritas de associação de vértices e arestas. Morfismos difusos são adequados para representar aspectos não bijetivos de um problema de reconhecimento.

No morfismo de grafos definido em [49, 50] e inicialmente introduzido por Rosenfeld [56], um único modelo é utilizado para realizar a correspondência entre os grafos. A imagem a ser reconhecida é submetida a um processo de segmentação automática. O método escolhido é o de supersegmentação, gerando uma imagem rotulada que possui mais regiões do que o modelo: parece ser mais fácil reunir vários "pedaços" e fazê-los corresponder a uma única região do modelo, do que identificar os contornos das regiões do modelo na imagem (no caso de subsegmentação). Assim, no caso de supersegmentação, cada região do modelo deve ser associada a várias regiões da cena e as relações de semelhança entre regiões são mais de inclusão do que de equivalência. Sejam dois grafos $G_i = (N_i, E_i), i \in \{1, 2\}$, onde N_i representa um conjunto de vértices e $E_i \subseteq N_i \times N_i$ um conjunto de arestas. Variáveis indexadas por σ se referem aos vértices, enquanto variáveis indexadas por μ se referem às arestas, de acordo com a notação proposta por Rosenfeld [56]. Variáveis indexadas por 1 se referem ao grafo G_1, enquanto que aquelas indexadas por 2 se referem a G_2.

Definição 2.8 *Seja X um conjunto qualquer. Uma aplicação $\rho : X \to [0, 1]$ é chamada de* função de pertinência *de X e define um conjunto difuso sobre X. O conjunto $sup(\rho) = \{x \in X | \rho(x) > 0\}$ é chamado de suporte de ρ.*

Definição 2.9 *Um morfismo difuso do grafo G_1 para G_2 é definido como um par de funções de pertinência $\rho_\sigma = \{\rho_\sigma^{u_1}, u_1 = 1, \ldots, |N_1|\}$ e $\rho_\mu = \{\rho_\mu^{e_1}, e_1 = 1, \ldots, |E_1|\}$*

definidas respectivamente sobre seus vértices e arestas, onde $\rho_\sigma^{u_1} = (\rho_\sigma^{u_1}(1), \rho_\sigma^{u_1}(2), \ldots, \rho_\sigma^{u_1}(|N_2|)) \in [0,1]^{|N_2|}$ *e* $\rho_\mu^{e_1} = (\rho_\mu^{e_1}(1), \rho_\mu^{e_1}(2), \ldots, \rho_\mu^{e_1}(|E_2|)) \in [0,1]^{|E_2|}$.

Cada aplicação $\rho_\sigma^{u_1}$ estabelece correspondências entre o vértice $u_1 \in N_1$ e todos os vértices de G_2 (com um valor associado definido no intervalo $[0,1]$). A Figura 2.6 mostra um exemplo das aplicações ρ_σ e ρ_μ. Para cada vértice u_1 (resp. aresta e_1) colorida com a mesma cor em G_1, todos os vértices v_2 (resp. todas as arestas) com a mesma cor em G_2 são aqueles (resp. aquelas) com $\rho_\sigma^{u_1}(u_2) > 0$ (resp. $\rho_\mu^{e_1}(e_2) > 0$).

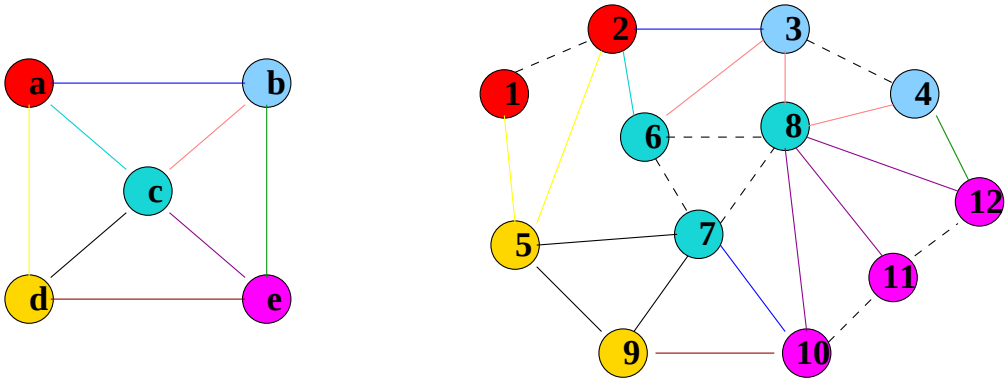


Figura 2.6: Exemplos de aplicações ρ_σ e ρ_μ

Condições que garantem a conservação da estrutura do grafo pela aplicação do morfismo foram também definidas e são apresentadas a seguir.

A Condição 2.1 abaixo foi proposta para permitir a detecção da estrutura do grafo G_1 em G_2. Ela garante que uma correspondência não nula entre arestas implica em uma correspondência não nula entre as extremidades dessas arestas, indicando a relaxação da propriedade de bijetividade (Figura 2.7).

Condição 2.1 $\forall(u_1,v_1) \in E_1, \forall(u_2,v_2) \in E_2 : \rho_\mu^{(u_1,v_1)}((u_2,v_2)) > 0 \Rightarrow$
$\rho_\sigma^{u_1}(u_2) > 0$ *e* $\rho_\sigma^{v_1}(v_2) > 0$.

A satisfação da Condição 2.1 garante a dependência entre as aplicações ρ_σ e ρ_μ, como apresentado na Figura 2.7, sempre que os grafos são associados. Isto é, as relações entre arestas são definidas de acordo com as relações entre vértices. A Condição 2.2 é uma extensão para grafos não direcionados, da condição anterior. Essa condição é válida para grafos direcionados ou não direcionados. Ela será utilizada no lugar da Condição 2.1 na formulação do problema de correspondência de grafos de atributos difusos apresentada no próximo capítulo.

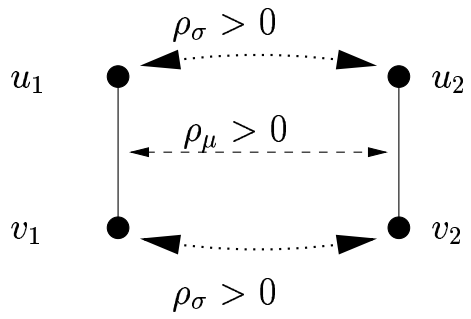


Figura 2.7: Condição 2.1 do morfismo

Condição 2.2 $\forall(u_1, v_1) \in E_1, \forall(u_2, v_2) \in E_2 : \rho_\mu^{(u_1,v_1)}((u_2, v_2)) > 0 \Rightarrow$
$(\rho_\sigma^{u_1}(u_2) > 0 \ e \ \rho_\sigma^{v_1}(v_2) > 0) \ ou \ (\rho_\sigma^{u_1}(v_2) > 0 \ e \ \rho_\sigma^{v_1}(u_2) > 0)$.

Uma outra condição [50] deve ser também considerada para garantir que a estrutura de G_1 seja preservada em G_2. A Condição 2.3 é usada para evitar associações entre pares de vértices terminais de arestas não associadas, como ilustrado na Figura 2.8. Essa condição garante a existência de pelo menos uma aresta pertencente a $sup(\rho_\sigma^{u_1}) \times sup(\rho_\sigma^{v_1})$ e associada a (u_1, v_1) por ρ_μ, se $sup(\rho_\sigma^{u_1}) \neq \emptyset$ e $sup(\rho_\sigma^{v_1}) \neq \emptyset$.

Condição 2.3 $\forall(u_1, v_1) \in E_1 : sup(\rho_\sigma^{u_1}) \neq \emptyset \quad e \quad sup(\rho_\sigma^{v_1}) \neq \emptyset \Rightarrow$
$\exists(u_2, v_2) \in sup(\rho_\sigma^{u_1}) \times sup(\rho_\sigma^{v_1}) : (u_2, v_2) \in sup(\rho_\mu^{(u_1,v_1)})$.

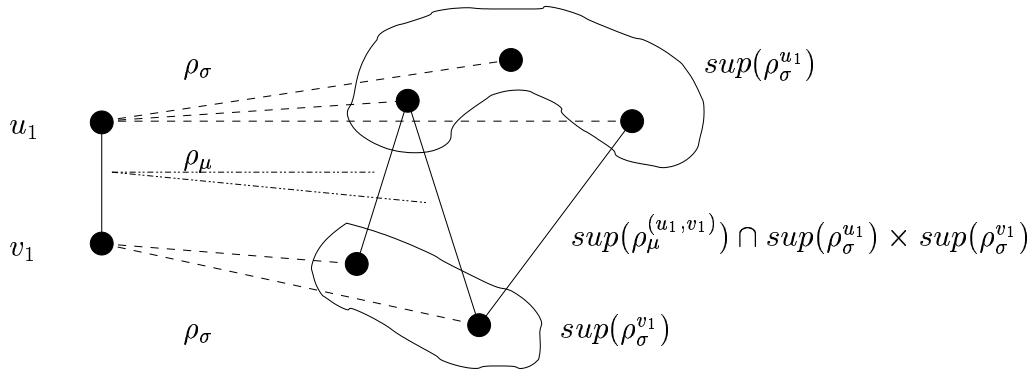


Figura 2.8: Condição 2.3 do morfismo

Com esta definição de morfismo, várias possibilidades de correspondência entre os grafos G_1 e G_2 são permitidas. Entre elas, deseja-se encontrar aquela que melhor represente o reconhecimento da imagem. Técnicas de otimização para encontrar a melhor correspondência considerando-se o morfismo difuso de grafos já foram investigadas. Foram propostos algoritmos determinísticos de relaxação difusa

[50], procurando-se identificar através de grafos de associação, valores de ρ_σ localmente compatíveis com valores de ρ_μ. Investigou-se também algoritmos genéticos utilizando-se uma versão simplificada da função f_1 proposta no próximo capítulo como função de adequação [52]. Algoritmos baseados na estimativa de distribuições de probabilidade (EDA) [2] foram também estudados. Essa abordagem foi previamente comentada na Seção 2.4.3.

A formulação do problema de correspondência de grafos de atributos difusos (PCGAD) como um problema de otimização combinatória é apresentada no próximo capítulo. Cada uma das possíveis correspondências estabelecidas entre os dois grafos é definida como uma solução do PCGAD e representada pelas associações entre vértices dos dois grafos.

Capítulo 3

O problema de correspondência de grafos de atributos difusos

Neste capítulo é formulado o problema de correspondência de grafos de atributos difusos (PCGAD) como um problema de otimização combinatória. Duas funções objetivo são definidas e investigadas. Além delas, um conjunto de restrições é proposto com o intuito de redução do espaço de soluções.

3.1 Introdução

A formulação do Problema de Correspondência entre Grafos de Atributos Difusos (PCGAD) é proposta neste capítulo. Uma solução para esse problema representa uma correspondência estabelecida entre os vértices dos grafos G_1 (modelo) e G_2 (imagem a ser reconhecida). Para cada região do modelo (um vértice do grafo G_1), buscam-se as regiões correspondentes na imagem segmentada (vértices do grafo G_2), usando-se para isso informações dos vértices e das arestas. A título de exemplo, são apresentadas na Figura 3.1 duas soluções distintas para uma pequena instância do PGCAD. Cada cor indica um vértice (ou uma aresta) do modelo e o subconjunto de vértices (ou de arestas) que possivelmente correspondem a esse vértice na imagem. A escolha de cada subconjunto depende das restrições do problema e dos pesos associados às correspondências entre vértices e arestas dos dois grafos.

Várias são as correspondências permitidas pelo morfismo discutido na Seção 2.6. Nelas, arestas de G_2 que ligam vértices associados a um mesmo vértice de G_1 não estão associadas a alguma aresta de G_1. Na Figura 3.1, as linhas tracejadas representam arestas de G_2 que não foram associadas a arestas de G_1. Sejam, por

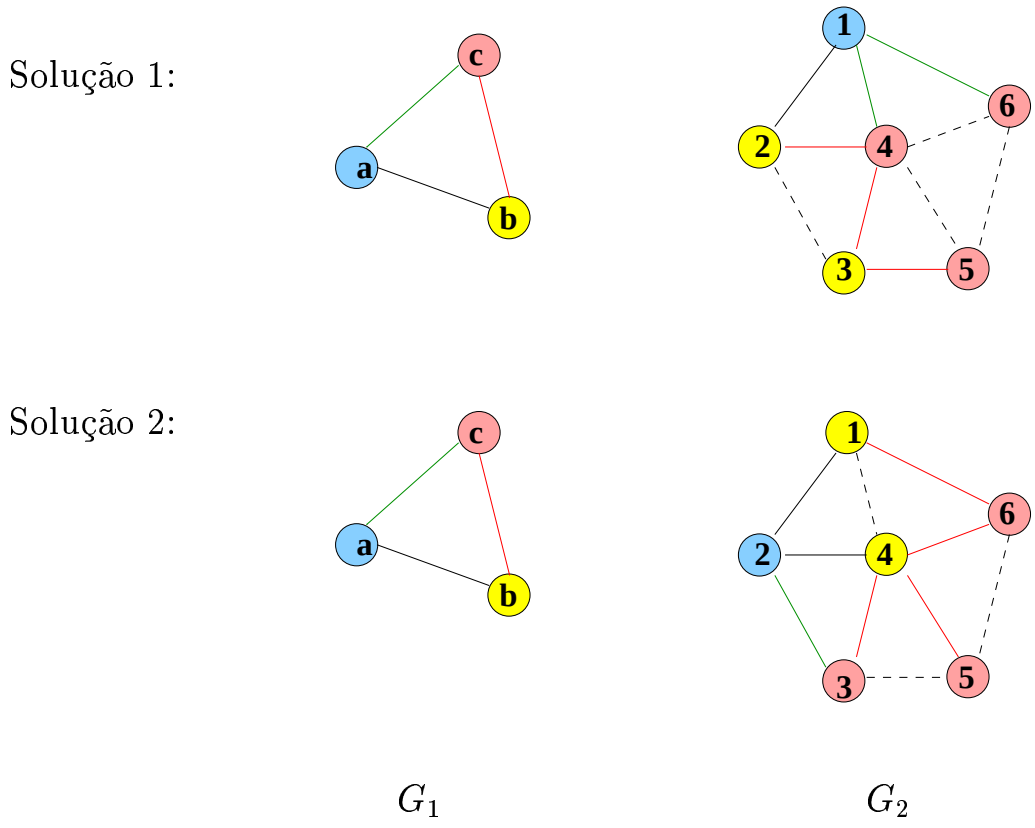


Figura 3.1: Duas soluções distintas para uma mesma instância do PGCAD

exemplo, as associações entre os vértices extremos do par de arestas $(c, b) \in E_1$ e $(4, 6) \in E_2$ da Solução 1 da Figura 3.1. Esse tipo de associação entre vértices atende à Condição 3.1 descrita a seguir. Neste caso, a existência de uma associação entre esse par de arestas violaria a Condição 2.2. A proibição da correspondência de arestas de G_2 que conectam vértices associados a um mesmo vértice de G_1 (dada pela Condição 3.1 abaixo) é conseqüência da Condição 2.2.

Condição 3.1 $\forall(u_1, v_1) \in E_1, \forall(u_2, v_2) \in E_2$:
$(\rho_\sigma^{u_1}(u_2) > 0, \rho_\sigma^{u_1}(v_2) > 0, \rho_\sigma^{v_1}(u_2) = 0, \rho_\sigma^{v_1}(v_2) = 0)$ *ou*
$(\rho_\sigma^{u_1}(u_2) = 0, \rho_\sigma^{u_1}(v_2) = 0, \rho_\sigma^{v_1}(u_2) > 0, \rho_\sigma^{v_1}(v_2) > 0) \Rightarrow$
$\rho_\mu^{(u_1,v_1)}((u_2, v_2)) = 0.$

Seja $\mathcal{S}$ o conjunto de todas as possíveis correspondências entre os grafos G_1 e G_2. O subconjunto $\mathcal{S}' \subseteq \mathcal{S}$ formado por todas as correspondências cujas associações entre arestas são determinadas pelas associações entre vértices [1] define o espaço de soluções

[1] São associadas a uma aresta de G_1 todas as arestas de G_2 cujos vértices extremos estejam associados.

do PCGAD. São consideradas apenas duas possibilidades para cada associação entre vértices e entre arestas: ou ela não existe, ou existe e sua intensidade é igual ao valor de similaridade calculado a partir dos atributos associados aos dados de entrada. Com essa definição, se reduz o número de correspondências entre G_1 e G_2 avaliadas no espaço de busca. Todas as soluções em $\mathcal{S}'$ podem ser representadas apenas pelas associações entre vértices. As associações entre arestas são implicitamente derivadas, de acordo com a seguinte regra: uma aresta (a, b) de G_1 é associada a todas as arestas (a', b') de G_2 tais que a é associado a a' (resp. b') e b é associado a b' (resp. a') (vide Figuras 3.1 e 3.2).

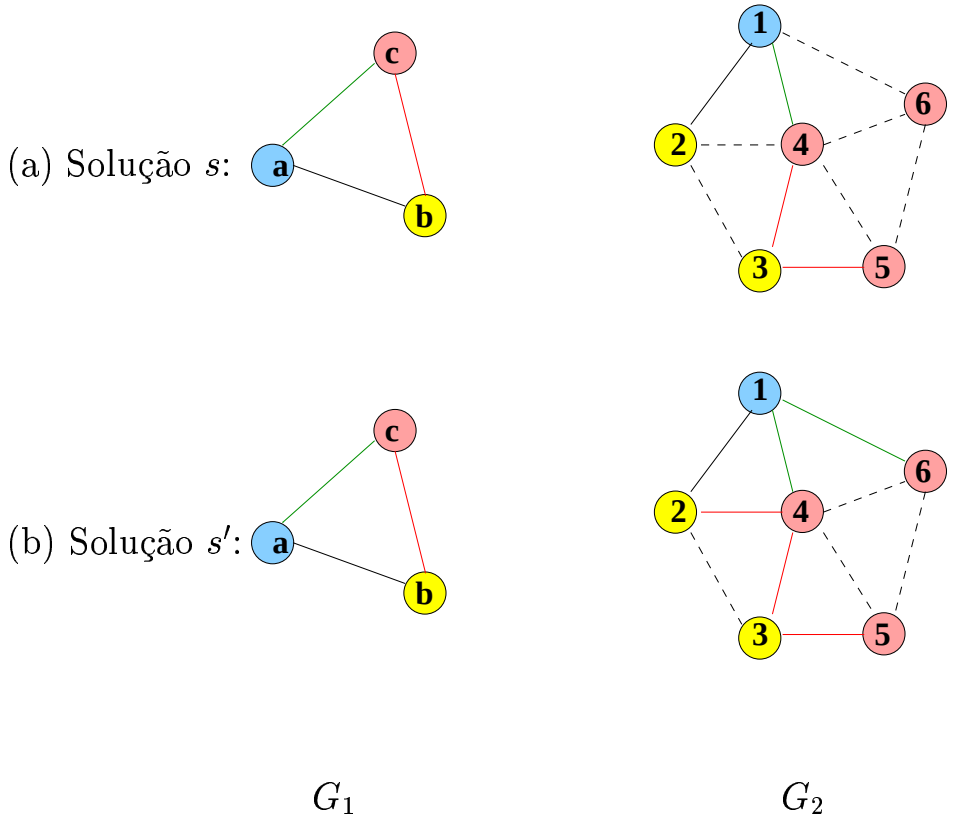


Figura 3.2: (a) $s \in \mathcal{S}$ e $s \notin \mathcal{S}'$; (b) $s' \in \mathcal{S}' \subseteq \mathcal{S}$

A melhor solução de $\mathcal{S}'$ deveria representar a correspondência na qual a estrutura do grafo modelo pudesse ser detectada na estrutura do grafo imagem. Sejam $G_1 = (N_1, E_1)$ o grafo que representa o grafo modelo e $G_2 = (N_2, E_2)$ o grafo que representa a imagem segmentada. Uma *solução* para o PCGAD pode ser representada por um grafo bipartido $G' = (N_1 \cup N_2, E')$, onde $E' \subseteq \{(i, j), i \in N_1, j \in N_2 \mid \rho^i_\sigma(j) > 0\}$.

O grafo G' pode ainda ser representado por suas listas de adjacência. Para cada vértice $i \in N_1$, define-se o subconjunto de vértices de N_2 a ele associados por

$A_i = \{j \in N_2 \mid (i,j) \in E'\}$.

Matrizes de similaridade são construídas com valores calculados a partir dos atributos dos grafos. A escolha desses atributos depende da imagem a ser reconhecida. Seja S^v (resp. S^a) uma matriz de similaridade de vértices (resp. de arestas) com elementos $s^v(i,j)$ (resp. $s^a(i,j)$) $\in [0,1]$ cujos valores representam as similaridades entre os vértices (resp. as arestas) $i \in N_1$ e $j \in N_2$ (resp. $i \in E_1$ e $j \in E_2$).

As aplicações ρ_σ e ρ_μ apresentadas na Seção 2.6 representam as correspondências que podem ser estabelecidas entre os grafos comparados, enquanto que as matrizes de similaridade representam os valores associados a elas. A diferença entre esses dois conceitos deve ser enfatizada. Um valor $\rho_\sigma^{u_1}(u_2) = 0$ (resp. $\rho_\mu^{e_1}(e_2) = 0$) significa que não existe associação estabelecida entre os vértices (resp. as arestas) u_1 e u_2 (resp. e_1 e e_2). Por outro lado, uma correspondência entre vértices (ou arestas) com valor nulo significa que os objetos podem estar associados, mas que a similaridade entre eles é nula.

Em alguns métodos de otimização propostos para o problema de correspondência entre grafos [50, 53, 57], as aplicações ρ_σ e ρ_μ são inicializadas com valores de similaridade e iterativamente atualizadas. De acordo com [53], existem algoritmos de correspondência de grafos baseados na definição de grafos de associação. Cada vértice de um grafo de associação representa um mapeamento entre nós dos grafos comparados. Cada aresta de um grafo de associação representa a compatibilidade entre esses mapeamentos. O objetivo consiste em achar o maior subgrafo conexo do grafo de associação, que representa a melhor correspondência. Pesos podem ser usados para privilegiar ou desqualificar vértices e arestas de um grafo de associação. Eles podem ser inicialmente calculados como valores de similaridade derivados de informações estruturais de regiões das imagens e podem ser intensificados ou enfraquecidos de acordo com informações contextuais, tais como objetos desaparecidos nas imagens comparadas. Em abordagens desse tipo, não existe uma função de custo associada aos mapeamentos. Nesta tese, o conceito de grafo de associação não é usado e os mapeamentos são avaliados por intermédio de uma função de custo. Os valores não nulos de ρ_σ e ρ_μ associados a uma solução do PGCAD são considerados como fixos e iguais aos valores de similaridade calculados a partir dos atributos que descrevem as imagens (valores de similaridade entre vértices e entre arestas). Por essa razão, neste e nos próximos capítulos, as funções $\rho_\sigma^i(j)$ e $\rho_\mu^l(k)$ serão substituídas pelos valores de similaridade $s^v(i,j)$ e $s^a(l,k)$.

3.2 Funções objetivo

Uma definição precisa da função objetivo que leva à melhor correspondência é difícil de ser formulada, devido a imprecisões e descrições ruins de aspectos estruturais relevantes dos dados de entrada. Nesta seção, duas funções objetivo distintas para o PCGAD são propostas e investigadas, em termos de suas propriedades e capacidade de reconhecimento. O objetivo principal consiste em identificar a função cujo valor máximo corresponda à imagem da solução que representa a correspondência mais adequada (denotada neste capítulo por s^*), isto é, a função que melhor representa o reconhecimento da imagem. Como a qualidade da entrada de dados afeta fortemente o comportamento das funções, dois pequenos exemplos artificiais serão utilizados para ilustrar esse estudo. A influência dos valores de similaridade na busca por s^* também será discutida.

Dois exemplos e suas respectivas soluções s^* são mostradas nas Figuras 3.3 e 3.4. As matrizes de similaridade de vértices e de arestas para esses exemplos são fornecidas nas Tabelas 3.1, 3.2, 3.3 e 3.4.

Exemplo 1

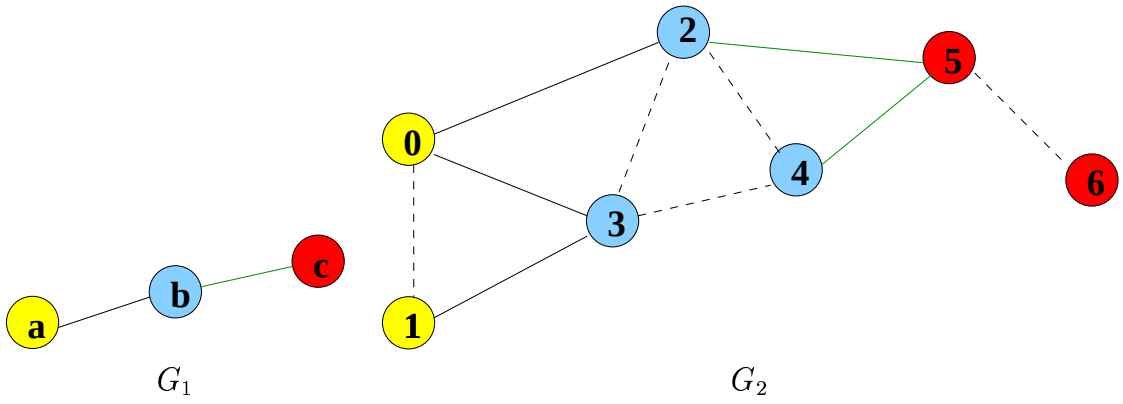


Figura 3.3: Correspondência mais adequada (Exemplo 1)

Observando-se as matrizes apresentadas nas Tabelas 3.1 e 3.2, nota-se que as similaridades de vértices naturalmente conduzem a busca da melhor correspondência para aquela ilustrada na Figura 3.3. Os valores de similaridade entre arestas calculadas neste caso não são discriminantes. Isso significa que eles não auxiliam de forma relevante na identificação das características estruturais de G_1 em G_2.

Exemplo 2

	0	1	2	3	4	5	6
a	0,913	1,	0,	0,	0,	0,	0,
b	0,	0,	1,	1,	1,	0,	0,
c	0,087	0,	0,	0,	0,	0,742	0,949

Tabela 3.1: Valores de similaridade entre vértices (S^v) para o Exemplo 1

	(0,1)	(0,2)	(0,3)	(1,3)	(2,3)	(2,4)	(2,5)	(3,4)	(4,5)	(5,6)
(a,b)	0,995	1,	1,	0,995	1,	0,995	0,995	0,995	0,995	1,
(b,c)	0,995	1,	1,	0,995	1,	0,995	0,995	0,995	0,995	1,

Tabela 3.2: Valores de similaridade entre arestas (S^a) para o Exemplo 1

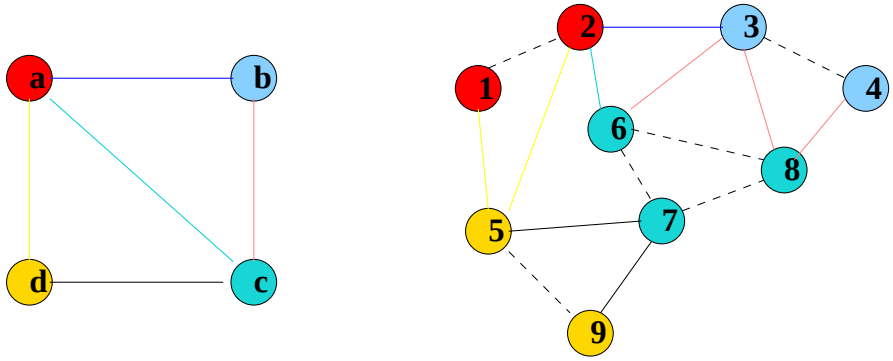


Figura 3.4: Correspondência mais adequada (Exemplo 2)

No segundo exemplo, as informações provenientes da matriz de similaridade entre arestas não são suficientes para permitir a detecção da estrutura de G_1 em G_2. Além disso, a matriz de similaridade entre vértices possui informações menos discriminantes do que aquela apresentada no Exemplo 1. Neste caso, várias são as correspondências entre os grafos com valores altos de similaridade entre vértices.

Algumas restrições devem ser impostas para reduzir o espaço de soluções do PCGAD e auxiliar a convergência da função objetivo para a solução ótima que coincida com aquela que melhor representa o reconhecimento da imagem.

A Restrição 3.1 garante que cada vértice de N_2 tem que ser associado a exatamente um vértice de N_1. Ao impor-se essa restrição ao problema, admite-se que o método de supersegmentação adotado não gera segmentos da imagem que correspondam simultaneamente a regiões adjacentes no modelo, ou seja, os contornos das regiões do modelo são preservados.

Restrição 3.1 *Existe exatamente um nó* $i \in N_1$ *tal que* $(i, j) \in E'$, $\forall j \in N_2$.

	1	2	3	4	5	6	7	8	9
a	0,91	1,	0,75	0,	0,75	0,85	0,20	0,10	0,
b	0,	0,	1,	1,	0,	0,	0,	0,90	0,
c	1,	0,	0,	0,	0,	0,80	1,	1,	0,
d	1,	0,	0,	0,	1,	0,	1,	0,	0,70

Tabela 3.3: Valores de similaridade entre vértices (S^v) para o Exemplo 2

	(1,2)	(1,5)	(2,3)	(2,5)	(2,6)	(3,4)	(3,6)	(3,8)
(a,b)	0,99	1,	1,	0,99	1,	0,99	0,99	0,99
(a,c)	0,99	1,	1,	0,99	1,	0,99	0,99	0,99
(a,d)	0,51	0,50	0,50	0,51	0,50	0,51	0,51	0,51
(b,c)	0,99	1,	1,	0,99	1,	0,99	0,99	0,99
(c,d)	0,99	1,	1,	0,99	1,	0,99	0,99	0,99

	(4,8)	(5,7)	(5,9)	(6,7)	(6,8)	(7,8)	(7,9)
(a,b)	0,99	1,	0,98	0,99	0,99	0,99	1,
(a,c)	0,99	1,	0,98	0,99	0,99	0,99	1,
(a,d)	0,51	0,50	0,52	0,51	0,51	0,51	0,50
(b,c)	0,99	1,	0,98	0,99	0,99	0,99	1,
(c,d)	0,99	1,	0,98	0,99	0,99	0,99	1,

Tabela 3.4: Valores de similaridades entre arestas (S^a) para o Exemplo 2

3.3 Função objetivo f_1

A função f_1 é composta de dois termos, que representam respectivamente as contribuições das associações entre vértices e entre arestas de G_1 e G_2 para medir a qualidade da correspondência representada por uma solução do PCGAD.

Associações entre vértices e entre arestas com valores altos de similaridade são privilegiadas por f_1, enquanto aquelas com valores baixos de similaridade são penalizadas. Arestas que não respeitam a Condição 2.2 do morfismo também são penalizadas. A partir de estudos realizados com os exemplos apresentados na seção anterior, observou-se que o comportamento da função f_1 é sensível a valores discriminantes de similaridade entre vértices e entre arestas. Caso isso não ocorra, correspondências que não correspondem à mais adequada podem ser erroneamente encontradas como se fossem soluções de alta qualidade.

A função f_1 a ser maximizada é definida como

$$f_1(G') = \frac{\alpha}{|N_1| \times |N_2|} \left[\sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|) \right] +$$

$$\frac{(1-\alpha)}{|E_1| \times |E_2|} \left[\sum_{(i,i') \in E_1} \sum_{(j,j') \in E_2} (1 - |max\{x_{ij}x_{i'j'}, x_{ij}x_{j'i'}\} - s^a((i,i'),(j,j'))|) \right], \quad (3.1)$$

onde α é um parâmetro usado para ponderar cada termo de f_1. A variável x_{ij} assume o valor 1 se o arco $(i,j) \in E'$, 0 caso contrário.

O primeiro termo do lado direito da Equação 3.1 representa a contribuição média dos vértices para a correspondência. Quatro situações limite são possíveis:

- $x_{ij} = 0$ e $s^v(i,j) = 0$: o termo $1 - |x_{ij} - s^v(i,j)|$ contribui com seu valor máximo para f_1 (vértices i e j com valores de similaridade nula não devem ser associados).
- $x_{ij} = 0$ e $s^v(i,j) = 1$: o termo $1 - |x_{ij} - s^v(i,j)|$ não contribui para f_1 em soluções que possuam vértices i e j com alto valor de similaridade mas que não tenham sido associados.
- $x_{ij} = 1$ e $s^v(i,j) = 0$: o termo $1 - |x_{ij} - s^v(i,j)|$ não contribui para f_1 em soluções que possuam vértices i e j associados, porém com valor de similaridade nulo.
- $x_{ij} = 1$ e $s^v(i,j) = 1$: o termo $1 - |x_{ij} - s^v(i,j)|$ contribui com seu valor máximo para f_1 (vértices i e j com alto valor de similaridade são associados).

Uma análise similar pode ser feita para os termos que compõem a segunda parte do lado direito da Equação 3.1, que envolve associações entre arestas e seus valores de similaridade.

Gráficos que representam valores da função objetivo para todas as soluções do PCGAD serão apresentados a seguir. O símbolo '+' presente no gráfico representa o valor de função associado à correspondência que denotamos por s^*.

A Figura 3.5 exibe os valores de f_1 em ordem crescente para todas as soluções do PCGAD para o Exemplo 1. A solução s^*, apresentada na Figura 3.3, é associada ao melhor valor de f_1, para os três valores testados do parâmetro α. Para esse exemplo, a função f_1 parece ser robusta com respeito à variação do parâmentro α.

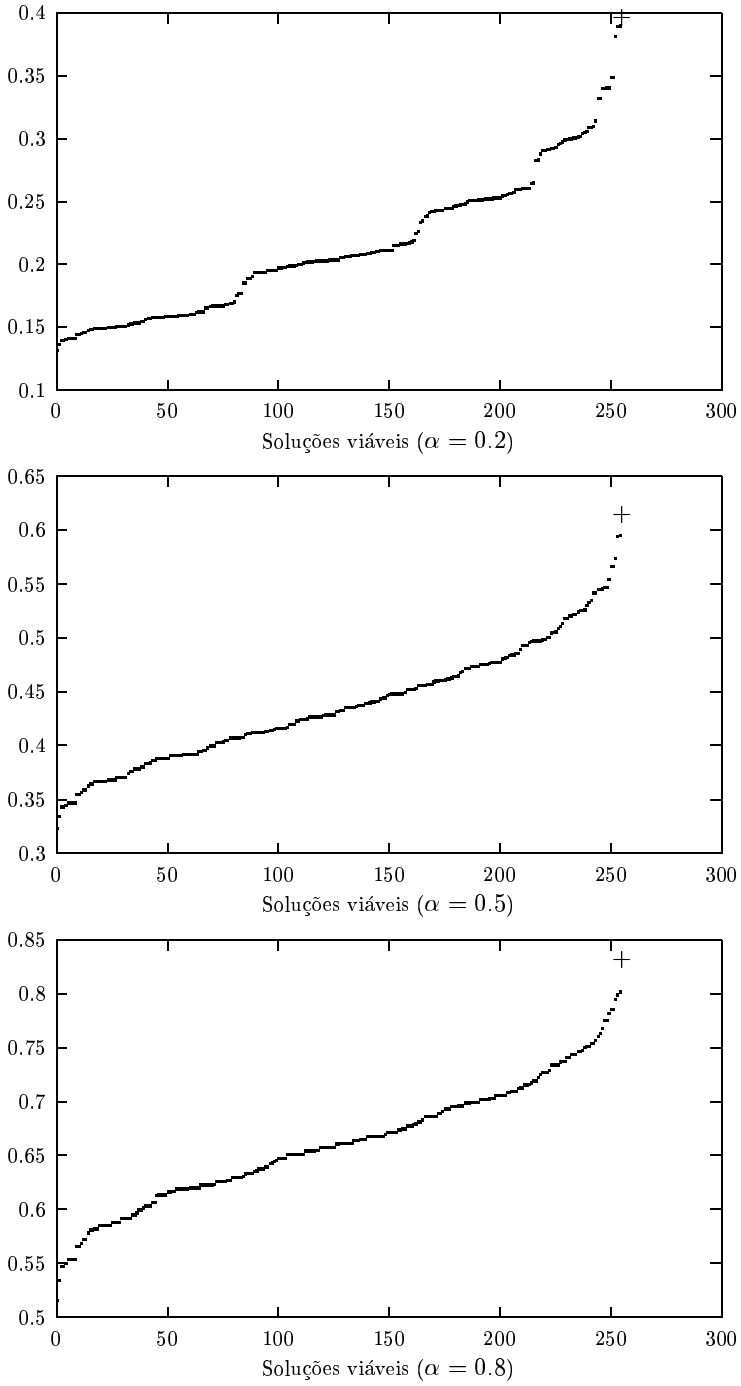


Figura 3.5: Valores da função f_1 para as soluções do Exemplo 1

A Figura 3.6 mostra valores de f_1 para soluções do Exemplo 2. Neste caso, existem soluções com valores de f_1 maiores do que aquele correspondente à solução s^*, apresentada na Figura 3.4. A Figura 3.7 ilustra uma solução com essa característica. Seu valor de f_1 é 0,707, enquanto que o valor correspondente a s^* é 0,699 (valores obtidos para $\alpha = 0{,}8$).

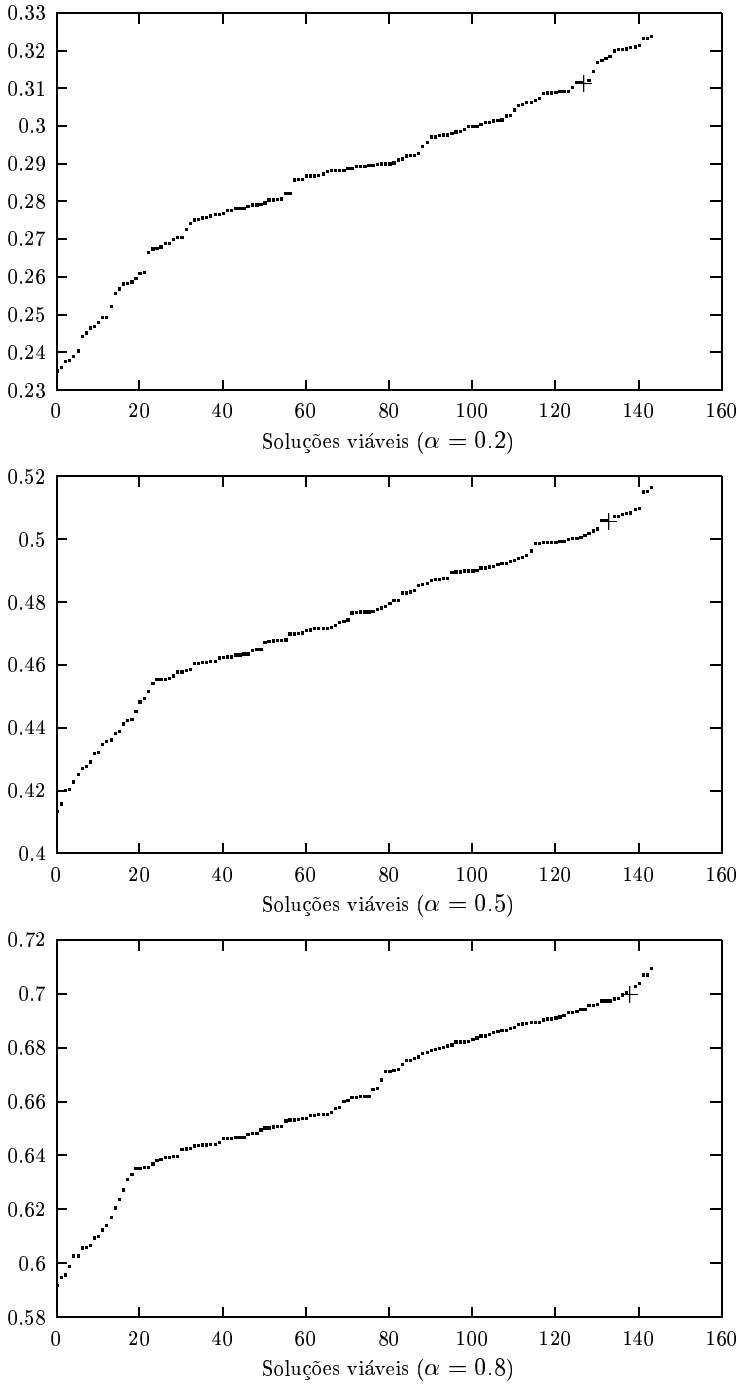


Figura 3.6: Valores da função f_1 para as soluções do Exemplo 2

As associações entre vértices da solução apresentada na Figura 3.7 são quase as mesmas presentes em s^*. No entanto, as associações entre arestas determinadas por elas são ligeiramente diferentes. Há mais arestas de G_2 associadas a G_1 (respeitando a Condição 2.2 do morfismo) nesta solução do que em s^*. O valor de f_1 calculado para essa solução pode ser eventualmente maior do que aquele relativo a s^*, porque o valor dado pela função f_1 é uma média das contribuições dos vértices e

das arestas. Em casos como este, a matriz de similaridade entre vértices não fornece informação significativa para bem discriminar as soluções em termos da qualidade do reconhecimento.

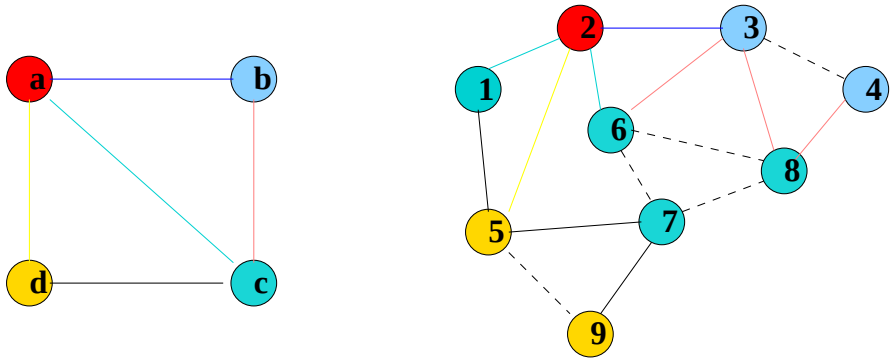


Figura 3.7: Solução com valor de f_1 maior que aquele associado à solução mais adequada

A qualidade dos dados de entrada (matrizes de similaridade de vértices e de arestas) é primordial para a identificação da melhor correspondência. No entanto, como essa qualidade nem sempre é fácil de ser alcançada em exemplos reais, alguns aspectos adicionais das correspondências devem ser utilizados para auxiliar a busca por s^*. Por exemplo, na solução apresentada na Figura 3.7, os vértices 1, 6, 7 e 8 de G_2, associados ao vértice c de G_1 não estão todos conectados entre si. Essa não é uma característica comum em situações reais (um método de supersegmentação pode partir um objeto em pequenos pedaços, mas não altera as posições desses pedaços). Uma boa estratégia para evitar esse tipo de solução é restringir o espaço $\mathcal{S}'$ de soluções do PCGAD àquelas onde cada conjunto A_i dos vértices de G_2 associados ao nó i de G_1 induz um subgrafo conexo em G_2, para todo vértice i de G_1 (restrição de conexidade). A Figura 3.8 mostra a melhoria de comportamento de f_1 quando esse aspecto é considerado. Além de reduzir a dimensão do espaço de busca, várias das soluções eliminadas tem valores de f_1 maiores do que aquele de s^*. Essa restrição será também considerada na busca de soluções para o PCGAD.

Restrição 3.2 *O subgrafo induzido em* $G_2 = (N_2, E_2)$ *por* A_i *é conexo,* $\forall i \in N_1$.

A Figura 3.9 mostra valores de f_1 calculados para o Exemplo 2 com os novos valores de similaridade entre arestas dados na Tabela 3.5, que propositadamente privilegiam as associações entre arestas que aparecem em s^* (Figura 3.4). A Figura

3.10 mostra valores de f_1 calculados para o Exemplo 2, considerando-se a nova matriz de similaridade entre arestas e a restrição de conexidade. Para os três valores testados para o parâmetro α, o maior valor de f_1 corresponde à solução s^*. Esses resultados confirmam a importância da qualidade dos dados de entrada para o bom desempenho da função de otimização na identificação da melhor correspondência.

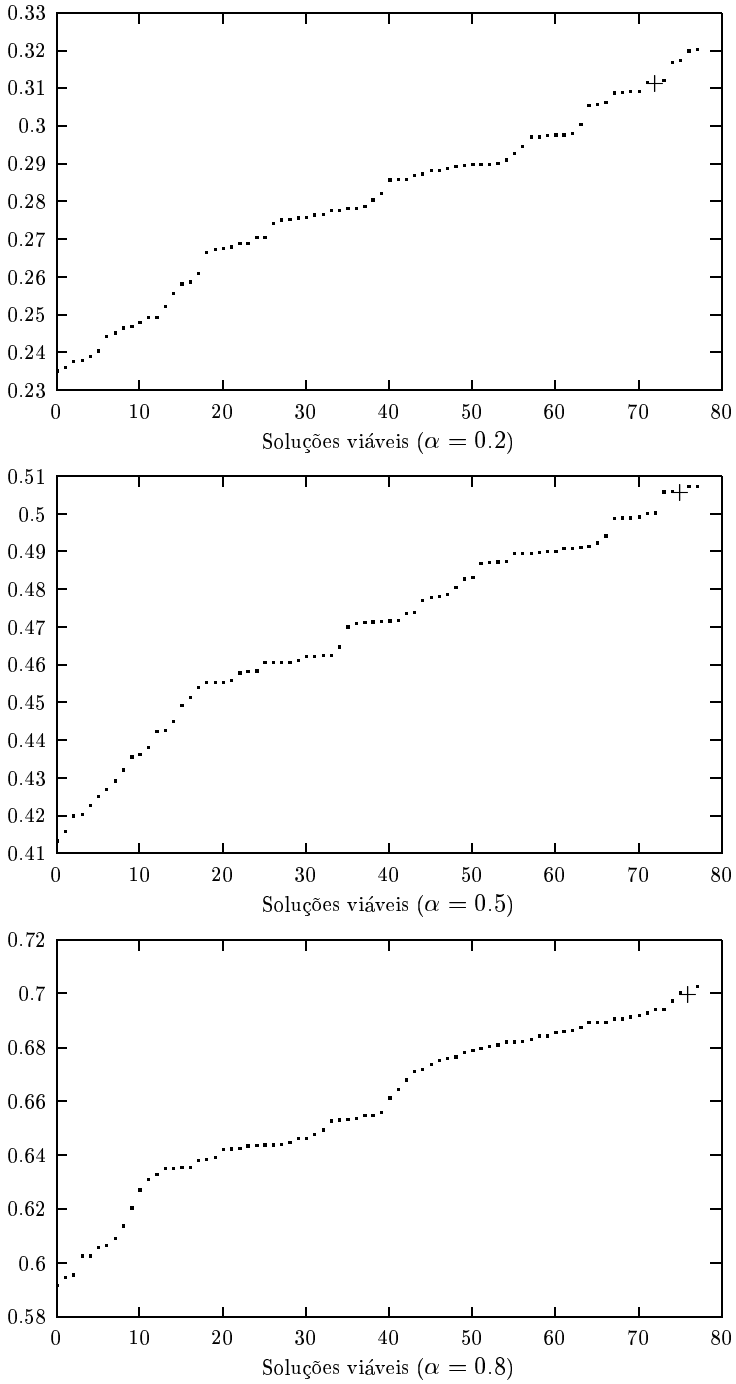


Figura 3.8: Valores de f_1 para o Exemplo 2 considerando-se apenas soluções que satisfazem à Restrição 3.2

	(1,2)	(1,5)	(2,3)	(2,5)	(2,6)	(3,4)	(3,6)	(3,8)
(a,b)	0,8	0,8	1,	0,8	0,8	0,8	0,8	0,8
(a,c)	0,332	0,327	0,327	0,332	1,	0,332	0,327	0,332
(a,d)	0,8	1,	0,8	1,	0,8	0,8	0,8	0,8
(b,c)	0,8	0,8	0,8	0,8	0,8	0,8	1,	1,
(c,d)	0,8	0,8	0,8	0,8	0,8	0,8	0,8	0,8

	(4,8)	(5,7)	(5,9)	(6,7)	(6,8)	(7,8)	(7,9)
(a,b)	0,8	0,8	0,8	0,8	0,8	0,8	0,8
(a,c)	0,332	0,332	0,332	0,332	0,332	0,332	0,332
(a,d)	0,8	0,8	0,8	0,8	0,8	0,8	0,8
(b,c)	1,	0,8	0,8	0,8	0,8	0,8	0,8
(c,d)	0,8	1,	0,8	0,8	0,8	0,8	1,

Tabela 3.5: Novos valores de similaridade entre arestas (S^e) para o Exemplo 2

Vértices com similaridade nula não devem ser associados. Esse tipo de associação é descartado pela Restrição 3.3:

Restrição 3.3 $\forall(i,j) \in E' \Rightarrow s^v(i,j) > 0$

Com a Restrição 3.3, vértices só são associados se o valor de similaridade entre eles for positivo, reforçando-se a característica da função f_1 de penalizar associações entre vértices com baixa ou mesmo nenhuma similaridade.

Para assegurar que todos os objetos do modelo aparecem no grafo que representa a imagem a ser reconhecida, uma última restrição é imposta ao PCGAD, forçando-se que cada nó de N_1 seja associado a pelo menos um nó de N_2:

Restrição 3.4 $|A_i| \geq 1, \forall i \in N_1$.

3.4 Função objetivo f_2

A função f_2 é também definida sobre o espaço de soluções do PGCAD. É importante relembrar, para um melhor entendimento de sua descrição, que para todo vértice $i \in N_1$, representa-se por $A_i \subseteq N_2$ o conjunto dos vértices de G_2 associados com o vértice i. Para cada aresta $(i,j) \in E_1$, seja $M_{ij} = \{(k,l) \in E_2 : k \in A_i, l \in A_j\}$ o conjunto de arestas de E_2 cujas extremidades são associadas aos vértices i e j de N_1. A função f_2 leva em consideração todas as arestas $(i,j) \in E_1$:

$$f_2(G') = \sum_{(i,j)\in E_1} \min\{VC_i, VC_j, EC_{i,j}\},$$

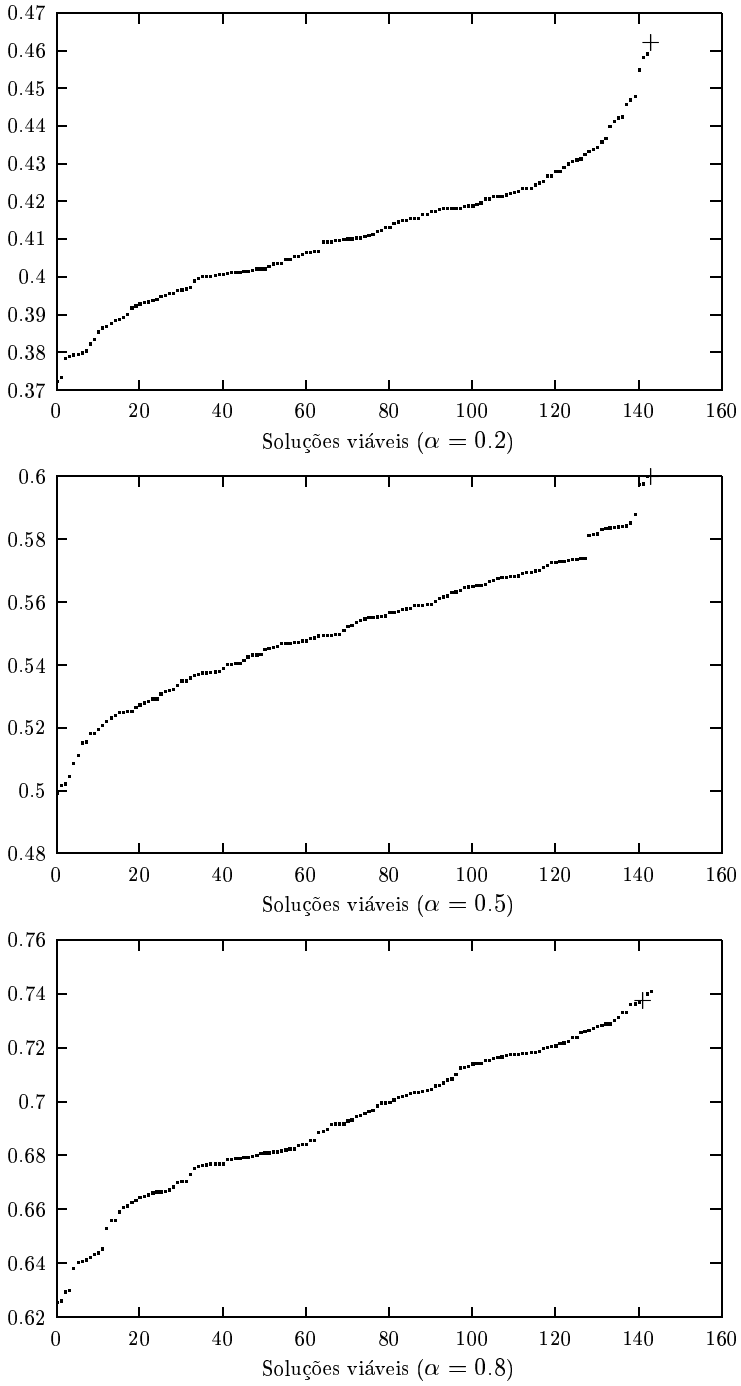


Figura 3.9: Valores de f_1 para o Exemplo 2 considerando-se os novos valores da matriz de similaridade entre arestas

onde VC_i (resp. VC_j) denota a contribuição média da associação do vértice $i \in N_1$ (resp. $j \in N_1$) com todos os vértices em A_i (resp. A_j) e EC_{ij} é a contribuição média da aresta $(i, j) \in E_1$:

$$VC_i = \sum_{k \in A_i} s^v(i, k)/|A_i|$$

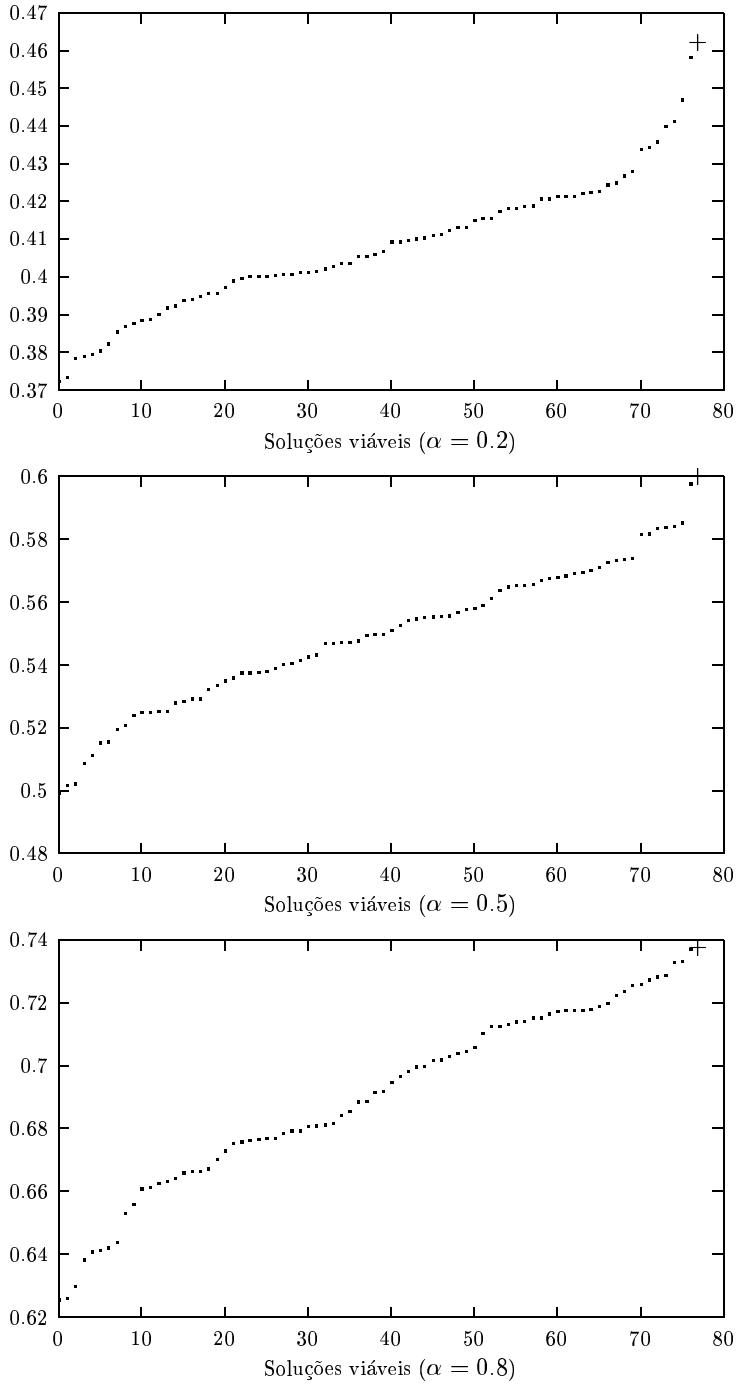


Figura 3.10: Valores de f_1 para o Exemplo 2 com a restrição de conexidade e a nova matriz de similaridade entre arestas apresentada na Tabela 3.5

e

$$EC_{ij} = \sum_{(k,l) \in M_{ij}} \min\{s^a((i,j),(k,l)), s^v(i,k), s^v(j,l)\}/|M_{ij}|.$$

A contribuição EC_{ij} de cada aresta (i,j) é avaliada como a média (sobre todas as arestas em M_{ij}) do mínimo entre o valor de associação entre o par de arestas

comparadas e os valores da associação entre as extremidades destas arestas.

A Figura 3.11 ilustra como a função f_2 avalia as correspondências entre os grafos.

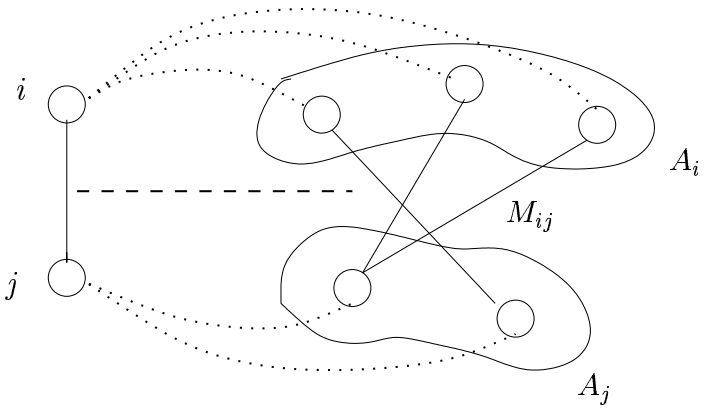


Figura 3.11: Cálculo de f_2

A busca de soluções para o PGCAD maximizando f_2 leva a correspondências que privilegiam associações com altas similaridades de vértices e de arestas, desde que associações com similaridades baixas entre vértices e entre arestas tendem a reduzir a contribuição dos termos de f_2.

A Figura 3.12 mostra valores de f_2 para o Exemplo 1. A Figure 3.13 mostra valores de f_2 para o Exemplo 2. Ambas figuras mostram soluções cujos valores de f_2 são maiores do que os de s^*. As contribuições da aresta (b, c) para s^* e para a solução com o maior valor de f_2 para o Exemplo 2 são ilustradas na Figura 3.14 (a) e (b). Sua diferença é principalmente devida aos valores de similaridade. Os melhores resultados são alcançados quando a informação das arestas é mais discriminante (Figura 3.15). A ausência de parâmetros a serem ajustados é uma importante característica dessa função.

Foram diversas as funções objetivo investigadas. Duas delas foram analisadas neste capítulo. A função f_1 foi escolhida como a mais adequada, já que é a mais simples e apresenta o melhor comportamento quando avaliada sobre os exemplos apresentados na Seção 3.2. Na próxima seção será resumida a formulação completa do PCGAD como um problema de otimização combinatória.

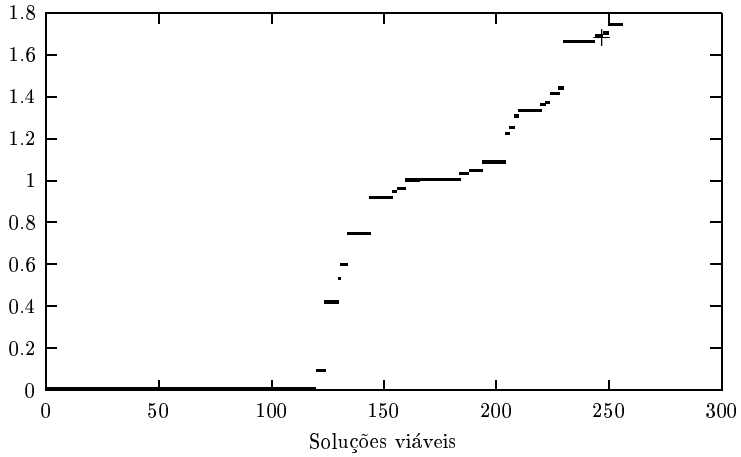


Figura 3.12: Valores da função f_2 para soluções do Exemplo 1

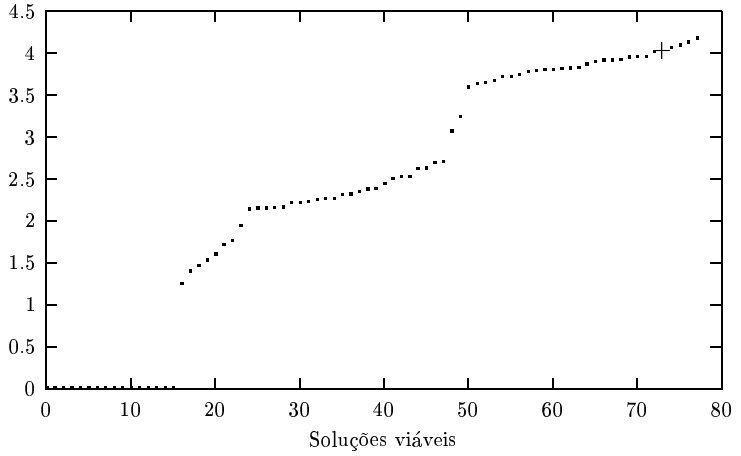
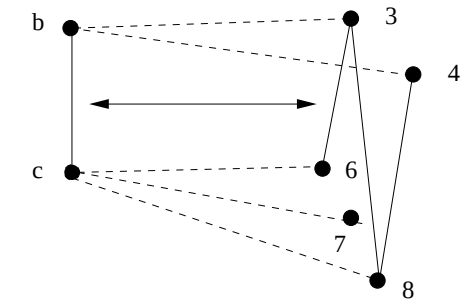


Figura 3.13: Valores da função f_2 para soluções do Exemplo 2 considerando o conjunto de restrições do PCGAD

3.5 Formulação do PCGAD como um problema de otimização combinatória

Dados dois grafos $G_1 = (N_1, E_1)$, que representa o modelo, e $G_2 = (N_2, E_2)$, que representa a imagem segmentada, e as matrizes de similaridade entre vértices e entre arestas, o problema de correspondência de grafos de atributos difusos (PCGAD) consiste em determinar a melhor correspondência entre os dois grafos, maximizando uma medida de similaridade. Cada solução para esse problema corresponde a um grafo bipartido $G' = (N_1 \cup N_2, E')$, com $E' = \{(i,j) | i \in N_1 \text{ e } j \in A_i\}$, onde A_i é o subconjunto de nós de N_2 associados com o nó $i \in N_1$. Para cada par $i \in N_1$ e $j \in N_2$, seja $x_{ij} = 1$ se $(i,j) \in E'$; $x_{ij} = 0$, caso contrário. O PCGAD é formulado

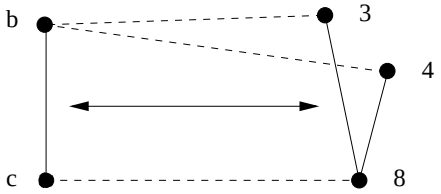
(a)



Vértice	Valor da contribuição para f_2
b	1.0
c	0.93
{b,c}	0.93

min = 0.93

(b)



Vértice	Valor da contribuição para f_2
b	1.0
c	1.0
{b,c}	0.99

min = 0.99

Figura 3.14: (a) Solução mais adequada (b) Solução com o maior valor de f_2

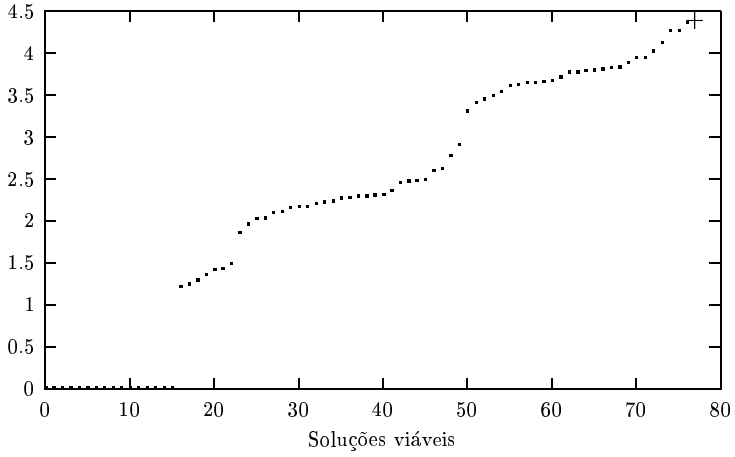


Figura 3.15: Valores da função f_2 para soluções do Exemplo 2 usando a matriz de similaridade de arestas da Tabela 3.5

como a determinação da correspondência x^* que maximiza

$$f_1(x^*) = \frac{\alpha}{|N_1| \times |N_2|} \left[\sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|) \right] +$$

$$\frac{(1-\alpha)}{|E_1| \times |E_2|} \left[\sum_{(i,i') \in E_1} \sum_{(j,j') \in E_2} (1 - |max\{x_{ij}x_{i'j'}, x_{ij}x_{j'i'}\} - s^a((i,i'),(j,j'))|) \right],$$

sujeito às restrições:

- Restrição 3.1: Existe exatamente um nó $i \in N_1$ tal que $(i, j) \in E'$, $\forall j \in N_2$.
- Restrição 3.2: O subgrafo induzido em $G_2 = (N_2, E_2)$ por A_i é conexo, $\forall i \in N_1$.
- Restrição 3.3: $\forall (i, j) \in E' \Rightarrow s^v(i, j) > 0$, onde $s^v(i, j)$ é o valor de similaridade entre os vértices i e j.
- Restrição 3.4: $|A_i| \geq 1, \forall i \in N_1$.

3.6 Conclusões

O objetivo do PCGAD é determinar a correspondência entre os grafos G_1 e G_2 que melhor descreva o reconhecimento da imagem. Devido a imprecisões nos dados de entrada e à complexidade do problema propriamente dito, uma função objetivo precisa é difícil de ser definida. Por essas razões, a solução desejada (denotada neste capítulo por s^*) pode não ser a solução ótima do PCGAD para uma dada instância do problema e para uma função objetivo adotada.

Neste capítulo foi apresentada uma formulação do PCGAD como um problema de otimização combinatória. Duas funções objetivo diferentes foram investigadas e restrições sobre o espaço de soluções foram impostas a fim de simplificar a busca da melhor solução. O conjunto de restrições foi definido a partir de características do problema, com o objetivo de favorecer a coincidência de s^* com a solução que maximiza a função objetivo (solução ótima).

As duas funções objetivo foram avaliadas sobre duas pequenas instâncias do PCGAD (Exemplos 1 e 2). Neste contexto, a função f_1 foi escolhida como a mais adequada, já que é a mais simples e apresentou o melhor comportamento. Observa-se através dos gráficos apresentados que, no caso da função f_1, a solução s^* correspondeu à solução ótima considerando-se apenas a Restrição 3.2, enquanto que no caso da função f_2, esse resultado foi atingido quando a matriz de similaridades que privilegia as associações entre arestas da solução s^* foi utilizada. Isso mostra a relevância da utilização do parâmetro α na definição da função objetivo.

A Condição 2.2 do morfismo descrito na Seção 2.6 foi incorporada ao termo referente à contribuição das arestas na definição de f_1. As associações que não

respeitam essa condição são penalizadas por f_1. Já a segunda condição do morfismo (Condição 2.3) é mais difícil de ser garantida, não sendo contemplada na definição de f_1. Os algoritmos de solução do PCGAD discutidos nos capítulos seguintes utilizam a função f_1 para guiar a busca da melhor solução.

Essa formulação permite que outras funções objetivo mais apropriadas ou que se beneficiem ainda mais da qualidade dos dados de entrada disponíveis possam ser facilmente adaptadas ao modelo, utilizando-se o mesmo conjunto de restrições.

No próximo capítulo será apresentado um algoritmo construtivo para gerar soluções viáveis para o PCGAD, respeitando as restrições discutidas neste capítulo.

Capítulo 4

Algoritmo construtivo

4.1 Introdução

Nesta seção é proposto um algoritmo construtivo para o PCGAD, que monta uma solução correspondente ao grafo bipartido $G' = (N_1 \cup N_2, E')$ definido no Capítulo 3, através da construção iterativa do conjunto E' de associações estabelecidas entre os vértices de $G_1 = (N_1, E_1)$ e $G_2 = (N_2, E_2)$. A função

$$f_1(G') = \frac{\alpha}{|N_1| \times |N_2|} \left[\sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|) \right] +$$

$$\frac{(1-\alpha)}{|E_1| \times |E_2|} \left[\sum_{(i,i') \in E_1} \sum_{(j,j') \in E_2} (1 - |max\{x_{ij}x_{i'j'}, x_{ij}x_{j'i'}\} - s^a((i,i'),(j,j'))|) \right]$$

cuja definição é reapresentada nesta seção, é computada progressivamente de acordo com a construção do conjunto E', que inicialmente é vazio. A inserção de cada nova aresta (i,j) em E' acarreta a atualização do primeiro termo de f_1 com o valor da contribuição da associação do vértice $i \in N_1$ ao vértice $j \in N_2$, como mostra o desenvolvimento a seguir:

$$\begin{aligned}
&\sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|) = \\
&= \sum_{(i,j) \in N_1 \cup N_2 - E'} (1 - |0 - s^v(i,j)|) + \sum_{(i,j) \in E'} (1 - |1 - s^v(i,j)|) = \\
&= \sum_{(i,j) \in N_1 \cup N_2} (1 - s^v(i,j)) - \sum_{(i,j) \in E'} (1 - s^v(i,j)) + \sum_{(i,j) \in E'} (1 - |1 - s^v(i,j)|) = \\
&= \sum_{(i,j) \in N_1 \cup N_2} (1 - s^v(i,j)) + \sum_{(i,j) \in E'} (2s^v(i,j) - 1).
\end{aligned}$$

Desta forma, o primeiro termo de f_1 pode ser calculado através do acréscimo ao

seu valor corrente de $(2s^v(i,j)-1)\cdot(\alpha/|N_1|\cdot|N_2|)$, para cada nova aresta (i,j) inserida em E'. No algoritmo construtivo apresentado a seguir, a inserção de uma aresta em E' é realizada na linha 18 e a atualização do primeiro termo de f_1, decorrente dessa inserção, é realizada na linha 20.

O segundo termo de f_1 é atualizado com o valor da contribuição das associações entre as arestas adjacentes ao vértice $i \in N_1$ e aquelas adjacentes ao nó $j \in N_2$ tais que $(i,j) \in E'$. Estas associações são representadas pelo conjunto $A' = \{((i,i'),(j,j'))|(i,j) \in E', (i',j') \in (\Gamma_i \times \Gamma_j) \cap E'\}$, onde Γ_i e Γ_j representam respectivamente os conjuntos de vértices adjacentes a $i \in N_1$ e a $j \in N_2$. O valor dessa contribuição é dado pelo acréscimo de $(2s^a((i,i'),(j,j'))-1)\cdot((1-\alpha)/|E_1|\cdot|E_2|)$ ao valor corrente de f_1, referente às associações estabelecidas entre as arestas adjacentes aos vértices i e j:

$$
\begin{aligned}
&\sum_{(i,i')\in E_1}\ \sum_{(j,j')\in E_2} (1 - |max\{x_{ij}x_{i'j'}, x_{ij}x_{j'i'}\} - s^a((i,i'),(j,j'))|) = \\
&= \sum_{((i,i'),(j,j'))\in E_1\times E_2 - A'} (1 - |0 - s^a((i,i'),(j,j'))|) \ + \\
&+ \sum_{((i,i'),(j,j'))\in A'} (1 - |1 - s^a((i,i'),(j,j'))|) = \\
&= \sum_{((i,i'),(j,j'))\in E_1\times E_2} (1 - |0 - s^a((i,i'),(j,j'))|) \ - \\
&- \sum_{((i,i'),(j,j'))\in A'} (1 - s^a((i,i'),(j,j'))) + \sum_{((i,i'),(j,j'))\in A'} s^a((i,i'),(j,j')) = \\
&= \sum_{((i,i'),(j,j'))\in E_1\times E_2} (1 - |0 - s^a((i,i'),(j,j'))|) - |A'| + \\
&+ \sum_{((i,i'),(j,j'))\in A'} 2s^a((i,i'),(j,j')).
\end{aligned}
$$

Como exemplo, a Figura 4.1 referente ao problema teste I_3 (cf. Seção 4.3.1) exibe as associações estabelecidas entre as arestas adjacentes aos nós $c \in N_1$ e $4 \in N_2$ a partir da aresta $(c,4) \in E'$. O cálculo da contribuição das associações entre arestas $\{((b,c),(4,5)),((c,d),(3,4)),((c,d),(4,9))\}$ é realizado nas linhas 21 a 27 do algoritmo construtivo.

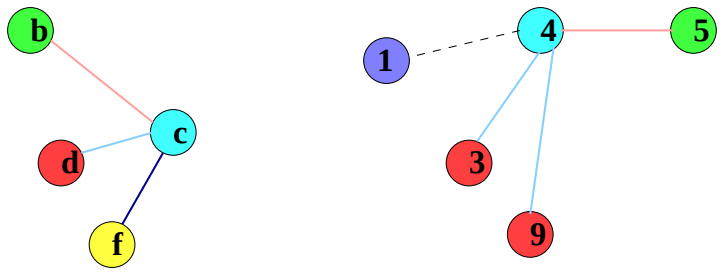


Figura 4.1: Associações entre arestas relativas à aresta $(c,4) \in E'$ (problema teste I_3)

O algoritmo construtivo é apresentado a seguir. Ele tem como objetivo gerar uma solução viável para o PCGAD. Por esse motivo, o processo de construção de uma solução é guiado pela tentativa de satisfação do conjunto de restrições do problema. Um número máximo de tentativas é permitido para a geração de soluções viáveis para o algoritmo. Se uma solução não viável é gerada, ela é descartada e uma nova tentativa é iniciada para a construção de uma nova solução. As variáveis c_v e c_a armazenam os valores correntes das contribuições de vértices e de arestas para a função f_1. Essas variáveis são inicializadas a cada nova tentativa de construção de uma nova solução $G' = (N_1 \cup N_2, E')$ com o valor dessas contribuições quando $E' = \emptyset$. O conjunto A_i, definido para cada vértice i de G_1, é formado por todos os vértices de N_2 associados a i em G'. Esses conjuntos são inicialmente vazios e atualizados a cada inserção de uma nova associação em E'. O algoritmo termina com $MaxSoluções$ viáveis geradas. O grafo bipartido G^* representa a solução $G' = (N_1 \cup N_2, E')$ com o melhor valor de f_1 (representado no algoritmo por f^*) dentre todas as $MaxSoluções$ geradas.

Algoritmo Construtivo ($MaxTentativas$, $MaxSoluções$)

Entrada: $G_1 = (N_1, E_1)$ e $G_2 = (N_2, E_2)$;
1. $n\_tentativas := 1$;
2. $n\_soluções := 1$;
3. $f^* := -\infty$;
4. enquanto ($n\_tentativas \leq MaxTentativas$ e $n\_soluções \leq MaxSoluções$)
5. $\quad G' = (N_1 \cup N_2, E')$, $E' := \emptyset$;
6. $\quad A_i := \emptyset$, $i = 1, \ldots, |N_1|$;
7. $\quad c_v := \alpha \cdot (\sum_{(i,j) \in N_1 \cup N_2} (1 - s^v(i,j)))/(|N_1| \cdot |N_2|)$;
8. $\quad c_a := (1 - \alpha) \cdot (\sum_{((i,i'),(j,j')) \in E_1 \times E_2} (1 - s^a((i,i'),(j,j'))))/(|E_1| \cdot |E_2|)$;
9. $\quad$ seja T_2 uma cópia do conjunto N_2;
10. $\quad$ enquanto ($T_2 \neq \emptyset$)
11. $\quad\quad$ selecione aleatoriamente $j \in T_2$;
12. $\quad\quad$ retire j de T_2;
13. $\quad\quad$ seja T_1 uma cópia do conjunto N_1;
14. $\quad\quad$ enquanto ($T_1 \neq \emptyset$ e o vértice $j \in N_2$ não tiver sido associado)
15. $\quad\quad\quad$ selecione aleatoriamente $i \in T_1$;
16. $\quad\quad\quad$ retire i de T_1;
17. $\quad\quad\quad$ se ($s^v(i,j) > 0$ e o grafo induzido em G_2 por $A_i \cup \{j\}$for conexo) então
18. $\quad\quad\quad\quad E' := E' \cup \{(i,j)\}$;
19. $\quad\quad\quad\quad A_i := A_i \cup \{j\}$;
20. $\quad\quad\quad\quad c_v := c_v + \alpha \cdot (2s^v(i,j) - 1)/(|N_1| \cdot |N_2|)$;
21. $\quad\quad\quad\quad$ para-todo ($i' \in \Gamma_i$)
22. $\quad\quad\quad\quad\quad$ para-todo ($j' \in \Gamma_j$)

23. se $((i', j') \in E')$ então
24. $c_a := c_a + (1 - \alpha) \cdot (2s^a((i, i'), (j, j')) - 1)/(|E_1| \cdot |E_2|)$;
25. fim-se
26. fim-para-todo
27. fim-para-todo
28. fim-se
29. fim-enquanto
30. fim-enquanto
31. $f_1 = c_v + c_a$;
32. se (todos os vértices de N_2 foram associados a algum vértice de N_1 e se cada vértice de N_1 está associado a pelo menos um vértice de N_2) então
33. $n\_soluções := n\_soluções + 1$;
34. se $(f_1 > f^*)$ então $f^* := f_1$ e $G^* := G'$;
35. fim-se
36. fim-se
37. $n\_tentativas := n\_tentativas + 1$;
38. fim-enquanto

Saída: G^*;

As inicializações do algoritmo são realizadas nas linhas 1 a 3. Como dados de entrada, consideram-se os grafos $G_1 = (N_1, E_1)$, $G_2 = (N_2, E_2)$ e os valores de similaridade. Um número $MaxTentativas$ de chances são dadas ao algoritmo, para que uma solução viável seja construída. O laço das linhas 4 a 38 é executado até que $MaxSoluções$ soluções viáveis sejam geradas ou até que $MaxTentativas$ tentativas sejam efetuadas para construir soluções viáveis. Uma solução viável corresponde ao grafo bipartido G', que inicialmente possui seu conjunto de arestas E' vazio (linha 5). As associações entre vértices são reinicializadas com $A_i = \emptyset, i = 1, \cdots, |N_1|$ (linha 6). A cada nova tentativa de construção de uma solução, as contribuições das associações entre vértices (c_v) e entre arestas (c_a) são reinicializadas com os valores da contribuição dos vértices e arestas para f_1 quando nenhuma associação entre os conjuntos de vértices dos dois grafos foi realizada (linhas 7 e 8). O laço das linhas 10 a 30 é executado enquanto existirem elementos de N_2 ainda não associados. Assim, para cada vértice j escolhido aleatoriamente no conjunto T_2 (linha 11), procura-se um vértice $i \in T_1$ de maneira que o valor de similaridade $s^v(i, j)$ seja não nulo e o subgrafo induzido em G_2 pelo conjunto A_i permaneça conexo com a inclusão do vértice j (linha 17). Se existe um vértice i que satisfaça estas condições, então a aresta (i, j) é adicionado a E' (linha 18), o vértice j é adicionado ao conjunto A_i (linha 19) e c_v é atualizada com a contribuição de (i, j) para f_1 (linha 20). As associações entre as arestas adjacentes à extremidade i e aquelas adjacentes à extremidade j são computadas e suas contribuições adicionadas a c_a apenas se as

associações entre seus vértices adjacentes formarem arestas de E' (linhas 21 a 27).

Se todos os vértices de N_2 foram associados a algum vértice de N_1 e se cada vértice de N_1 estiver associado a pelo menos um vértice de N_2, então a solução construída é viável e se constituirá em uma das $MaxSoluções$ geradas pelo algoritmo (linhas 32 a 36). Em seguida, o contador do número de tentativas é atualizado (linha 37) e uma nova tentativa de construção de uma solução que atenda os critérios de viabilidade será iniciada. A melhor das $MaxSoluções$ viáveis geradas (linha 34) é fornecida como resposta do algoritmo construtivo.

Com o objetivo de garantir a viabilidade da solução gerada, as restrições do PCGAD são tratadas da seguinte forma pelo algoritmo construtivo:

1. Todo vértice $j \in N_2$ é associado a um único vértice $i \in N_1$.

   A representação da solução através de um vetor de $|N_2|$ posições garante que um mesmo vértice j de N_2 não será associado a dois ou mais vértices distintos de N_1. No entanto, o algoritmo não garante que todos os vértices de N_2 sejam associados a algum vértice de N_1. De acordo com o processo de construção da solução, é possível que não existam candidatos no conjunto N_1 à associação com um dado j de N_2 que satisfaçam às condições de conexidade e de similaridade não nula. Neste caso, uma nova tentativa de construção é realizada.

2. Todo vértice $i \in N_1$ é associado a pelo menos um vértice $j \in N_2$.

   A determinação de cada aresta (i, j) de E' implica no exame de todo o conjunto N_1 para efetuar a escolha de um elemento i desse conjunto para ser associado com cada vértice j de N_2. Como um mesmo vértice $i \in N_1$ pode ser escolhido repetidas vezes como o melhor candidato para associar-se a diversos vértices de N_2, é possível que alguns vértices de N_1 não sejam selecionados para associação com nenhum vértice de N_2. Ao término da construção de uma solução, verifica-se se existe algum vértice $i \in N_1$ para o qual $A_i = \emptyset$. Se isso acontece, essa solução é descartada por não ser viável e uma nova solução G' é construída.

3. Associações são estabelecidas apenas entre vértices com valores de similaridade não nulos.

   O teste realizado na linha 17 do algoritmo construtivo garante a satisfação dessa restrição.

4. O subgrafo induzido em G_2 pelo conjunto A_i é conexo, $\forall i \in N_1$.

   O teste de conexidade aplicado ao subgrafo induzido em G_2 pelo conjunto $A_i \cup \{j\}$ (realizado na linha 17 do algoritmo) verifica se a inclusão de j em A_i preserva esta propriedade.

   Os conjuntos A_i são inicialmente vazios, para todo $i \in N_1$. Após cada iteração do laço das linhas 10-30 do algoritmo, o conjunto E' está parcialmente definido e os conjuntos A_i atualizados, garantindo-se para todos eles a preservação da condição de conexidade. Seja j o vértice a ser inserido em A_i, para algum $i \in N_1$. A inclusão do vértice j em A_i só é possível se j for adjacente a algum vértice de A_i em G_2, ou seja, se existe $v \in A_i$ tal que $(j, v) \in E_2$. Como o subgrafo induzido em G_2 por A_i é conexo, então aquele induzido em G_2 por $A_i \cup \{j\}$ pela inclusão de uma aresta no subgrafo também o será.

4.2 Complexidade do algoritmo

A complexidade de construção de uma solução viável para o PCGAD no algoritmo proposto neste capítulo é $O(|N_1|^2 \cdot |N_2|^2)$, como mostrado a seguir.

A complexidade dos cálculos das contribuições dos vértices e das arestas quando nenhuma associação foi realizada é $O(|N_1| \cdot |N_2| + |E_1| \cdot |E_2|)$. A inicialização dos conjuntos A_i, para cada vértice $i \in N_1$ é $O(|N_1|)$. Todos os elementos de N_2 devem ser selecionados para que o laço das linhas 10 a 30 termine. Para cada um desses elementos, N_1 vértices são avaliados como candidatos à associação. Assim, $|N_2| \cdot |N_1|$ testes são necessários. Para cada um desses testes, é preciso verificar as condições de viabilidade. O custo mais alto neste caso é atribuído à condição de conexidade. No pior caso, o conjunto A_i possui $|N_2| - 1$ elementos e verificar se a inclusão de j ao conjunto o mantém conexo significa verificar se j é terminal de uma aresta formada com algum elemento de A_i ($O(|N_2|)$). Como a condição de conexidade é verificada para cada um dos $|N_2| \cdot |N_1|$ testes anteriores, então sua complexidade é de $O(|N_2| \cdot |N_1| \cdot |N_2|)$. Se as condições de viabilidade são satisfeitas, então atualizações devem ser realizadas. O custo maior neste caso é devido à atualização da contribuição das arestas para f_1 que é $O(|N_1| \cdot |N_2|)$. Assim, o custo de construção de uma solução é de $O(|N_1|^2 \cdot |N_2|^2)$.

A verificação das condições de viabilidade descrita na linha 32 é $O(|N_1| + |N_2|)$.

No pior caso, $MaxTentativas$ são realizadas pelo algoritmo, em busca de uma solução viável para o problema. Desta forma, a complexidade total do algoritmo construtivo é dada por $O(MaxTentativas.(|N_1|^2.|N_2|^2+|N_1|+|N_2|)) = O(MaxTentativas \cdot |N_1|^2 \cdot |N_2|^2)$.

4.3 Resultados computacionais

O algoritmo construtivo foi implementado na linguagem C. O compilador utilizado foi o *gcc*, versão 2.96. A representação de uma solução do PCGAD foi implementada como um vetor de inteiros de $|N_2|$ posições. Um vetor de $|N_1|$ ponteiros para listas encadeadas armazenam os elementos de todos os conjuntos A_i para $i = 1, \ldots, |N_1|$. A cada associação do tipo $(i, j) \in E'$ adicionada à solução, a lista encadeada da posição i do vetor de ponteiros é atualizada com a inclusão do elemento j. Os conjuntos N_1 e N_2 são implementados como listas encadeadas. A rotina de geração de números aleatórios utilizada é aquela proposta em [59].

Os experimentos computacionais foram realizados com 12 problemas teste em um microcomputador Pentium II 450 MHz, sob o sistema operacional Linux Red Hat, versão 7.1.

4.3.1 Problemas teste

O estudo do PCGAD foi motivado por aplicações reais, não existindo uma biblioteca de problemas teste. Neste trabalho foram utilizados 12 problemas teste do PCGAD com o objetivo de analisar os resultados dos algoritmos apresentados. Os problemas teste I_0 e I_1 correspondem aos Exemplos 1 e 2 apresentados no Capítulo 3. Os problemas teste I_2, I_3 e I_4 também foram construídos artificialmente com valores de similaridade entre vértices e entre arestas gerados aleatoriamente no intervalo [0,1]. Os grafos G_1 e G_2 relativos a esses problemas teste são apresentados na Figura 4.2, assim como a correspondência que maximiza o valor de f_1 (com $\alpha = 0{,}5$) para esses problemas teste.

Os problemas teste I_5, I_8 e I_9, também gerados artificialmente, foram fornecidos por Bengoetxea [2]. O problema teste I_8 foi utilizado em [2]. Existem vértices isolados nos grafos dos problemas teste I_5 e I_8. São eles os vértices 0 e 5 do grafo G_2 do problema teste I_5 e os vértices 5, 15 e 21 do grafo G_1 do problema teste I_8.

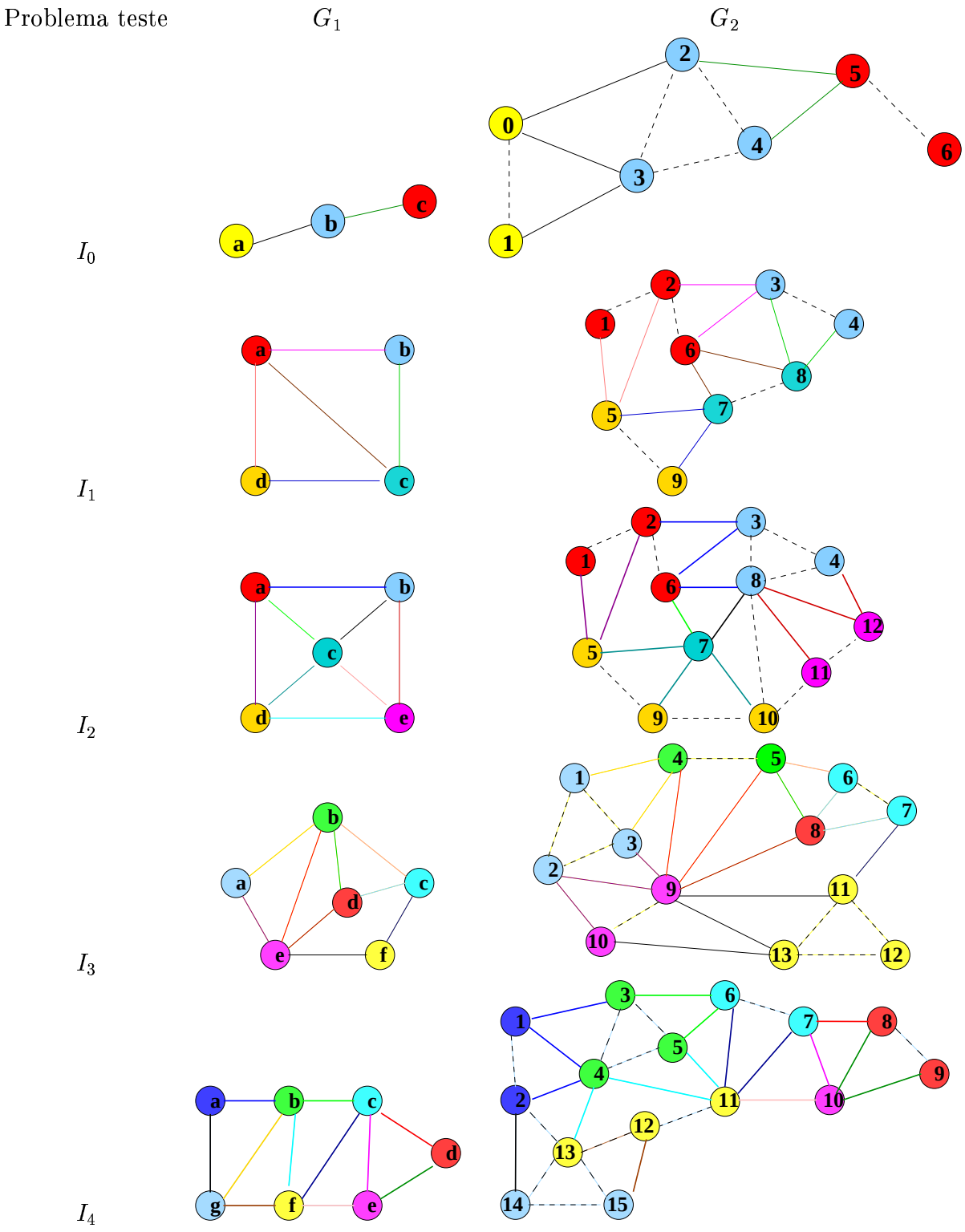


Figura 4.2: Problemas teste I_0, I_1, I_2, I_3 e I_4 e suas respectivas soluções ótimas

Foram criados os problemas teste I_{5a} e I_{8a}, obtidos respectivamente a partir de I_5 e I_8, com grafos sem vértices isolados.

Os problemas teste I_6 e I_7 foram construídos a partir da descrição de imagens.

Esses problemas teste foram fornecidos por Perchant e Bengoetxea [49]. O problema teste I_6 foi construído a partir de imagens do cérebro humano, como as apresentadas na Figura 4.3. O problema teste I_7 foi produzido para os testes computacionais realizados em [52] e foi construído a partir das imagens 2D apresentadas na Figura 4.4. A imagem da Figura 4.4-(a) foi supersegmentada em 28 regiões (Figura 4.4-(c)) e comparada com uma imagem modelo de 14 regiões bem definidas (Figura 4.4-(b)). O grafo G_1 que representa o modelo tem 14 vértices e 27 arestas e o grafo G_2 que representa a imagem supersegmentada tem 28 vértices e 63 arestas. Eles são apresentados na Figura 4.5, em conjunto com a solução mais adequada para representar o reconhecimento da imagem (Figura 4.4-(c)) em relação ao modelo (Figura 4.4-(a)). No cálculo de similaridades entre vértices foram usados atributos como o nível de cinza, enquanto no cálculo de similaridade entre arestas foram usados atributos como distância difusa e adjacência difusa.

(a) (b) (c)

Figura 4.3: Cérebro humano: (a) é a imagem original, (b) é o atlas, e (c) é uma imagem supersegmentada.

(a) (b) (c)

Figura 4.4: Problema teste I_7, associado ao corte de um músculo: (a) é a imagem original, (b) é o atlas, e (c) é a imagem supersegmentada.

4.3.2 Resultados experimentais

As Tabelas 4.1, 4.2 e 4.3 mostram os resultados obtidos pela aplicação do algoritmo construtivo aos problemas teste I_0 a I_9, I_{5a} e I_{8a}. Os blocos de valores

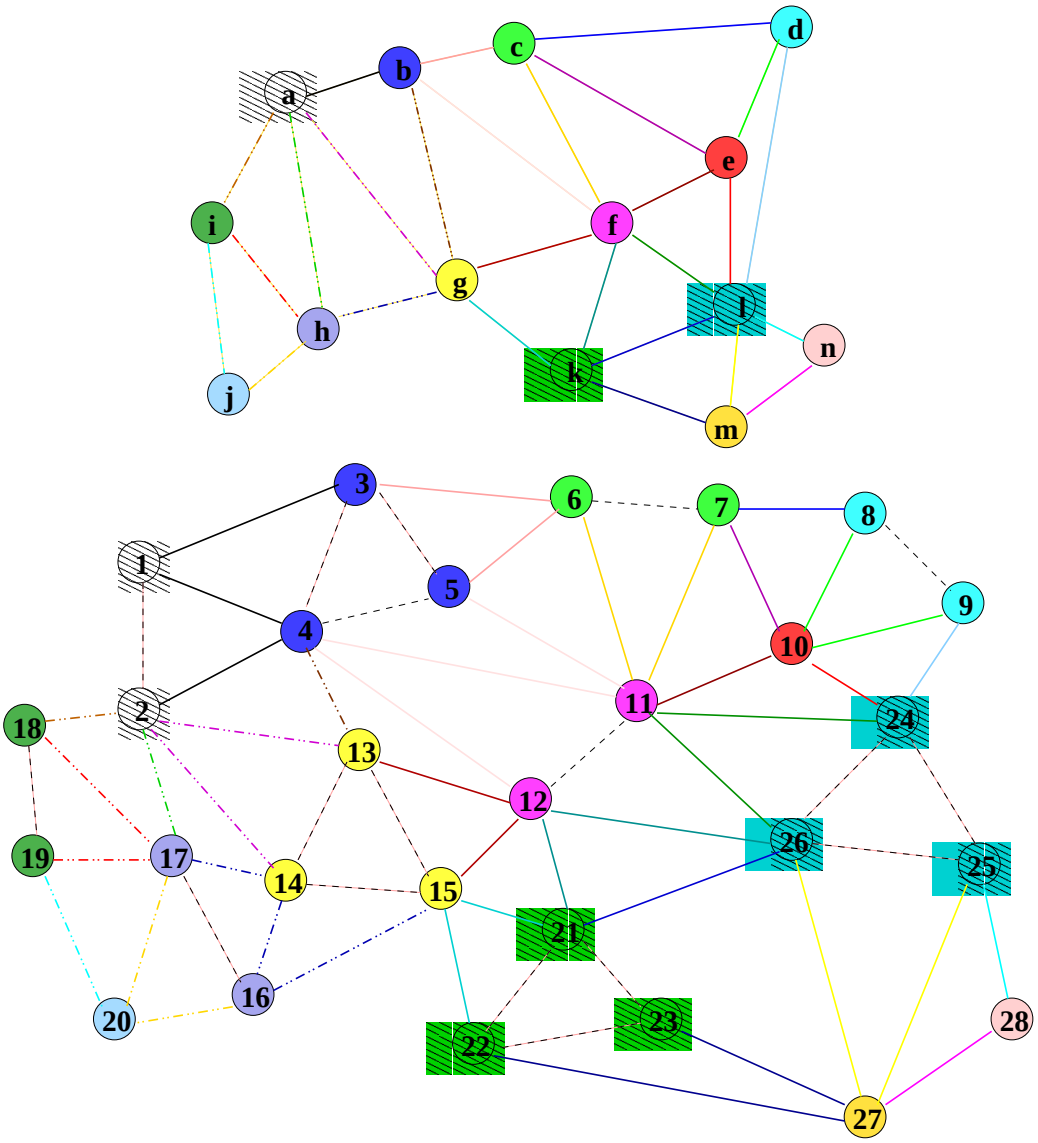


Figura 4.5: Problema teste I_7: Solução mais adequada

denotados por AC_1 (Tabela 4.1), AC_{10} (Tabela 4.2) e AC_{100} (Tabela 4.3) são relativos aos resultados obtidos pelo algoritmo construtivo respectivamente quando são geradas, no máximo, 1, 10 ou 100 solucões viáveis. Em cada um dos blocos, a coluna tentativas indica o índice da tentativa na qual foi encontrada a melhor solução viável dentre as $MaxSoluções$ geradas. A coluna NSV mostra a posição da solução apresentada pelo algoritmo na seqüência de soluções viáveis geradas. São também fornecidos os números de vértices e de arestas dos grafos $G_1 = (N_1, E_1)$ e $G_2 = (N_2, E_2)$ de cada problema teste. Os valores de α testados foram $0,1$, $0,5$ e $0,9$. O parâmetro $MaxTentativas$ foi fixado em 500. Para cinco (I_0, I_1, I_2, I_3 e I_4) dos problemas teste, os valores ótimos da função f_1 foram calculados pelo exame

exaustivo de todas as associações possíveis (as soluções correspondentes para $\alpha =$ 0,5 foram apresentadas na Figura 4.2). O tempo médio em segundos TM gasto pelo algoritmo em cada tentativa de construção também é apresentado.

Os valores de f_1 referentes aos resultados apresentados em AC_1 para os problemas teste em que se conhece as soluções ótimas estão respectivamente a 0%, 4,9%, 6,1% e 12,2% ($\alpha = 0{,}1$), 0%, 2,8%, 2,2%, 7,8% e 8,2% ($\alpha = 0{,}5$) e 0%, 2,1%, 1,8%, 4,4% e 6,8% ($\alpha = 0{,}9$) do valor ótimo. Esses mesmos valores, relativos às Tabelas 4.2 e 4.3, decrescem respectivamente para 0%, 4,5%, 4,4% e 7,0% ($\alpha = 0{,}1$), 0%, 1,5%, 2,2%, 5,6% e 3,8% ($\alpha = 0{,}5$) e 0%, 0,5%, 1,8%, 4,4% e 2,4% ($\alpha = 0{,}9$) e para 0%, 0,4%, 1,3% e 7,0% ($\alpha = 0{,}1$), 0%, 1,3%, 0,2%, 5,6% e 2,9% ($\alpha = 0{,}5$) e 0%, 0,3%, 0,5%, 2,5% e 1,4% ($\alpha = 0{,}9$) do valor ótimo. Quanto maior o valor de α, mais próxima do ótimo é a solução obtida pelo algoritmo construtivo. Como esperado, a qualidade das soluções obtidas melhora de acordo com o aumento do número de soluções viáveis geradas. O algoritmo se mostrou bastante rápido, já que para a maioria dos problemas teste, poucas tentativas foram necessárias para se encontrar a primeira solução viável e cada iteração não gastou mais do que 0,01 segundo.

A Tabela 4.4 mostra os percentuais de aumento no valor de f_1, calculados a partir da comparação dos resultados de AC_1 e AC_{10} e também de AC_{10} e AC_{100}. As soluções obtidas por AC_{10} foram em média 1,4% ($\alpha = 0{,}1$), 1,1% ($\alpha = 0{,}5$) e 1,2% ($\alpha = 0{,}9$) melhores que aquelas obtidas por AC_1. As soluções obtidas por AC_{100} foram em média 1% ($\alpha = 0{,}1$), 0,4% ($\alpha = 0{,}5$) e 0,6% ($\alpha = 0{,}9$) melhores que as soluções obtidas por AC_{10}. Observa-se que o ganho de qualidade na solução gerada foi maior, em média, no primeiro caso (AC_1 – AC_{10}) do que no segundo (AC_{10} – AC_{100}). Isto ocorre mesmo para os testes com $\alpha = 0{,}1$, que apresentam soluções com ganhos percentuais maiores no caso AC_{10} – AC_{100} para um número maior de problemas teste (I_1, I_2, I_6, I_{8a} e I_7 a I_9). O menor ganho na qualidade da solução gerada pelo algoritmo pode não justificar o esforço computacional da geração de um número muito elevado de soluções viáveis.

A Figura 4.6 mostra os resultados obtidos pelo algoritmo construtivo (AC_1 e AC_{100}) para o problema teste I_3, considerando $\alpha = 0{,}5$. Observa-se na primeira solução que sete (1, 5, 8, 10, 11, 12 e 13) dos 13 vértices foram corretamente associados, quando comparados à solução ótima (vide Figura 4.2) desse problema teste. Na segunda solução, oito vértices (1, 2, 6, 7, 8, 9, 11 e 13) foram corretamente associados. Ao priorizar a satisfação das condições de viabilidade, o algoritmo construtivo

Problema teste	N_1	E_1	N_2	E_2	f_1^*	AC_1		
						tentativas	f_1	TM
I_0 $(\alpha = 0,1)$	3	2	7	10	0,3240	1	0,3240	-
I_1	4	5	9	15	0,2707	1	0,2574	-
I_2	5	8	12	23	0,2818	1	0,2646	-
I_3	6	9	13	25	0,5603	1	0,4921	-
I_4	7	11	15	29	n.d.	3	0,5166	-
I_5	10	15	30	39	n.d.	297	0,4908	-
I_{5a}	10	15	30	41	n.d.	9	0,5106	0,001
I_6	12	42	95	1434	n.d.	1	0,3922	-
I_7	14	27	28	63	n.d.	5	0,1226	0,002
I_8	30	39	100	297	n.d.	26	0,5020	0,002
I_{8a}	30	42	100	297	n.d.	26	0,5339	0,002
I_9	50	88	250	1681	n.d.	1	0,5001	0,005
I_0 $(\alpha = 0,5)$	3	2	7	10	0,6142	1	0,6142	-
I_1	4	5	9	15	0,5202	1	0,5055	-
I_2	5	8	12	23	0,5098	1	0,4986	-
I_3	6	9	13	25	0,6375	1	0,5878	-
I_4	7	11	15	29	0,6754	3	0,6200	-
I_5	10	15	30	39	n.d.	297	0,5038	-
I_{5a}	10	15	30	41	n.d.	9	0,5044	-
I_6	12	42	95	1434	n.d.	1	0,4022	-
I_7	14	27	28	63	n.d.	5	0,3704	-
I_8	30	39	100	297	n.d.	26	0,4999	0,002
I_{8a}	30	42	100	297	n.d.	26	0,5176	0,002
I_9	50	88	250	1681	n.d.	1	0,5025	0,005
I_0 $(\alpha = 0,9)$	3	2	7	10	0,9045	1	0,9045	-
I_1	4	5	9	15	0,7698	1	0,7535	-
I_2	5	8	12	23	0,7465	1	0,7327	-
I_3	6	9	13	25	0,7147	1	0,6835	-
I_4	7	11	15	29	0,7761	3	0,7235	-
I_5	10	15	30	41	n.d.	297	0,5168	-
I_{5a}	10	15	30	41	n.d.	9	0,4981	0,001
I_6	12	42	95	1434	n.d.	1	0,4122	-
I_7	14	27	28	63	n.d.	5	0,6182	-
I_8	30	39	100	297	n.d.	26	0,4978	0,002
I_{8a}	30	42	100	297	n.d.	26	0,5014	0,002
I_9	50	88	250	1681	n.d.	1	0,5049	0,010

n.d.: não disponível

- : tempo de processamento inferior a 10^{-3} segundo

Tabela 4.1: Resultados obtidos pelo algoritmo construtivo, quando uma solução viável é gerada (AC_1).

Problema teste	N_1	E_1	N_2	E_2		AC_{10}			
					f_1^*	tentativas	NSV	f_1	TM
I_0 $(\alpha = 0,1)$	3	2	7	10	0,3240	1	1	0,3240	-
I_1	4	5	9	15	0,2707	10	6	0,2586	-
I_2	5	8	12	23	0,2818	6	4	0,2694	-
I_3	6	9	13	25	0,5603	7	7	0,5213	-
I_4	7	11	15	29	n.d.	26	7	0,5485	-
I_5	10	15	30	39	n.d.	451	2	0,4941	-
I_{5a}	10	15	30	41	n.d.	36	8	0,5173	-
I_6	12	42	95	1434	n.d.	8	8	0,3923	0,001
I_7	14	27	28	63	n.d.	11	4	0,1236	-
I_8	30	39	100	297	n.d.	26	1	0,5020	0,002
I_{8a}	30	42	100	297	n.d.	123	6	0,5339	0,002
I_9	50	88	250	1681	n.d.	18	7	0,5001	0,010
I_0 $(\alpha = 0,5)$	3	2	7	10	0,6142	1	1	0,6142	-
I_1	4	5	9	15	0,5202	10	6	0,5123	-
I_2	5	8	12	23	0,5098	8	6	0,4987	-
I_3	6	9	13	25	0,6375	7	7	0,6020	-
I_4	7	11	15	29	0,6754	20	5	0,6501	-
I_5	10	15	30	39	n.d.	297	1	0,5038	-
I_{5a}	10	15	30	41	n.d.	27	4	0,5171	-
I_6	12	42	95	1434	n.d.	8	8	0,4037	0,001
I_7	14	27	28	63	n.d.	32	10	0,3735	-
I_8	30	39	100	297	n.d.	123	6	0,5015	0,002
I_{8a}	30	42	100	297	n.d.	123	6	0,5193	0,002
I_9	50	88	250	1681	n.d.	18	7	0,5030	0,010
I_0 $(\alpha = 0,9)$	3	2	7	10	0,9045	1	1	0,9045	-
I_1	4	5	9	15	0,7698	10	6	0,7660	-
I_2	5	8	12	23	0,7465	1	1	0,7327	-
I_3	6	9	13	25	0,7147	1	1	0,6835	-
I_4	7	11	15	29	0,7761	33	10	0,7578	-
I_5	10	15	30	41	n.d.	297	1	0,5168	-
I_{5a}	10	15	30	41	n.d.	27	4	0,5188	-
I_6	12	42	95	1434	n.d.	2	2	0,4150	0,001
I_7	14	27	28	63	n.d.	32	10	0,6252	-
I_8	30	39	100	297	n.d.	123	6	0,5011	0,002
I_{8a}	30	42	100	297	n.d.	123	6	0,5047	0,002
I_9	50	88	250	1681	n.d.	18	7	0,5059	0,010

n.d.: não disponível

- : tempo de processamento inferior a 10^{-3} segundo

Tabela 4.2: Resultados obtidos pelo algoritmo construtivo, quando dez soluções viáveis são geradas (AC_{10}).

Problema teste	N_1	E_1	N_2	E_2	f_1^*	AC_{100} tentativas	NSV	f_1	TM
I_0 $(\alpha = 0,1)$	3	2	7	10	0,3240	1	1	0,3240	-
I_1	4	5	9	15	0,2707	99	72	0,2696	-
I_2	5	8	12	23	0,2818	117	78	0,2780	-
I_3	6	9	13	25	0,5603	7	7	0,5213	-
I_4	7	11	15	29	n.d.	206	71	0,5519	-
I_5	10	15	30	39	n.d.	451	2	0,4941	-
I_{5a}	10	15	30	41	n.d.	237	24	0,5215	-
I_6	12	42	95	1434	n.d.	11	11	0,3927	0,001
I_7	14	27	28	63	n.d.	198	46	0,1269	-
I_8	30	39	100	297	n.d.	436	37	0,5024	0,002
I_{8a}	30	42	100	297	n.d.	292	22	0,5343	0,002
I_9	50	88	250	1681	n.d.	86	33	0,5002	0,010
I_0 $(\alpha = 0,5)$	3	2	7	10	0,6142	1	1	0,6142	-
I_1	4	5	9	15	0,5202	99	72	0,5135	-
I_2	5	8	12	23	0,5098	50	36	0,5090	-
I_3	6	9	13	25	0,6375	7	7	0,6020	-
I_4	7	11	15	29	0,6754	206	71	0,6559	-
I_5	10	15	30	39	n.d.	297	1	0,5038	-
I_{5a}	10	15	30	41	n.d.	417	47	0,5223	-
I_6	12	42	95	1434	n.d.	40	38	0,4045	0,001
I_7	14	27	28	63	n.d.	34	11	0,3757	-
I_8	30	39	100	297	n.d.	292	22	0,5023	0,002
I_{8a}	30	42	100	297	n.d.	292	22	0,5200	0,002
I_9	50	88	250	1681	n.d.	207	96	0,5031	0,010
I_0 $(\alpha = 0,9)$	3	2	7	10	0,9045	1	1	0,9045	-
I_1	4	5	9	15	0,7698	41	32	0,7672	-
I_2	5	8	12	23	0,7465	50	36	0,7427	-
I_3	6	9	13	25	0,7147	67	49	0,6966	-
I_4	7	11	15	29	0,7761	43	13	0,7655	-
I_5	10	15	30	41	n.d.	297	1	0,5168	-
I_{5a}	10	15	30	41	n.d.	417	47	0,5243	-
I_6	12	42	95	1434	n.d.	40	38	0,4168	0,001
I_7	14	27	28	63	n.d.	34	11	0,6282	-
I_8	30	39	100	297	n.d.	292	22	0,5022	0,002
I_{8a}	30	42	100	297	n.d.	292	22	0,5058	0,002
I_9	50	88	250	1681	n.d.	207	96	0,5060	0,010

n.d.: não disponível

- : tempo de processamento inferior a 10^{-3} segundo

Tabela 4.3: Resultados obtidos pelo algoritmo construtivo, quando 100 soluções viáveis são geradas (AC_{100}).

	Ganho percentual (%)	
Problema teste	$AC_1 - AC_{10}$	$AC_{10} - AC_{100}$
I_0 ($\alpha = 0{,}1$)	0,000	0,000
I_1	0,466	4,254
I_2	1,814	3,192
I_3	5,934	0,000
I_4	6,175	0,620
I_5	0,672	0,000
I_{5a}	1,312	0,812
I_6	0,026	0,102
I_7	0,816	2,670
I_8	0,000	0,080
I_{8a}	0,000	0,075
I_9	0,000	0,020
Média	1,503	0,985
I_0 ($\alpha = 0{,}5$)	0,000	0,000
I_1	1,345	0,234
I_2	0,020	2,065
I_3	2,416	0,000
I_4	4,855	0,892
I_5	0,000	0,000
I_{5a}	2,518	1,006
I_6	0,373	0,198
I_7	0,837	0,589
I_8	0,320	0,160
I_{8a}	0,328	0,135
I_9	0,100	0,020
Média	1,093	0,442
I_0 ($\alpha = 0{,}9$)	0,000	0,000
I_1	1,659	0,157
I_2	0,000	1,365
I_3	0,000	1,917
I_4	4,741	1,016
I_5	0,000	0,000
I_{5a}	4,156	1,060
I_6	0,679	0,434
I_7	1,132	0,480
I_8	0,663	0,220
I_{8a}	0,658	0,218
I_9	0,198	0,020
Média	1,157	0,574

Tabela 4.4: Percentuais de aumento no valor de f_1 em função do aumento de $MaxSoluções$

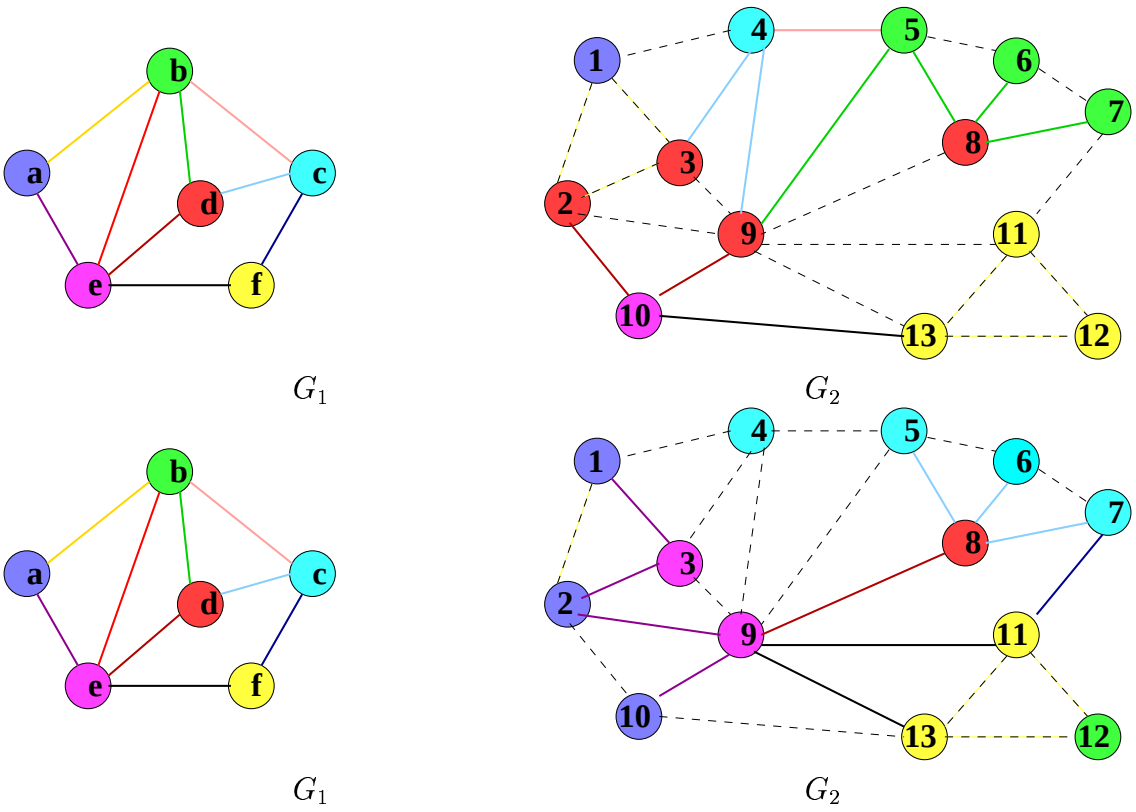


Figura 4.6: Soluções obtidas pelo algoritmo construtivo para o problema teste I_3.

não necessariamente conduz a construção de uma solução a maximizar o número de associações entre arestas. Como consequência, a estrutura do grafo G_1 pode não ser totalmente preservada em G_2, eventualmente acarretando a associação incorreta das posições relativas dos vértices de G_1 em G_2. Quanto mais próxima a solução está do ótimo, maior é a preservação da estrutura de G_1 em G_2.

As Figuras 4.7 a 4.11 mostram as diferentes soluções obtidas pelo algoritmo construtivo para o problema teste I_7, respectivamente para os casos (a) AC_1 com $\alpha = 0{,}1$, $\alpha = 0{,}5$ e $\alpha = 0{,}9$ (Figura 4.7), (b) AC_{10} com $\alpha = 0{,}1$ (Figura 4.8), (c) AC_{10} com $\alpha = 0{,}5$ e $\alpha = 0{,}9$ (Figura 4.9), (d) AC_{100} com $\alpha = 0{,}1$ (Figura 4.10) e (e) AC_{100} com $\alpha = 0{,}5$ e $\alpha = 0{,}9$ (Figura 4.11). Ao compará-las com a solução mais adequada para esse problema teste (apresentada na Figura 4.5), observa-se que quatro vértices (1, 10, 24 e 26) são associados corretamente no caso (a), dois vértices (6 e 12) no caso (b), cinco vértices (1, 2, 16, 17 e 24) no caso (c), 12 vértices (8, 11, 13, 14, 15, 17, 20, 22, 23, 25, 26 e 27) no caso (d) e, finalmente, sete vértices (1, 2, 8, 9, 11, 12 e 22) no caso (e). Os resultados indicam que o algoritmo construtivo gera melhores soluções para o problema teste I_7 quando 100 soluções iniciais são geradas (caso AC_{100}) e que o algoritmo é sensível à variação de α. Nestes testes, o maior

número de associações corretas entre os vértices foi encontrado no caso $\alpha = 0{,}1$, ou seja, quando o peso relativo das associações entre as arestas é maior.

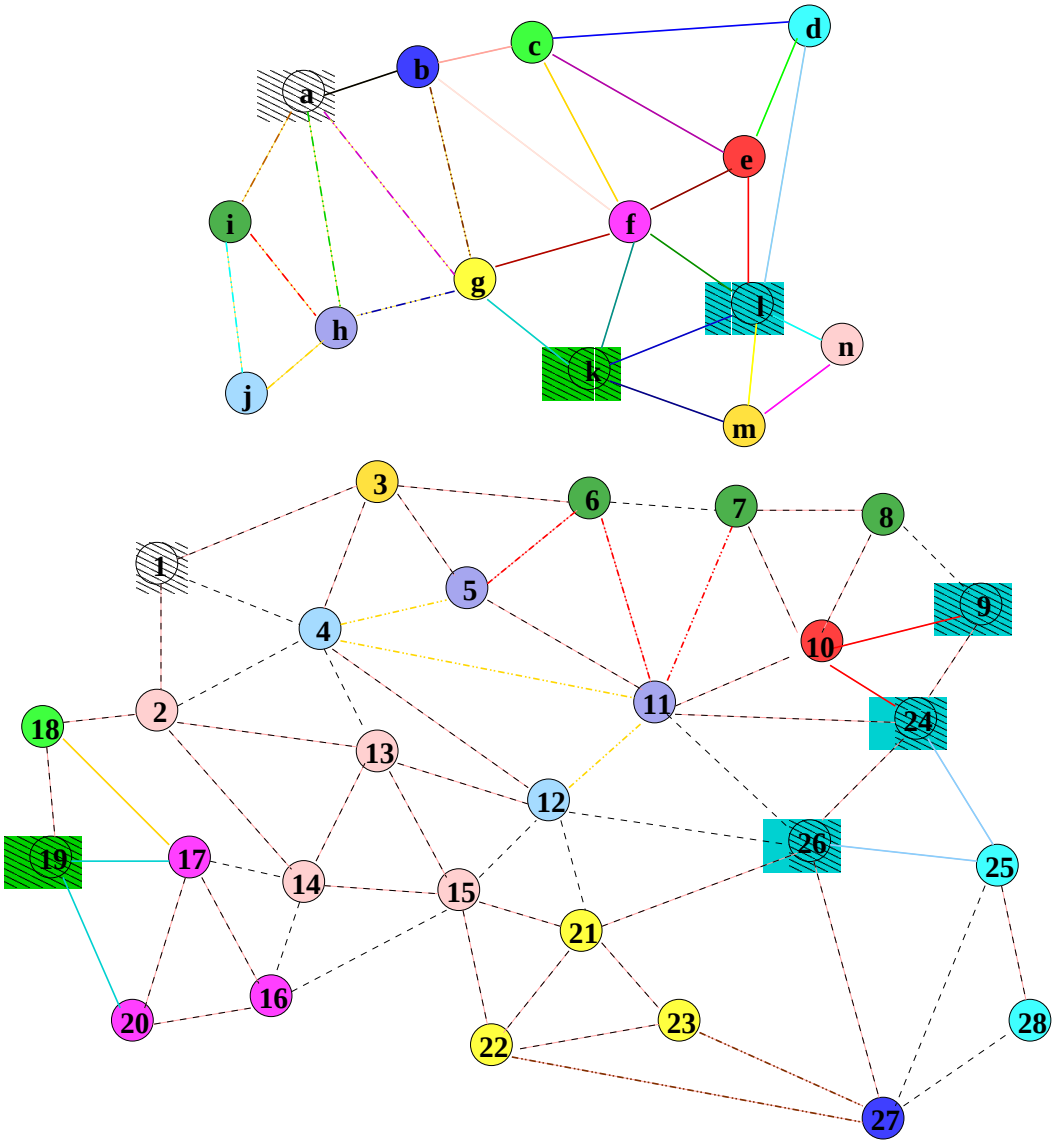


Figura 4.7: Problema teste I_7, caso AC_1 com $\alpha = 0{,}1$, $\alpha = 0{,}5$ e $\alpha = 0{,}9$

Em seis (I_0 a I_3, I_6 e I_9) dos 12 problemas teste uma solução viável foi obtida na primeira tentativa de construção. O maior número de tentativas necessárias para encontrar a primeira solução viável ocorreu para os problemas teste I_5 e I_8, que possuem vértices isolados. No problema teste I_5, os vértices 0 e 5 estão isolados do grafo G_2. Neste caso, para se obter uma solução viável, os conjuntos A_i associados a esses vértices só podem ser unitários. Caso contrário, os grafos induzidos por tais conjuntos seriam não conexos, violando um dos critérios de viabilidade. No problema teste I_8, os vértices 5, 15 e 21 do grafo G_1 estão isolados. Pode-se garantir

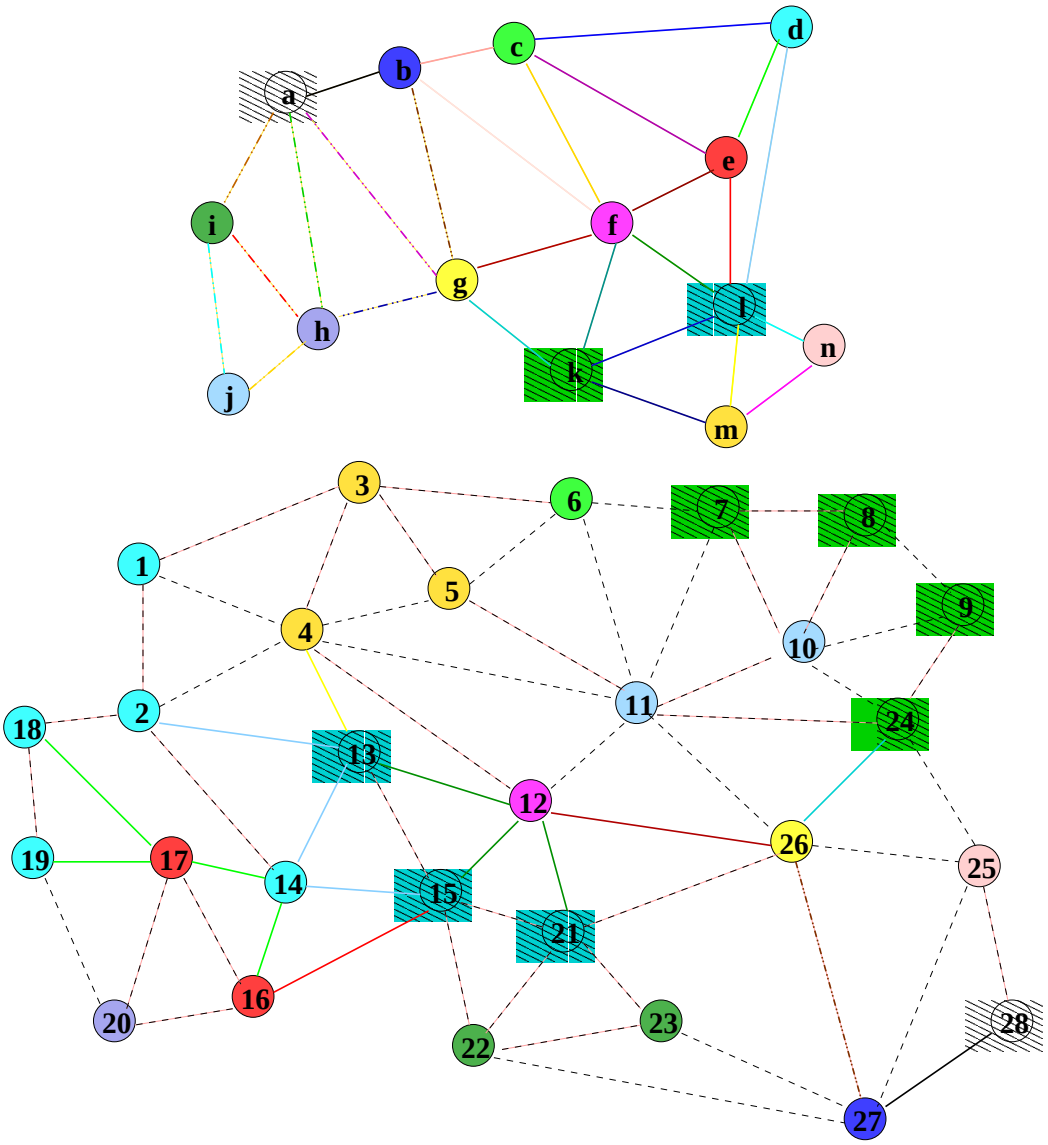


Figura 4.8: Problema teste I_7, caso AC_{10} com $\alpha = 0{,}1$

que os grafos induzidos em G_2 por A_5, A_{15} e A_{21} são conexos, porém não deve existir aresta alguma em G_2, ou seja, adjacente a vértices desses conjuntos e associada a alguma aresta de G_1, pois senão os vértices 5, 15 e 21 não estariam isolados.

Embora seja possível a geração de soluções viáveis pelo algoritmo construtivo para problemas teste com grafos não conexos, neste caso a busca por uma solução G' viável é mais longa e explica o número elevado de tentativas para obtenção da solução. Os problemas teste I_{5a} e I_{8a} foram geradas a partir de I_5 e I_8, acrescentando-se arestas para conectar os vértices isolados. Foram escolhidos valores baixos de similaridade entre essas arestas e aquelas de G_2, com o objetivo de não alterar as informações já estabelecidas entre os grafos dos problemas teste originais. Os

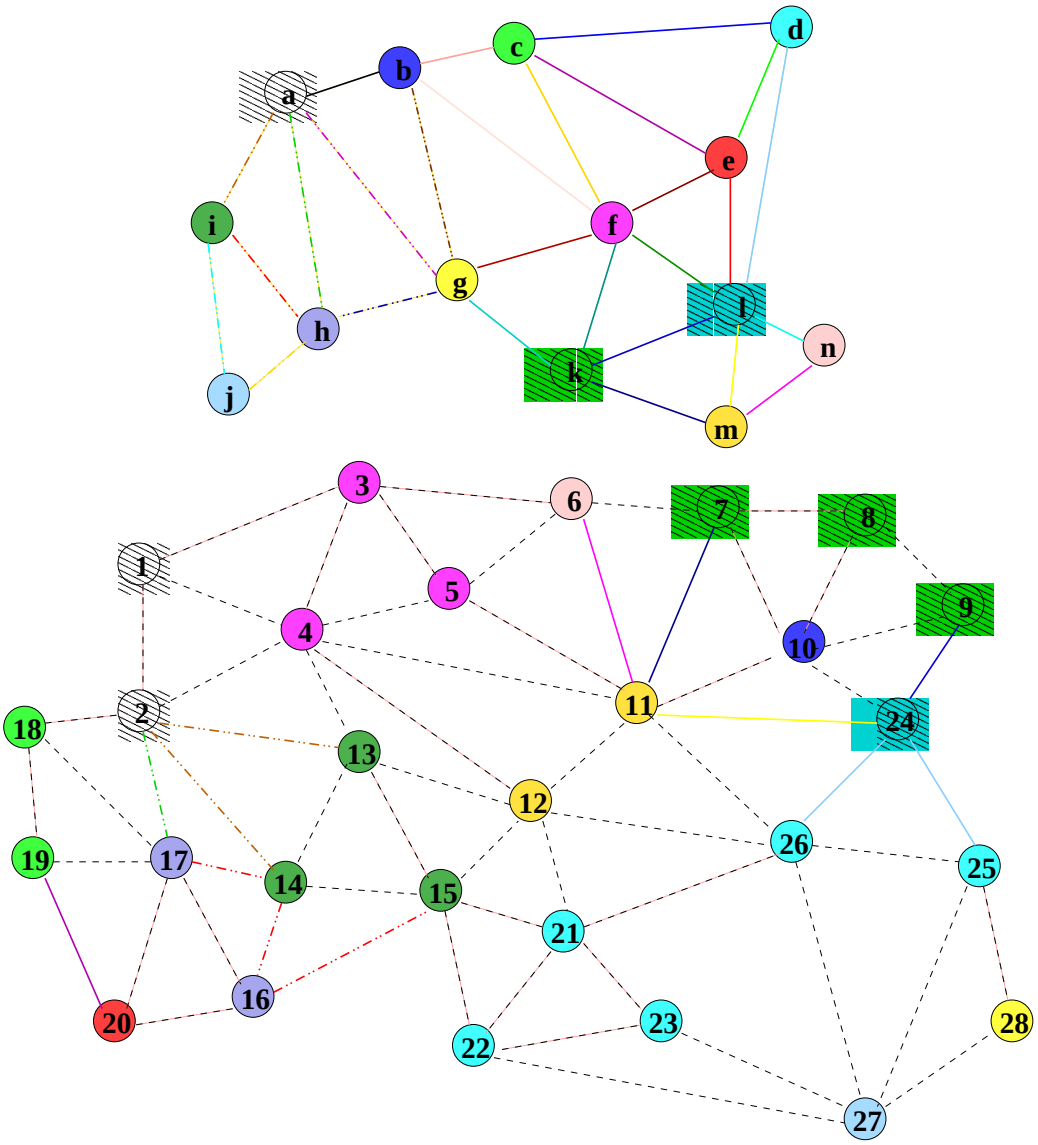


Figura 4.9: Problema teste I_7, caso AC_{10} com $\alpha = 0{,}5$ e $\alpha = 0{,}9$

resultados obtidos para esses problemas teste (Tabelas 4.1, 4.2 e 4.3) indicam que o algoritmo converge para uma solução viável com poucas tentativas de construção quando os grafos comparados são conexos.

O algoritmo construtivo proposto neste capítulo mostra-se uma boa ferramenta para a geração rápida de soluções viáveis de boa qualidade. No próximo capítulo apresenta-se um algoritmo de busca local que permite melhorar a solução construída pelo algoritmo discutido neste capítulo.

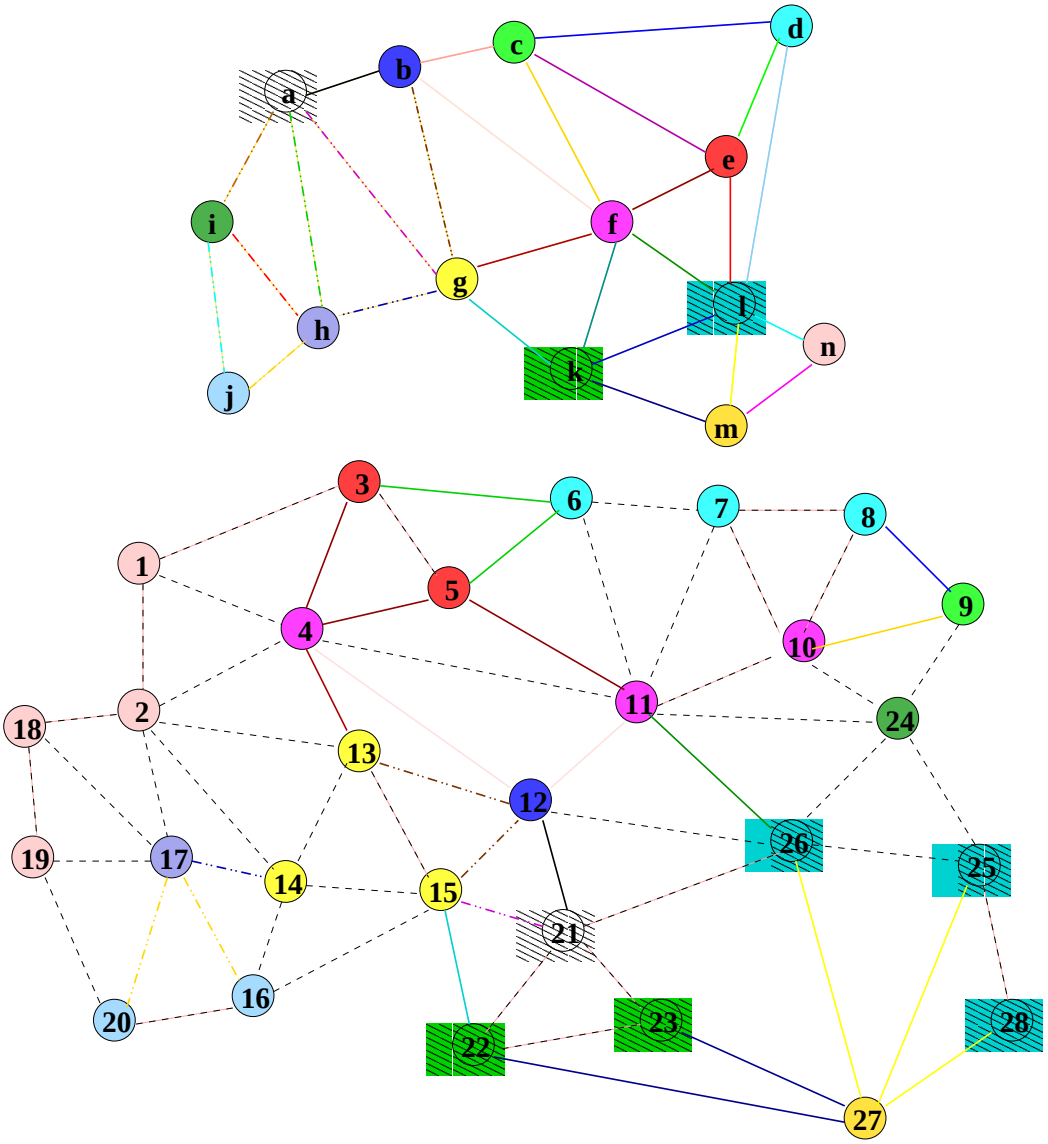


Figura 4.10: Problema teste I_7, caso AC_{100} com $\alpha = 0{,}1$

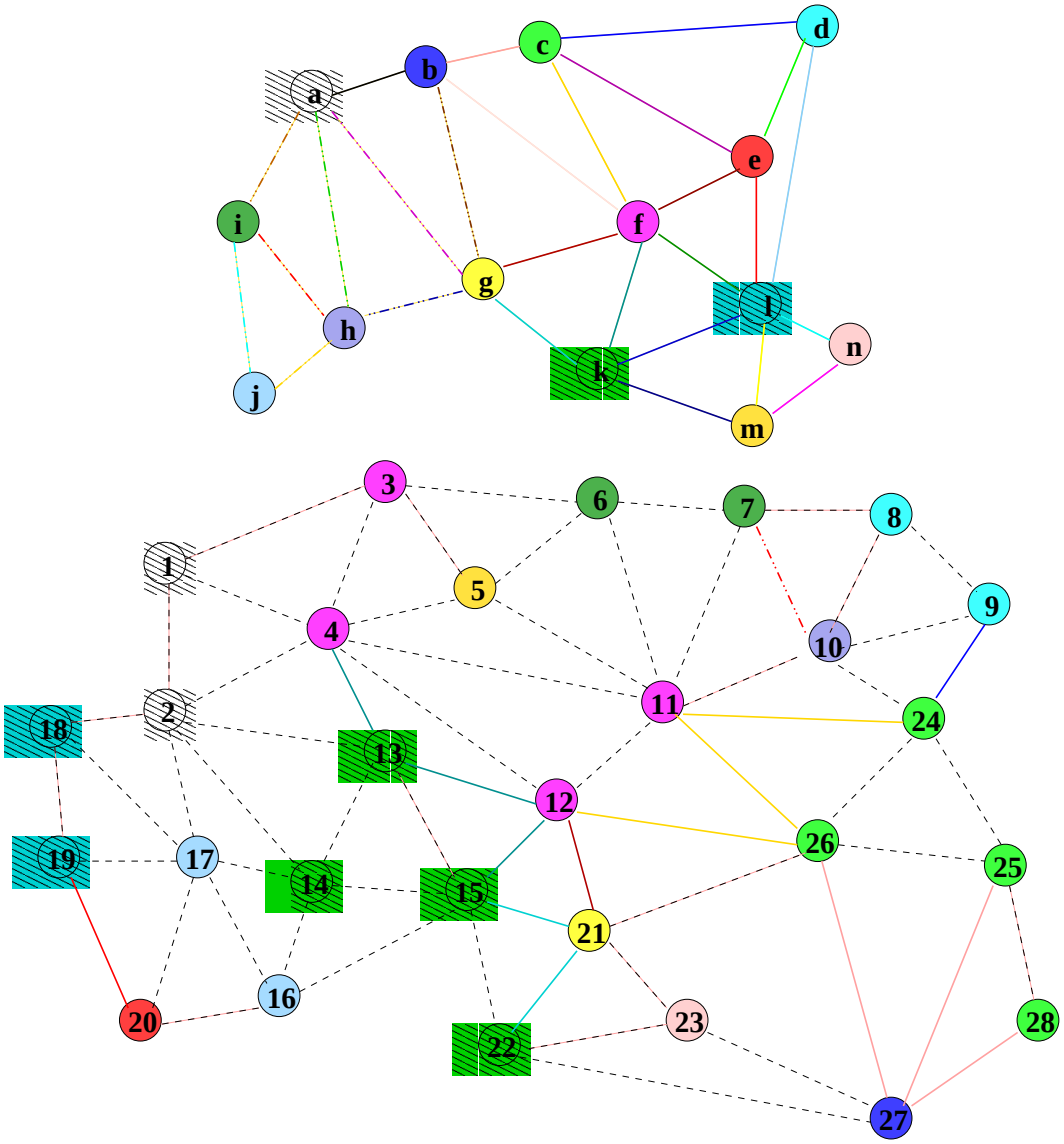


Figura 4.11: Problema teste I_7, caso AC_{100} com $\alpha = 0{,}5$ e $\alpha = 0{,}9$

Capítulo 5

Busca local

5.1 Introdução

Uma *instância* I de um problema de otimização combinatória P consiste de um conjunto S de soluções viáveis e uma função de custo f. Deseja-se encontrar uma solução globalmente ótima, ou seja, uma solução em S cujo custo f seja máximo (no caso do problema ser de maximização) ou mínimo (no caso do problema ser de minimização).

Algoritmos de busca local são amplamente usados na resolução de problemas de otimização combinatória. O uso de algoritmos desse tipo implica na definição de uma estrutura de *vizinhança*. De acordo com [1], uma *vizinhança* é um mapeamento $V : S \rightarrow 2^S$, que define para cada solução $s \in S$, um conjunto $V(s) \subseteq S$ de soluções próximas a s. O conjunto $V(s)$ é chamado de *vizinhança* da solução s, e cada $s' \in V(s)$ é um *vizinho* de s.

Um algoritmo de busca local consiste na melhora iterativa de uma solução inicial escolhida em S, através da busca de melhores soluções em sua vizinhança. Quando encontrada, a nova solução substitui a atual e a busca continua a partir dela. Caso contrário, a solução corrente (um ótimo local) é fornecida como resposta do algoritmo.

A qualidade de soluções obtidas a partir de estratégias de busca local está intimamente ligada à escolha de uma boa vizinhança. Na definição de boas vizinhanças são normalmente consideradas características específicas do problema tratado.

A descrição de um procedimento básico de busca local para a maximização de

uma função f, conforme apresentada em Resende e Ribeiro [54], é reproduzida a seguir:

Procedimento *Busca-local*
Entrada: uma solução $s \in S$;
1. enquanto s não for localmente ótima
2. achar $s' \in V(s)$ tal que $f(s') > f(s)$;
3. Substituir s por s';
4. fim-enquanto
Saída: a solução s;

5.2 Vizinhanças

São propostos dois algoritmos de busca local para o PCGAD, que utilizam duas vizinhanças distintas denotadas por V_a e V_b.

A vizinhança $V_a(s)$ é composta por soluções viáveis geradas a partir da modificação de A_j^{-1} para algum $j \in N_2$ pertencente à solução s, representada pelo grafo $G' = (N_1 \cup N_2, E')$ corrente. O conjunto C_j^s, para cada vértice $j \in N_2$ da solução corrente s, é formado pelos vértices de N_1 candidatos à substituição do vértice associado com j na solução corrente. Assim,

$$C_j^s = \{i \in N_1 \mid \overline{G}_j = (N_1 \cup N_2, E' - \{(A_j^{-1}, j)\} \cup \{(i,j)\}) \text{ é uma solução viável}\}.$$

Já a vizinhança $V_b(s)$, ela é formada por soluções viáveis obtidas pela troca dos valores de $A_{j'}^{-1}$ por $A_{j''}^{-1}$ em s, para dois vértices distintos j' e j'' de N_2.

Na Figura 5.1 são apresentadas três soluções s_1, s_2 e s_3 do PCGAD para o problema teste I_3. Os valores de $f_1(s_1)$, $f_1(s_2)$ e $f_1(s_3)$ são, respectivamente, 0,5877, 0,5906 e 0,6073.

A solução s_2 pertence à vizinhança $V_a(s_1)$, pois é obtida através da troca da associação ao vértice 2 de G_2: $a \in C_2^{s_1}$, $A_2^{-1} = d$ em s_1 e $A_2^{-1} = a$ em s_2. A solução s_3 pertence à vizinhança $V_b(s_2)$, pois foi obtida a partir de s_2 pela troca de A_4^{-1} por A_7^{-1}. Os conjuntos $C_j^s, j = 1, \ldots, 13$ definidos para as soluções s_1, s_2 e s_3 da Figura 5.1 são apresentados na Tabela 5.2.

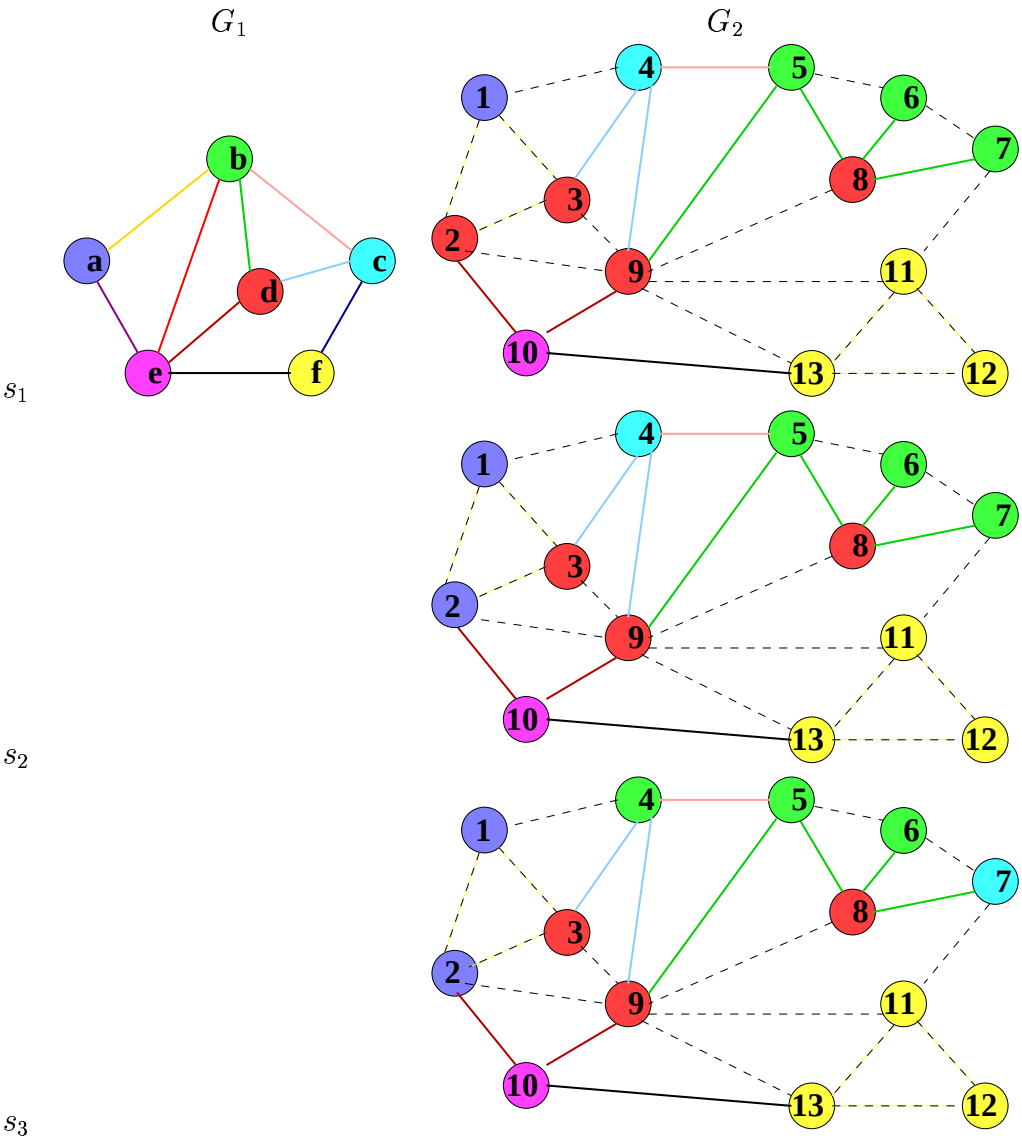


Figura 5.1: Soluções vizinhas

j	1	2	3	4	5	6	7	8	9	10	11	12	13
$C_j^{s_1}$	$\emptyset$	$\{a,e\}$	$\{a\}$	$\emptyset$	$\{c\}$	$\emptyset$	$\{f\}$	$\emptyset$	$\{e\}$	$\emptyset$	$\{b\}$	$\emptyset$	$\emptyset$
$C_j^{s_2}$	$\{d\}$	$\{d,e\}$	$\{a\}$	$\emptyset$	$\{c\}$	$\emptyset$	$\{f\}$	$\emptyset$	$\{e\}$	$\emptyset$	$\{b\}$	$\emptyset$	$\emptyset$
$C_j^{s_3}$	$\{d\}$	$\{d,e\}$	$\{a\}$	$\emptyset$	$\emptyset$	$\{c\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{c\}$	$\emptyset$	$\emptyset$

Tabela 5.1: Conjuntos C_j^s para as soluções s_1, s_2 e s_3 apresentadas na Figura 5.1

O número máximo de soluções vizinhas geradas a partir de uma solução s na vizinhança V_a é igual a $\sum_{j \in N_2} |C_j^s| \leq |N_2| \cdot |N_1|$. Já o número máximo de soluções geradas a partir de uma solução s na vizinhança V_b é igual a $|N_1| \cdot (|N_1| - 1)/2$, pois, neste caso, o número de pares de vértices de N_2 associados a vértices de N_1 distintos que podem ser trocados é no máximo igual ao número de combinações dois a dois dos vértices de N_1.

5.3 Busca local

A busca local tem como objetivo fornecer a melhor solução obtida a partir da solução corrente pela exploração das vizinhanças V_a e V_b. Os dados de entrada consistem dos grafos G_1 e G_2 e de uma solução inicial G', obtida pelo algoritmo construtivo apresentado no Capítulo 4. Para um dado grafo $G = (N, E)$, Γ_j representa o conjunto de vértices adjacentes em G ao vértice $j \in N$. Dois procedimentos de busca local (BL_a e BL_{a+b}) são propostos, de acordo com o mecanismo de exploração das vizinhanças. Em BL_a, a vizinhança V_a é explorada enquanto houver melhoria no valor de f_1. O algoritmo termina com a melhor solução encontrada. O procedimento BL_{a+b} pode ser dividido em duas etapas. A primeira etapa assemelha-se a BL_a. Na segunda etapa, quando não é mais possível melhorar f_1 considerando-se apenas a vizinhança V_a, a vizinhança V_b passa a ser explorada. A vizinhança V_a volta a ser explorada logo que a vizinhança V_b permita encontrar uma solução melhor do que o ótimo local corrente segundo V_a. Esse processo se repete até que o ótimo local relativo à vizinhança V_a seja também um ótimo local relativo à vizinhança V_b, quando o algoritmo termina.

O algoritmo de busca local proposto para o PCGAD que utiliza apenas a vizinhança V_a é apresentado a seguir:

Procedimento BL_a ( )

Entrada: $G_1 = (N_1, E_1)$, $G_2 = (N_2, E_2)$ e uma solução viável $G' = (N_1 \cup N_2, E')$;
1. $c_v := \sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|)$;
2. $c_a := \sum_{(i,i') \in E_1} \sum_{(j,j') \in E_2} (1 - |max\{x_{ij}x_{i'j'}, x_{ij}x_{j'i'}\} - s^a((i,i'),(j,j'))|)$;
3. $f_1 = \alpha \cdot c_v/(|N_1| \cdot |N_2|) + (1 - \alpha) \cdot c_a/(|E_1| \cdot |E_2|)$;
4. *ótimo* := .FALSE.;
5. construir os conjuntos C_j^s, $j \in N_2$;
6. enquanto (.NOT.*ótimo*)
7. *houve_melhora* := .FALSE.;

8. para-todo ($j \in N_2$ enquanto .NOT.*houve_melhora*)
9. $i := A_j^{-1}$;
10. para-todo ($l \in C_j^s$ enquanto .NOT.*houve_melhora*)
11. $\Delta_v := 2 \cdot s^v(l,j) - 2 \cdot s^v(i,j)$;
12. $\Delta_a := 0$;
13. para-todo ($j' \in \Gamma_j$)
14. para-todo ($i' \in \Gamma_i$)
15. se ($i' = A_{j'}^{-1}$) então
16. $\Delta_a := \Delta_a + 1 - (2s^a((i,i'),(j,j')))$;
17. fim-se
18. fim-para-todo
19. para-todo ($l' \in \Gamma_l$)
20. se ($l' = A_{j'}^{-1}$) então
21. $\Delta_a := \Delta_a - 1 + (2s^a((l,l'),(j,j')))$;
22. fim-se
23. fim-para-todo
24. fim-para-todo
25. $\Delta := \alpha \cdot \Delta_v/(|N_1| \cdot |N_2|) + (1-\alpha) \cdot \Delta_a/(|E_1| \cdot |E_2|)$;
26. se ($\Delta > 0$) então
27. *houve_melhora* := .TRUE.;
28. $E' := E' \cup \{(l,j)\} - \{(i,j)\}$;
29. $A_i := A_i - \{j\}$;
30. $A_l := A_l \cup \{j\}$;
31. $f_1 := f_1 + \Delta$;
32. Atualizar os conjuntos C_j^s;
33. fim-se
34. fim-para-todo
35. fim-para-todo
36. se (.NOT.*houve_melhora*) então
37. *ótimo* := .TRUE.;
38. fim-se
39. fim-enquanto

Saída: $G' = (N_1 \cup N_2, E')$;

Os valores das contribuições das associações entre vértices (c_v), entre arestas (c_a) e da função f_1 são calculados para a solução inicial G' (linhas 1-3). Além disso, são construídos os conjuntos de candidatos a troca de associação para cada vértice $j \in N_2$ (linha 5). A busca local baseada na vizinhança V_a continua enquanto um ótimo local não tiver sido obtido (laço das linhas 6-39). O laço das linhas 8-35 se repete até que alguma troca de associações permita melhorar o valor corrente de f_1. A troca do valor corrente de A_j^{-1} (linha 9) por um elemento de C_j^s (laço das linhas 10-34) para cada solução G' possível é avaliada através do cálculo de Δ_v e Δ_a, de acordo com o ajuste realizado no valor de contribuição dos vértices (linha 11) e de

arestas para f_1 (linhas 12-24). Se o valor Δ (linha 25) calculado a partir de Δ_v e Δ_a for positivo para algum $l \in C_j^s$ (linha 26), então a troca do valor corrente de A_j^{-1} por l é efetivamente realizada e todas as informações relativas à nova solução corrente são atualizadas (linhas 27 a 32).

O procedimento BL_{a+b} é apresentado a seguir. Neste procedimento, a busca local é efetuada sobre as vizinhanças V_a e V_b. Os elementos da vizinhança V_b são utilizados para diversificar as soluções geradas pela busca após a obtenção de um ótimo local segundo V_a. Neste procedimento, o bloco de linhas 1-36 é idêntico ao mesmo bloco em BL_a. A vizinhança V_b é explorada nas linhas 37-82. São examinadas todas as possíveis trocas de valores entre $A_{j'}^{-1}$ e $A_{j''}^{-1}$, para vértices distintos j' e j'' de N_2, que não violem os critérios de viabilidade. A primeira que causar uma melhoria no valor de f_1 é realizada, interrompendo-se o processo de exploração de V_b e retomando-se a exploração da vizinhança V_a.

Procedimento BL_{a+b} **( )**

Entrada: $G_1 = (N_1, E_1)$, $G_2 = (N_2, E_2)$ e uma solução viável $G' = (N_1 \cup N_2, E')$;
1. $c_v := \sum_{i \in N_1} \sum_{j \in N_2} (1 - |x_{ij} - s^v(i,j)|)$;
2. $c_a := \sum_{(i,i') \in E_1} \sum_{(j,j') \in E_2} (1 - |max\{x_{ij}x_{i'j'}, x_{ij}x_{j'i'}\} - s^a((i,i'),(j,j'))|)$;
3. $f_1 = \alpha \cdot c_v/(|N_1| \cdot |N_2|) + (1-\alpha) \cdot c_a/(|E_1| \cdot |E_2|)$;
4. *ótimo* := .FALSE.;
5. construir os conjuntos C_j^s, $j \in N_2$;
6. enquanto (.NOT.*ótimo*)
7. *houve_melhora* := .FALSE.;
8. para-todo ($j \in N_2$ enquanto .NOT.*houve_melhora*)
9. $i := A_j^{-1}$;
10. para-todo ($l \in C_j^s$ enquanto .NOT.*houve_melhora*)
11. $\Delta_v := 2 \cdot s^v(l,j) - 2 \cdot s^v(i,j)$;
12. $\Delta_a := 0$;
13. para-todo ($j' \in \Gamma_j$)
14. para-todo ($i' \in \Gamma_i$)
15. se ($i' = A_{j'}^{-1}$) então
16. $\Delta_a := \Delta_a + 1 - (2s^a((i,i'),(j,j')))$;
17. fim-se
18. fim-para-todo
19. para-todo ($l' \in \Gamma_l$)
20. se ($l' = A_{j'}^{-1}$) então
21. $\Delta_a := \Delta_a - 1 + (2s^a((l,l'),(j,j')))$;
22. fim-se
23. fim-para-todo
24. fim-para-todo
25. $\Delta := \alpha \cdot \Delta_v/(|N_1| \cdot |N_2|) + (1-\alpha) \cdot \Delta_a/(|E_1| \cdot |E_2|)$;

```
26.                se (Δ > 0) então
27.                   houve_melhora := .TRUE.;
28.                   E' := E' ∪ {(l, j)} − {(i, j)};
29.                   A_i := A_i − {j};
30.                   A_l := A_l ∪ {j};
31.                   f_1 := f_1 + Δ;
32.                   Atualizar os conjuntos C_j^s;
33.                fim-se
34.          fim-para-todo
35.    fim-para-todo
36.    se (.NOT.houve_melhora) então
37.       para-todo (j ∈ N_2 enquanto .NOT.houve_melhora)
38.             i := A_j^{-1};
39.             para-todo (j' ∈ N_2 satisfazendo j' > j enquanto .NOT.houve_melhora)
40.                   i' := A_{j'}^{-1};
41.                   se ((s^v(i, j') > 0) e (s^v(i', j) > 0) e
                          (o grafo induzido em G_2 por A_i − {j} ∪ {j'} for conexo) e
                          (o grafo induzido em G_2 por A_{i'} − {j'} ∪ {j}) for conexo) então
42.                      Δ_v := 2 · s^v(i, j') + 2 · s^v(i', j) − 2 · s^v(i, j) − 2 · s^v(i', j');
43.                      Δ_a := 0;
44.                      para-todo (j'' ∈ Γ_j)
45.                            para-todo (i'' ∈ Γ_i)
46.                                  se (i'' = A_{j''}^{-1}) então
47.                                     Δ_a := Δ_a + 1 − (2s^a((i, i''), (j, j'')));
48.                                  fim-se
49.                            fim-para-todo
50.                            para-todo (i'' ∈ Γ_{i'})
51.                                  se (i'' = A_{j''}^{-1}) então
52.                                     Δ_a := Δ_a − 1 + (2s^a((i', i''), (j, j'')));
53.                                  fim-se
54.                            fim-para-todo
55.                      fim-para-todo
56.                      para-todo (j'' ∈ Γ_{j'})
57.                            para-todo (i'' ∈ Γ_{i'})
58.                                  se (i'' = A_{j''}^{-1}) então
59.                                     Δ_a := Δ_a + 1 − (2s^a((i', i''), (j', j'')));
60.                                  fim-se
61.                            fim-para-todo
62.                            para-todo (i'' ∈ Γ_i)
63.                                  se (i'' = A_{j''}^{-1}) então
64.                                     Δ_a := Δ_a − 1 + (2s^a((i, i''), (j', j'')));
65.                                  fim-se
66.                            fim-para-todo
67.                      fim-para-todo
68.                      Δ := α · Δ_v/(|N_1| · |N_2|) + (1 − α) · Δ_a/(|E_1| · |E_2|);
69.                      se (Δ > 0) então
70.                         houve_melhora := .TRUE.;
71.                         E' := E' − {(i, j), (i', j')} ∪ {(i, j'), (i', j)};
```

72. $A_i := A_i - \{j\} \cup \{j'\}$;
73. $A_{i'} := A_{i_1} - \{j_1\} \cup \{j\}$;
74. $f_1 := f_1 + \Delta$;
75. Atualizar os conjuntos C_j^s;
76. fim-se
77. fim-para-todo
78. fim-para-todo
79. se (.NOT.*houve_melhora*) então
80. *ótimo* := .TRUE.;
81. fim-se
82. fim-se
83. fim-enquanto

Saída: $G' = (N_1 \cup N_2, E')$;

Para cada vértice $j \in N_2$, examinam-se os vértices $j' \in N_2$ para os quais os valores de A_j^{-1} e $A_{j'}^{-1}$ podem ser trocados (linhas 37-78). A nova solução gerada a partir de cada possível troca é submetida ao critério de viabilidade (linha 41). Se a viabilidade da solução é mantida, então verifica-se se a troca acarreta uma melhoria no valor de f_1. Isso é realizado através dos cálculos das diferenças (Δ_v e Δ_a) entre os valores das contribuições de vértices e de arestas da solução corrente e daquela gerada pela troca (linhas 42-67). A diferença Δ no valor de f_1 (linha 68) é calculada a partir de Δ_v e Δ_a, de forma semelhante àquela efetuada na exploração das soluções de V_a. Se Δ for positivo (linha 69), então a troca é efetuada e todas as informações relativas à nova solução corrente são atualizadas (linhas 70-75). A exploração de V_b termina quando a vizinhança é totalmente explorada ou quando se encontra a primeira solução que melhora o valor de f_1. No primeiro caso, como melhorias no valor de f_1 não são mais possíveis, o algoritmo termina (linhas 79-81). No segundo caso, a vizinhança V_a volta a ser explorada e o processo de busca de uma solução melhor se repete (laço das linhas 6-83).

5.4 Complexidade

No procedimento BL_a, a complexidade da construção de cada conjunto C_j^s é $O(|N_1| \cdot |N_2|)$, pois todos os vértices de N_1 são examinados para a determinação daqueles cuja troca de associação mantém a viabilidade da solução. Para cada um deles, a verificação do critério de viabilidade é $O(|N_2|)$. Como esses conjuntos são construídos para todos os vértices de N_2 (linha 5), então a complexidade destes cálculos

é $O(|N_1| \cdot |N_2|^2)$. A atualização de uma solução pertencente a V_a (linhas 27-32) tem complexidade $O(|N_1| \cdot |N_2|^2)$, pois é necessária a atualização dos conjuntos A_i e A_l e dos conjuntos C_j e $C_{j''}$, para o vértice j e seus adjacentes j'' em G_2. De acordo com o exemplo apresentado na Figura 5.1, observa-se que a transformação da solução s_1 na solução s_2 acarreta a atualização dos conjuntos $C_1^{s_1}$ e $C_2^{s_1}$ para os conjuntos $C_1^{s_2}$ e $C_2^{s_2}$ (ver Tabela 5.2). Os conjuntos C_j^s devem ser atualizados para os vértices j de N_2 cujos valores A_j^{-1} foram trocados e para os vértices j'' adjacentes em N_2, tais que $|A_{A_j^{-1}}| = 1$ antes da troca e $|A_{A_j^{-1}}| > 1$ após a troca ou vice-versa. Essa atualização pode ser feita, no pior caso, apenas no último dos $|N_1|$ vértices examinados no laço das linhas 10-34. Por isso, a complexidade deste laço é $O(|N_1| + |N_1| \cdot |N_2|^2) = O(|N_1| \cdot |N_2|^2)$. O laço das linhas 8-35 é executado enquanto não houver melhoria no valor de f_1 ou o conjunto N_2 não tiver sido totalmente examinado. Sua complexidade é $O(|N_2| + |N_1| \cdot |N_2|^2) = O(|N_1| \cdot |N_2|^2)$. A execução do laço das linhas 6-39 termina quando nenhuma melhoria em f_1 é possível (linhas 36-38).

No procedimento BL_{a+b}, a complexidade de uma iteração do laço das linhas 6-83, equivale à complexidade de uma iteração do procedimento BL_a somado à complexidade calculada para as linhas 36-82 do algoritmo, relativas à exploração da vizinhança V_b. Neste bloco de linhas, a verificação da viabilidade da solução decorrente da troca dos valores A_j^{-1} e $A_{j'}^{-1}$, para $j < j'$ em N_2 (linha 41), tem complexidade $O(|N_2|)$. Se a troca é possível, então a complexidade de atualização da solução das informações relativas a ela (linhas 70-75) é $O(|N_1| \cdot |N_2|)^2$. Como, no pior caso, todo o conjunto N_2 é examinado sem gerar uma melhoria em f_1, então a complexidade de execução do laço das linhas 37-80 é de $O(|N_2| + |N_1| \cdot |N_2|^2) = O(|N_1| \cdot |N_2|^2)$. Assim, a complexidade de uma iteração do laço das linhas 6-83 é $O(|N_1| \cdot |N_2|^2)$.

5.5 Resultados computacionais

Os algoritmos apresentados neste capítulo foram implementados na linguagem C. O compilador utilizado foi o *gcc*, versão 2.96. Uma solução do PCGAD é representada como um vetor de inteiros de N_2 posições. Os conjuntos $A_i, i \in N_1$, foram implementados como vetores de N_1 ponteiros para listas encadeadas.

Um vetor de $|N_2|$ ponteiros para listas encadeadas é utilizado para representar

os conjuntos C_j^s, para cada $j \in N_2$. A cada nova solução gerada, são reconstruídos os conjuntos referentes aos vértices j de N_2 que sofreram uma troca no valor de A_j^{-1} e a seus vértices adjacentes.

Os testes experimentais foram realizados com os mesmos 12 problemas teste apresentados no Capítulo 4, em um microcomputador Pentium II 450 MHz, sob o sistema operacional Linux Red Hat, versão 7.1.

Os resultados obtidos pelos algoritmos de busca local apresentados neste capítulo são mostrados nas Tabelas 5.2 e 5.3. Nestas tabelas, são reapresentados os resultados do algoritmo construtivo considerando-se a geração de no máximo uma solução viável (AC_1) e a geração de no máximo 100 soluções viáveis (AC_{100}). As soluções referentes aos resultados de AC_1 e de AC_{100} são utilizadas como iniciais para BL_a e BL_{a+b}. Os testes foram realizados para valores de α iguais a 0,1, 0,5 e 0,9. Cada iteração dos procedimentos de busca local equivale respectivamente a uma iteração do laço das linhas 6-39 (BL_a) e a uma iteração do laço das linhas 6-83 (BL_{a+b}). O número de iterações necessárias para a obtenção da solução localmente ótima obtida para cada problema teste é indicado na coluna Iteração e o tempo médio de processamento em segundos por iteração do algoritmo é apresentado na coluna TM.

Considerando-se as soluções iniciais AC_1 (Tabela 5.2) e AC_{100} (Tabela 5.3) para ambos procedimentos BL_a e BL_{a+b} e para todos os valores de α testados, observa-se que a busca local melhorou as soluções obtidas pelo algoritmo construtivo em 11 (I_1 a I_9, I_{5a}, I_{8a}) dos 12 problemas teste. A mesma solução do algoritmo construtivo foi obtida para I_0, que coincide com a ótima para esse problema teste. Esses resultados justificam o uso dos algoritmos de busca local para obtenção de melhores soluções para o PCGAD.

Na primeira situação (AC_1, Tabela 5.2), o procedimento BL_{a+b} melhorou a solução obtida por BL_a em sete (I_5 a I_9, I_{5a}, I_{8a}) dos 12 problemas teste, para todos os valores de α testados. Na segunda situação (AC_{100}, Tabela 5.3), essa melhora ocorreu para a mesma quantidade de problemas teste (I_5 a I_9, I_{5a}, I_{8a}) para $\alpha =$ 0,1 e $\alpha =$ 0,5, e para I_3, I_5 a I_9, I_{5a}, I_{8a} para $\alpha =$ 0,9, ou seja, apenas para um problema teste a mais que no caso AC_1.

Na Tabela 5.4 são exibidos os resultados da comparação dos valores obtidos por BL_{a+b}, partindo de soluções iniciais relativas a AC_1 e AC_{100}. Valores melhores da função f_1 foram obtidos por BL_{a+b} quando a busca local partiu de melhores

Problema teste	f_1^*	AC_1			BL_a			BL_{a+b}		
		tentativa	f_1	TM	Iteração	f_1	TM	Iteração	f_1	TM
I_0 ($\alpha = 0{,}1$)	0,3240	1	0,3240	-	1	0,3240	-	1	0,3240	-
I_1	0,2707	1	0,2574	-	4	0,2707	-	4	0,2707	-
I_2	0,2818	1	0,2646	-	6	0,2789	-	6	0,2789	-
I_3	0,5603	1	0,4921	-	5	0,5039	-	5	0,5039	-
I_4	n.d.	3	0,5166	-	9	0,5477	0,001	9	0,5477	-
I_5	n.d.	297	0,4908	-	15	0,5044	0,001	17	0,5056	0,001
I_{5a}	n.d.	9	0,5106	0,001	14	0,5195	0,001	29	0,5242	0,001
I_6	n.d.	1	0,3922	-	280	0,3953	0,020	289	0,3954	0,021
I_7	n.d.	5	0,1226	0,002	10	0,1276	0,001	12	0,1298	0,001
I_8	n.d.	26	0,5020	0,002	113	0,5046	0,018	132	0,5050	0,021
I_{8a}	n.d.	26	0,5339	0,002	112	0,5363	0,019	137	0,5368	0,020
I_9	n.d.	1	0,5001	0,005	488	0,5018	0,148	570	0,5020	0,171
I_0 ($\alpha = 0{,}5$)	0,6142	1	0,6142	-	1	0,6142	-	1	0,6142	-
I_1	0,5202	1	0,5055	-	4	0,5202	-	4	0,5202	-
I_2	0,5098	1	0,4986	-	5	0,5098	-	5	0,5098	-
I_3	0,6375	1	0,5878	-	7	0,6104	-	7	0,6104	-
I_4	0,6754	3	0,6200	-	9	0,6492	-	9	0,6492	-
I_5	n.d.	297	0,5038	-	17	0,5212	0,001	18	0,5239	0,001
I_{5a}	n.d.	9	0,5044	-	22	0,5189	0,001	24	0,5223	0,001
I_6	n.d.	1	0,4022	-	257	0,4094	0,018	280	0,4097	0,020
I_7	n.d.	5	0,3704	-	8	0,3771	-	17	0,3814	0,001
I_8	n.d.	26	0,4999	0,002	117	0,5092	0,019	148	0,5107	0,021
I_{8a}	n.d.	26	0,5176	0,002	113	0,5266	0,020	164	0,5289	0,023
I_9	n.d.	1	0,5025	0,005	380	0,5082	0,156	551	0,5099	0,185
I_0 ($\alpha = 0{,}9$)	0,9045	1	0,9045	-	1	0,9045	-	1	0,9045	-
I_1	0,7698	1	0,7535	-	4	0,7698	-	4	0,7698	-
I_2	0,7465	1	0,7327	-	6	0,7462	-	6	0,7462	-
I_3	0,7147	1	0,6835	-	5	0,6937	-	5	0,6937	-
I_4	0,7761	3	0,7235	-	10	0,7577	-	10	0,7577	-
I_5	n.d.	297	0,5168	-	16	0,5402	0,001	19	0,5474	0,001
I_{5a}	n.d.	9	0,4981	0,001	13	0,5173	0,001	25	0,5329	0,002
I_6	n.d.	1	0,4122	-	302	0,4246	0,015	337	0,4251	0,017
I_7	n.d.	5	0,6182	-	7	0,6287	-	20	0,6338	0,001
I_8	n.d.	26	0,4978	0,002	111	0,5144	0,018	145	0,5183	0,021
I_{8a}	n.d.	26	0,5014	0,002	111	0,5180	0,018	145	0,5218	0,022
I_9	n.d.	1	0,5049	0,010	396	0,5159	0,155	554	0,5184	0,178

n.d.: não disponível

- : tempo de processamento inferior a 10^{-3} segundo

Tabela 5.2: Resultados obtidos pelos algoritmos de busca local, com a solução inicial gerada por AC_1.

Problema teste		AC_{100}				BL_a			BL_{a+b}		
	f_1^*	Tent.	NSV	f_1	TM	Iteração	f_1	TM	Iteração	f_1	TM
I_0 ($\alpha = 0{,}1$)	0,3240	1	1	0,3240	-	1	0,3240	-	1	0,3240	-
I_1	0,2707	99	72	0,2696	-	2	0,2707	-	2	0,2707	-
I_2	0,2818	117	78	0,2780	-	2	0,2818	-	2	0,2818	0,005
I_3	0,5603	7	7	0,5213	-	3	0,5371	-	3	0,5371	-
I_4	n.d.	206	71	0,5519	-	4	0,5621	-	4	0,5621	-
I_5	n.d.	451	2	0,4941	-	11	0,5023	0,002	23	0,5066	0,002
I_{5a}	n.d.	237	24	0,5215	-	5	0,5238	0,002	6	0,5242	0,002
I_6	n.d.	11	11	0,3927	0,001	235	0,3952	0,019	269	0,3954	0,021
I_7	n.d.	198	46	0,1269	-	5	0,1297	-	7	0,1308	0,001
I_8	n.d.	436	37	0,5024	0,002	69	0,5047	0,013	115	0,5055	0,017
I_{8a}	n.d.	292	22	0,5343	0,002	86	0,5362	0,013	113	0,5369	0,015
I_9	n.d.	86	33	0,5002	0,010	430	0,5017	0,146	577	0,5021	0,210
I_0 (α =0,5)	0,6142	1	1	0,6142	-	1	0,6142	-	1	0,6142	-
I_1	0,5202	99	72	0,5135	-	2	0,5202	-	2	0,5202	-
I_2	0,5098	50	36	0,5090	-	2	0,5098	-	2	0,5098	-
I_3	0,6375	7	7	0,6020	-	3	0,6131	-	3	0,6131	-
I_4	0,6754	206	71	0,6559	-	5	0,6754	-	5	0,6754	-
I_5	n.d.	297	1	0,5038	-	17	0,5212	0,001	18	0,5239	0,001
I_{5a}	n.d.	417	47	0,5223	-	6	0,5284	0,002	8	0,5299	0,002
I_6	n.d.	40	38	0,4045	0,001	204	0,4090	0,016	241	0,4099	0,017
I_7	n.d.	34	11	0,3757	-	7	0,3771	-	14	0,3806	0,001
I_8	n.d.	292	22	0,5023	0,002	84	0,5086	0,012	119	0,5108	0,015
I_{8a}	n.d.	292	22	0,5200	0,002	83	0,5266	0,012	114	0,5285	0,014
I_9	n.d.	207	96	0,5031	0,010	399	0,5087	0,123	544	0,5099	0,137
I_0 ($\alpha = 0{,}9$)	0,9045	1	1	0,9045	-	1	0,9045	-	1	0,9045	-
I_1	0,7698	41	32	0,7672	-	2	0,7698	-	2	0,7698	-
I_2	0,7465	50	36	0,7427	-	3	0,7465	-	3	0,7465	-
I_3	0,7147	67	49	0,6966	-	2	0,6966	-	8	0,7147	-
I_4	0,7761	43	13	0,7655	-	6	0,7714	-	6	0,7714	-
I_5	n.d.	297	1	0,5168	-	16	0,5402	0,001	19	0,5474	0,002
I_{5a}	n.d.	417	47	0,5243	-	6	0,5351	0,002	13	0,5434	0,002
I_6	n.d.	40	38	0,4168	0,001	266	0,4226	0,017	320	0,4248	0,020
I_7	n.d.	34	11	0,6282	-	3	0,6283	-	12	0,6319	0,001
I_8	n.d.	292	22	0,5022	0,002	93	0,5164	0,012	118	0,5186	0,014
I_{8a}	n.d.	292	22	0,5058	0,002	92	0,5198	0,012	120	0,5222	0,014
I_9	n.d.	207	96	0,5060	0,010	364	0,5160	0,114	511	0,5187	0,134

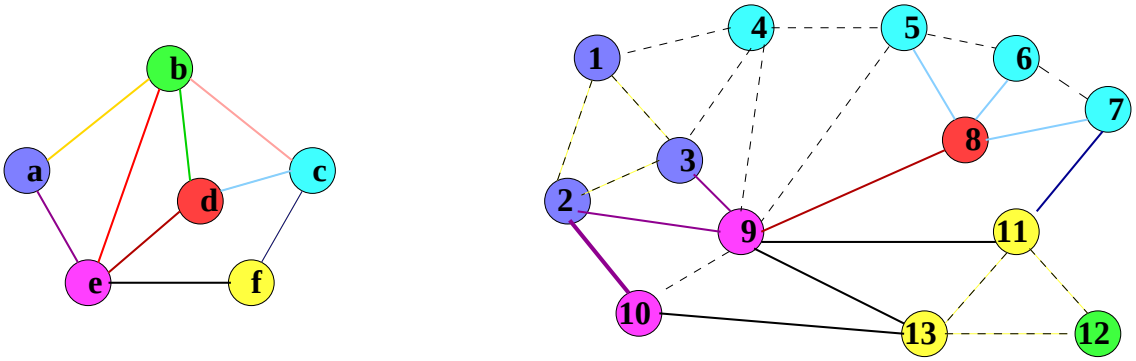
n.d.: não disponível

- : tempo de processamento inferior a 10^{-3} segundo

Tabela 5.3: Resultados obtidos pelos algoritmos de busca local, com a solução inicial gerada por AC_{100}.

soluções iniciais (AC_{100}) em oito problemas teste (I_2 a I_5, I_7 a I_9, I_{8a}) para $\alpha =$ 0,1, em cinco problemas teste (I_3, I_4, I_6, I_8, I_{5a}) para $\alpha = 0{,}5$ e em sete problemas teste (I_2 a I_4, I_8, I_9, I_{5a}, I_{8a}) para $\alpha = 0{,}9$. Em um número pequeno de problemas teste (I_7, I_{8a} para $\alpha = 0{,}5$ e I_6, I_7 para $\alpha = 0{,}9$), o valor de f_1 relativo a BL_{a+b} considerando-se soluções iniciais obtidas através de AC_1 foi melhor. Assim, constata-se que melhores resultados foram alcançados quando o algoritmo de busca local partiu de uma solução melhor. Os algoritmos de busca local são sensíveis ao valor do parâmetro α.

A Figura 5.2 mostra a solução obtida pelos algoritmos BL_a e BL_{a+b} para o problema teste I_3 ($\alpha = 0{,}5$). A mesma solução foi obtida pelos dois procedimentos, com valor de f_1 melhor do que aquele obtido pelo algoritmo construtivo (Figura 4.6). Comparando-se as duas soluções, observa-se o aumento no número de associações entre arestas. Apesar dessa solução não corresponder à ótima para esse problema teste, o maior número de arestas de G_1 associadas com arestas de G_2 indica que uma maior parte da estrutura de G_1 foi mantida em G_2. O valor de f_1 para essa solução é 0,6131, enquanto o valor ótimo é 0,6375.



G_1 G_2

Figura 5.2: Representação da mesma solução obtida pelos algoritmos de busca local para o problema teste I_3.

A Figura 5.3 mostra a melhor solução obtida para o problema teste I_7 e $\alpha = 0{,}1$ dentre aquelas obtidas pelas diferentes combinações do número de soluções iniciais construídas (AC_1 e AC_{100}) e dos procedimentos de busca local utilizados (BL_a e BL_{a+b}), correspondendo ao caso $AC_{100} + BL_{a+b}$. Comparando-se as Figuras 5.3 e 4.5 , observa-se que 13 vértices de N_2 (6, 7, 11, 13, 14, 15, 16, 17, 20, 23, 25, 26 e 27) foram corretamente associados. A solução encontrada pela busca local é melhor que aquela obtida pelo algoritmo construtivo tanto no valor de f_1 (para essa solução é igual a 0,1308, enquanto que para aquela obtida pelo algoritmo construtivo é igual

Problema teste	BL_{a+b} (AC_{100}/AC_1)
I_0 $(\alpha = 0{,}1)$	0,000
I_1	0,000
I_2	1,040
I_3	6,589
I_4	2,629
I_5	0,198
I_{5a}	0,000
I_6	0,000
I_7	0,770
I_8	0,099
I_{8a}	0,019
I_9	0,020
Média	0,947
I_0 $(\alpha = 0{,}5)$	0,000
I_1	0,000
I_2	0,000
I_3	0,442
I_4	4,036
I_5	0,000
I_{5a}	1,455
I_6	0,049
I_7	-0,210
I_8	0,020
I_{8a}	-0,076
I_9	0,000
Média	0,476
I_0 $(\alpha = 0{,}9)$	0,000
I_1	0,000
I_2	0,040
I_3	3,027
I_4	1,808
I_5	0,000
I_{5a}	1,970
I_6	-0,071
I_7	-0,300
I_8	0,058
I_{8a}	0,077
I_9	0,058
Média	0,556

Tabela 5.4: Percentuais relativos de aumento no valor de f_1

a 0,1269), quanto no número de vértices e de arestas corretamente associados em relação à solução mais adequada.

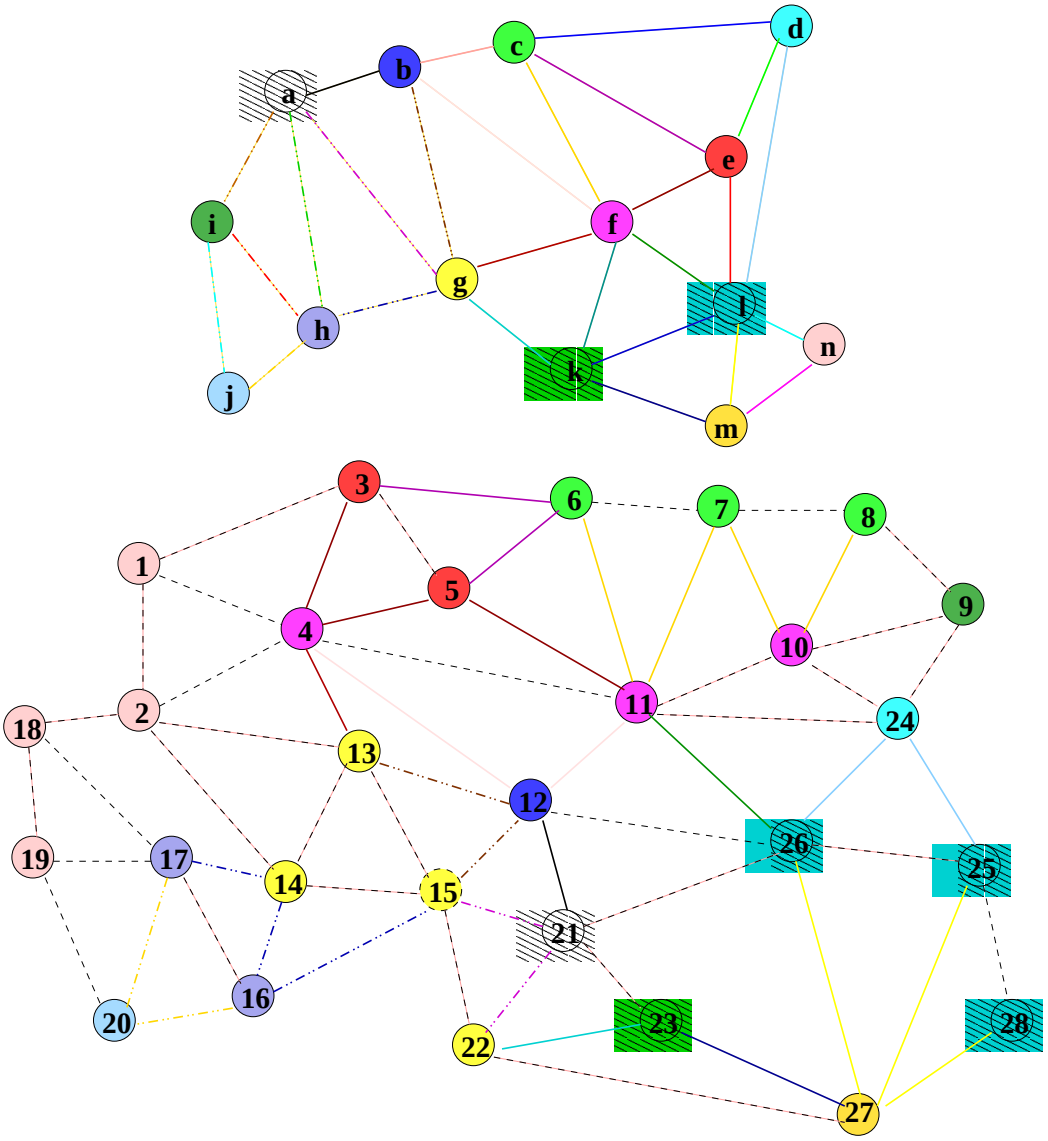


Figura 5.3: Problema teste I_7: $AC_{100} + BL_{a+b}$ com $\alpha = 0{,}1$

A solução obtida pelo algoritmo BL_{a+b} corresponde à ótima para todos os problemas teste para as quais essa solução é conhecida, exceto para I_3 com $\alpha = 0{,}1$ e $\alpha = 0{,}5$ e para I_4 com $\alpha = 0{,}9$. Neste caso, os valores de f_1 para as soluções obtidas estão respectivamente 4,1%, 3,8% e 6,1% abaixo do valor ótimo.

Os gráficos apresentados nas Figuras 5.4 e 5.5 mostram, para todos os problemas teste e todos os valores de α testados, os valores de f_1 calculados para as soluções encontradas pelos procedimentos BL_a e BL_{a+b} e seus tempos totais de processamento. Esses testes foram realizados para $\alpha = 0{,}1$, $\alpha = 0{,}5$ e $\alpha = 0{,}9$

considerando-se: (a) soluções do procedimento BL_a com soluções iniciais relativas a AC_1 ($AC_1 + BL_a$); (b) soluções do procedimento BL_a com soluções iniciais relativas a AC_{100} ($AC_{100} + BL_a$); (c) soluções do procedimento BL_{a+b} com soluções iniciais relativas a AC_1 ($AC_1 + BL_{a+b}$); e (d) soluções do procedimento BL_{a+b} com soluções iniciais relativas a AC_{100} ($AC_{100} + BL_{a+b}$).

Apesar de muito próximos, observa-se que os melhores resultados foram aqueles obtidos pelo procedimento BL_{a+b} com soluções iniciais de melhor qualidade (AC_{100}). Essa afirmação pode ser verificada na Tabela 5.5, onde são apresentados os valores de f_1 obtidos pela busca local nos casos (a), (b), (c) e (d) para $\alpha = 0{,}1$.

Alternativamente, considerando-se os 36 problemas teste (12 intâncias e três valores de α), pode-se observar pelos resultados das Tabelas 5.2 e 5.3 que a melhor solução é encontrada em 7 ocasiões pela combinação do caso (a), em 14 ocasiões no caso (b), em 16 ocasiões no caso (c) e em 32 ocasiões no caso (d), confirmando-se a superioridade da estratégia $AC_{100} + BL_{a+b}$. Observa-se pelos gráficos que o comportamento da busca local é semelhante nos quatro casos avaliados, mesmo com a variação de valores do parâmetro α. Com exceção da instância I_9, os tempos totais de processamento dos procedimentos testados não diferem muito. Isso indica que o procedimento BL_{a+b}, que efetua uma busca mais criteriosa do espaço de soluções, pode ser utilizado sem onerar o tempo total de processamento.

Problema teste	$AC_1 + BL_a$	$AC_{100} + BL_a$	$AC_1 + BL_{a+b}$	$AC_{100} + BL_{a+b}$
I_0	0,3240	0,3240	0,3240	0,3240
I_1	0,2707	0,2707	0,2707	0,2707
I_2	0,2789	0,2818	0,2789	0,2818
I_3	0,5039	0,5371	0,5039	0,5371
I_4	0,5477	0,5621	0,5477	0,5621
I_5	0,5044	0,5023	0,5056	0,5066
I_{5a}	0,5195	0,5238	0,5242	0,5242
I_6	0,3953	0,3952	0,3954	0,3954
I_7	0,1276	0,1297	0,1298	0,1308
I_8	0,5046	0,5047	0,5050	0,5055
I_{8a}	0,5363	0,5362	0,5368	0,5369
I_9	0,5018	0,5017	0,5020	0,5021

Tabela 5.5: Valores de f_1 obtidos pela busca local com $\alpha = 0{,}1$ para $AC_1 + BL_a$, $AC_{100} + BL_a$, $AC_1 + BL_{a+b}$ e $AC_{100} + BL_{a+b}$.

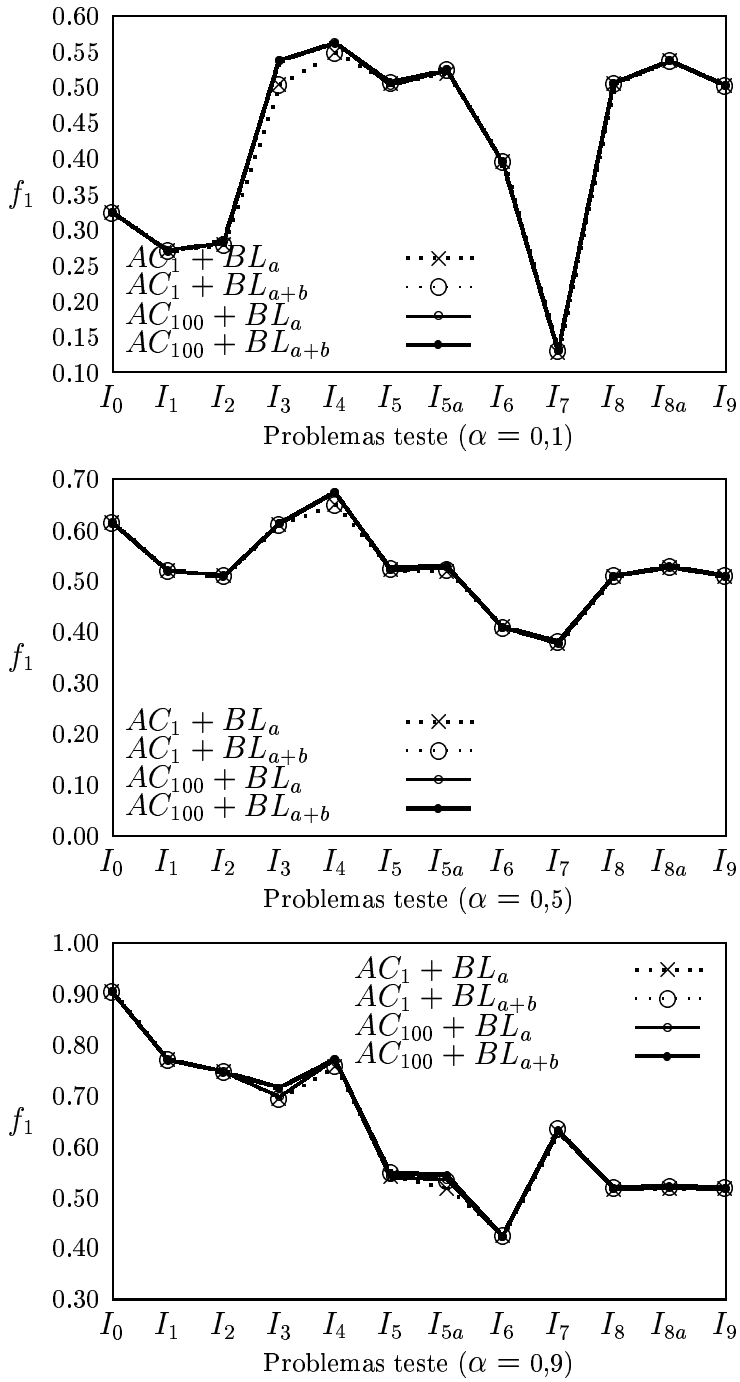


Figura 5.4: Valores de f_1 relativos às soluções obtidas por BL_a e BL_{a+b} com diferentes soluções iniciais

5.6 Conclusões

Através da análise dos resultados obtidos, pode-se perceber a utilidade dos algoritmos de busca local explorando as vizinhanças V_a e V_b para se atingir melhores soluções para o PCGAD. Em todos os testes realizados, os resultados alcançados

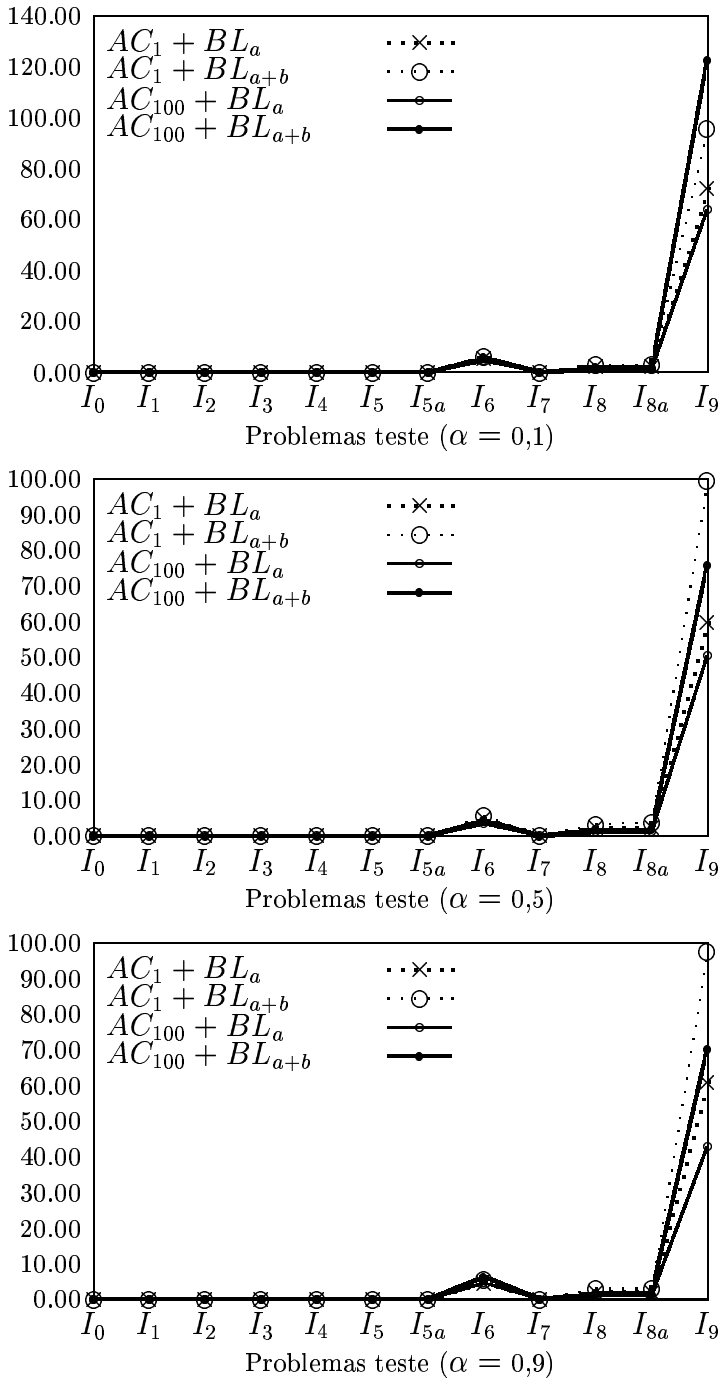


Figura 5.5: Tempo total de processamento dos procedimentos BL_a e BL_{a+b} com diferentes soluções iniciais

por BL_{a+b} são iguais ou melhores do que aqueles obtidos por BL_a. Estes resultados são ainda melhorados se a busca começa de melhores soluções (AC_{100}). Além disso, os tempos médios por iteração e os tempos totais dos dois procedimentos (BL_a e BL_{a+b}) são muito próximos.

A busca de soluções pelos algoritmos apresentados neste capítulo é guiada pela

preservação de viabilidade e pela melhoria do valor de f_1. Os algoritmos de busca local foram capazes de melhorar as soluções iniciais construídas para todos os problemas teste, exceto para I_0, cuja solução obtida pelo algoritmo construtivo já é a ótima. Esses procedimentos, executados a partir de soluções iniciais relativas a AC_{100}, alcançaram o valor ótimo em todas as instâncias para as quais se conhece uma solução ótima, exceto para a instância I_3 para $\alpha = 0{,}1$ e $\alpha = 0{,}5$. Também permitiram encontrar uma solução melhor para o problema teste I_7 do que aquelas obtidas pelo algoritmo construtivo, em termos do número de regiões (nós do grafo imagem) reconhecidas corretamente na imagem supersegmentada.

No próximo capítulo será apresentado um algoritmo do tipo GRASP para o PC-GAD, que utiliza nas fases de construção e de busca local os algoritmos construtivo e de busca local propostos respectivamente nos Capítulos 4 e 5.

Capítulo 6

Uma heurística GRASP

6.1 Introdução

Neste capítulo será apresentada uma implementação da metaheurística GRASP (*Greedy Randomized Adaptative Search Procedure*) para o PCGAD. GRASP [20, 21, 22, 54] é uma metaheurística proposta para problemas combinatórios, na qual cada iteração consiste basicamente de duas fases: construção de uma solução viável para o problema e sua utilização como solução inicial para um procedimento de busca local. Nesta última fase, vizinhanças da solução construída são sucessivamente investigadas até que um ótimo local seja encontrado. Novas soluções construídas a cada iteração do GRASP permitem a exploração do espaço de soluções pela busca local a partir de múltiplos pontos iniciais. A melhor dentre todas as soluções visitadas é fornecida como resultado do algoritmo.

Na etapa de construção, uma solução é gradativamente montada por elementos escolhidos dentre os candidatos que não violam critérios de viabilidade. Essa escolha depende da avaliação desses candidatos por uma função gulosa e os melhores são colocados numa lista chamada *Lista Restrita de Candidatos* (LRC). A seleção dos melhores candidatos baseada no uso de uma função de avaliação representa o aspecto guloso do algoritmo. O elemento a ser incluído na solução é aleatoriamente escolhido na LRC (aspecto probabilístico do algoritmo). A cada nova inclusão a LRC é atualizada (aspecto adaptativo do algoritmo). Em seguida, a solução gerada na fase de construção de cada iteração do GRASP (que não é necessariamente ótima) é melhorada através de um procedimento de busca local. São definidos como parâmetros do procedimento: o número máximo $MaxIter$ de iterações e uma semente inicial $Seed$ para o gerador de números aleatórios, usado para a escolha aleatória dos ele-

mentos que irão compor a solução construída. O algoritmo termina com a melhor das soluções obtidas após $MaxIter$ iterações.

O procedimento *GRASP* proposto para a maximização da função objetivo do PCGAD é apresentado a seguir, conforme descrito em [54].

Procedimento *GCG* *(MaxIter, Seed)*
Entrada: $G_1 = (N_1, E_1)$ e $G_2 = (N_2, E_2)$;
1. $iter := 1$;
2. para-todo $iter < MaxIter$
3. $Solução :=$ AlgoritmoConstrutivo($Seed$);
4. $Solução :=$ BuscaLocal($Solução$);
5. $iter := iter + 1$;
6. AtualizaSolução ($Solução$, $MelhorSolução$);
7. fim-para-todo
Saída: $MelhorSolução$;

Nas etapas de construção (linha 3 acima) e de buscal local (linha 4 acima) do algoritmo *GCG* são utilizados respectivamente os algoritmos construtivo e de busca local anteriormente apresentados nos Capítulos 4 e 5. O algoritmo *GCG* não possui a característica gulosa, pois o algoritmo construtivo não considera o valor da função de avaliação para a escolha de cada novo elemento a ser inserido na solução em construção, mas sim apenas a preservação da viabilidade da solução. Desta forma, não há a necessidade de definição da LRC.

6.2 Resultados computacionais

O algoritmo *GCG* foi implementado na linguagem C. O compilador utilizado foi o *gcc*, versão 2.96. Os testes experimentais foram realizados com os mesmos 12 problemas teste (apresentadas no Capítulo 4) em um microcomputador Pentium II 450 MHz, sob o sistema operacional Linux Red Hat, versão 7.1.

Os resultados dos experimentos realizados com o algoritmo *GCG* são apresentados na Tabela 6.1. Foram realizados testes para valores do parâmetro α iguais a 0,1, 0,5 e 0,9 e para $MaxIter = 200$. Inicialmente são reapresentados nesta tabela os

valores de f_1 relativos às soluções encontradas pelos algoritmos construtivo (AC_{100}) e de busca local (BL_a e BL_{a+b}). De acordo com os resultados apresentados nos capítulos anteriores, o algoritmo GCG foi executado com os seguintes parâmetros: o algoritmo construtivo fornece como solução inicial a melhor das 100 primeiras soluções viáveis construídas (obtidas em no máximo 500 tentativas) e utiliza-se o algoritmo de buscal local BL_{a+b}. A coluna "Iteração" exibe a iteração na qual foi obtida a melhor solução e na coluna TM é indicado o tempo médio de processamento em segundos de cada iteração GRASP.

O algoritmo GCG melhorou os resultados obtidos por BL_{a+b} em nove ($\alpha = 0{,}1$) e oito ($\alpha = 0{,}5$ e $\alpha = 0{,}9$) dos 12 problemas teste. Os problemas teste para as quais não houve melhora pelo algoritmo GCG são aqueles que a solução ótima já havia sido obtida por BL_{a+b}. A Tabela 6.2 apresenta a média dos percentuais de aumento em f_1 das soluções obtidas pelo algoritmo GCG em relação à BL_{a+b}. De acordo com esta tabela, observa-se que quanto maior é o valor de α, menor é a melhoria alcançada pelo algoritmo GCG em relação ao algoritmo BL_{a+b}. Estes resultados mostram que o algoritmo GCG parece ser mais eficaz em relação à busca local para valores menores de α, quando o peso relativo das associações entre arestas é maior. O algoritmo GCG atingiu o ótimo global de todos os problemas teste para os quais esse valor é conhecido (Tabela 6.1).

Na Figura 6.1 é apresentada a solução obtida pelo algoritmo GCG para o problema teste I_3, considerando-se $\alpha = 0{,}5$. Observa-se, ao contrário das soluções obtidas pelos outros algoritmos, que a estrutura do grafo G_1 é totalmente preservada no grafo G_2, isto é, todas as arestas de G_1 estão associadas com arestas de G_2. Essa solução corresponde à ótima para esse problema teste e seu valor de f_1 é 0,6375. O valor de f_1 para a solução obtida por BL_{a+b} para esse problema teste é 0,6131. Desta forma, o algoritmo GCG conseguiu melhorar em 4% a solução alcançada por BL_{a+b}.

A Figura 6.2 mostra a melhor solução obtida para o problema teste I_7 no caso $\alpha = 0{,}5$, considerando-se soluções iniciais relativas a AC_{100} e à busca local BL_{a+b}. Comparando-se essa solução com a mais adequada (Figura 4.5), observa-se que apenas quatro vértices (3, 5, 20 e 21) não foram corretamente associados, ou seja, 86% das regiões que compõem a imagem representada por G_2 foram reconhecidas. O valor de f_1 para essa solução é 0,3855, que é 1,3% maior que o valor 0,3806 relativo à solução obtida por $AC_{100}+BL_{a+b}$ apresentada no Capítulo 5. A solução encontrada

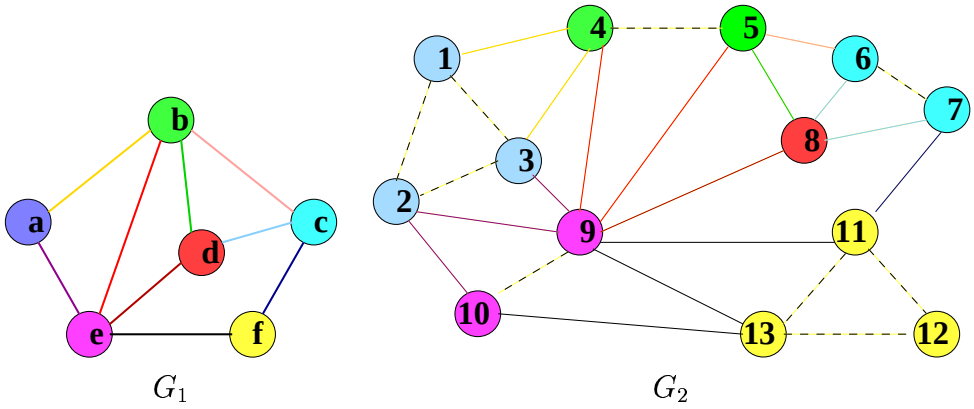
Problema teste	f_1^*	AC_{100} f_1	BL_a f_1	BL_{a+b} f_1	GCG Iteração	GCG f_1	GCG TM
I_0 ($\alpha = 0{,}1$)	0,3240	0,3240	0,3240	0,3240	1	0,3240	0,003
I_1	0,2707	0,2696	0,2707	0,2707	2	0,2707	0,005
I_2	0,2818	0,2780	0,2818	0,2818	4	0,2818	0,010
I_3	0,5603	0,5213	0,5371	0,5371	2	0,5603	0,011
I_4	n.d.	0,5519	0,5621	0,5621	4	0,5766	0,026
I_5	n.d.	0,4941	0,5023	0,5066	83	0,5086	0,140
I_{5a}	n.d.	0,5215	0,5238	0,5242	38	0,5337	0,133
I_6	n.d.	0,3927	0,3952	0,3954	23	0,3955	5,918
I_7	n.d.	0,1269	0,1297	0,1308	75	0,1395	0,108
I_8	n.d.	0,5024	0,5047	0,5055	189	0,5071	3,825
I_{8a}	n.d.	0,5343	0,5362	0,5369	38	0,5388	3,842
I_9	n.d.	0,5002	0,5017	0,5021	115	0,5024	109,493
I_0 ($\alpha = 0{,}5$)	0,6142	0,6142	0,6142	0,6142	1	0,6142	0,003
I_1	0,5202	0,5135	0,5202	0,5202	2	0,5202	0,004
I_2	0,5098	0,5090	0,5098	0,5098	0	0,5098	0,009
I_3	0,6375	0,6020	0,6131	0,6131	8	0,6375	0,010
I_4	0,6754	0,6559	0,6754	0,6754	133	0,6754	0,027
I_5	n.d.	0,5038	0,5212	0,5239	118	0,5271	0,139
I_{5a}	n.d.	0,5223	0,5284	0,5299	174	0,5389	0,132
I_6	n.d.	0,4045	0,4090	0,4099	190	0,4102	5,259
I_7	n.d.	0,3757	0,3771	0,3806	161	0,3855	0,108
I_8	n.d.	0,5023	0,5086	0,5108	55	0,5124	3,557
I_{8a}	n.d.	0,5200	0,5266	0,5285	172	0,5302	3,626
I_9	n.d.	0,5031	0,5087	0,5099	13	0,5106	76,529
I_0 ($\alpha = 0{,}9$)	0,9045	0,9045	0,9045	0,9045	0	0,9045	0,003
I_1	0,7698	0,7672	0,7698	0,7698	2	0,7698	0,004
I_2	0,7465	0,7427	0,7465	0,7465	0	0,7465	0,009
I_3	0,7147	0,6966	0,6966	0,7147	0	0,7147	0,010
I_4	0,7761	0,7655	0,7714	0,7714	7	0,7761	0,027
I_5	n.d.	0,5168	0,5402	0,5474	12	0,5543	0,140
I_{5a}	n.d.	0,5243	0,5351	0,5434	16	0,5532	0,133
I_6	n.d.	0,4168	0,4226	0,4248	82	0,4254	5,645
I_7	n.d.	0,6282	0,6283	0,6319	141	0,6367	0,109
I_8	n.d.	0,5022	0,5164	0,5186	172	0,5210	3,525
I_{8a}	n.d.	0,5058	0,5198	0,5222	172	0,5245	3,484
I_9	n.d.	0,5060	0,5160	0,5187	47	0,5193	71,977

n.d.: não disponível

Tabela 6.1: Resultados obtidos pelo algoritmo GRASP (soluções iniciais para a fase de busca local geradas por AC_{100}).

α	GCG/BL_{a+b}
0,1	1,370
0,5	0,708
0,9	0,466

Tabela 6.2: Percentual médio de aumento no valor das soluções produzidas pelo algoritmo GCG em relação àquelas produzidas por BL_{a+b} para os três valores de α testados.



G_1 G_2

Figura 6.1: Representação da solução obtida pelos algoritmo GCG para o problema teste I_3 ($\alpha = 0{,}5$).

pelo algoritmo GCG tem 24 vértices de N_2 corretamente associados, 11 a mais do que a solução encontrada por $AC_{100} + BL_{a+b}$.

Os gráficos dos valores de f_1 relativos às soluções encontradas pelo GCG, assim como dos tempos totais de processamento desse procedimento para todos os problemas teste são apresentados nas Figuras 6.3 e 6.4. Foram considerados valores de α iguais a 0,1, 0,5 e 0,9. O comportamento do algoritmo GRASP para o PCGAD foi avaliado nas mesmas situações para as quais o algoritmo de busca local apresentado no Capítulo 5 foi submetido. Assim, os algoritmos de construção e de busca local foram executados a cada iteração GRASP considerando-se: (a) o procedimento de busca local BL_a com soluções iniciais relativas a AC_1 (AC_1+BL_a); (b) o procedimento de busca local BL_a com soluções iniciais relativas a AC_{100} (AC_{100}+BL_a); (c) o procedimento de busca local BL_{a+b} com soluções iniciais relativas a AC_1 (AC_1+BL_{a+b}); e (d) o procedimento de busca local BL_{a+b} com soluções iniciais relativas a AC_{100} ($AC_{100} + BL_{a+b}$).

Para todos os valores de α testados, constata-se que os melhores resultados obtidos para todos os problemas teste foram aqueles encontrados pelo algoritmo GCG usando a busca BL_{a+b} a partir das soluções iniciais AC_{100}. Essa conclusão pode ser

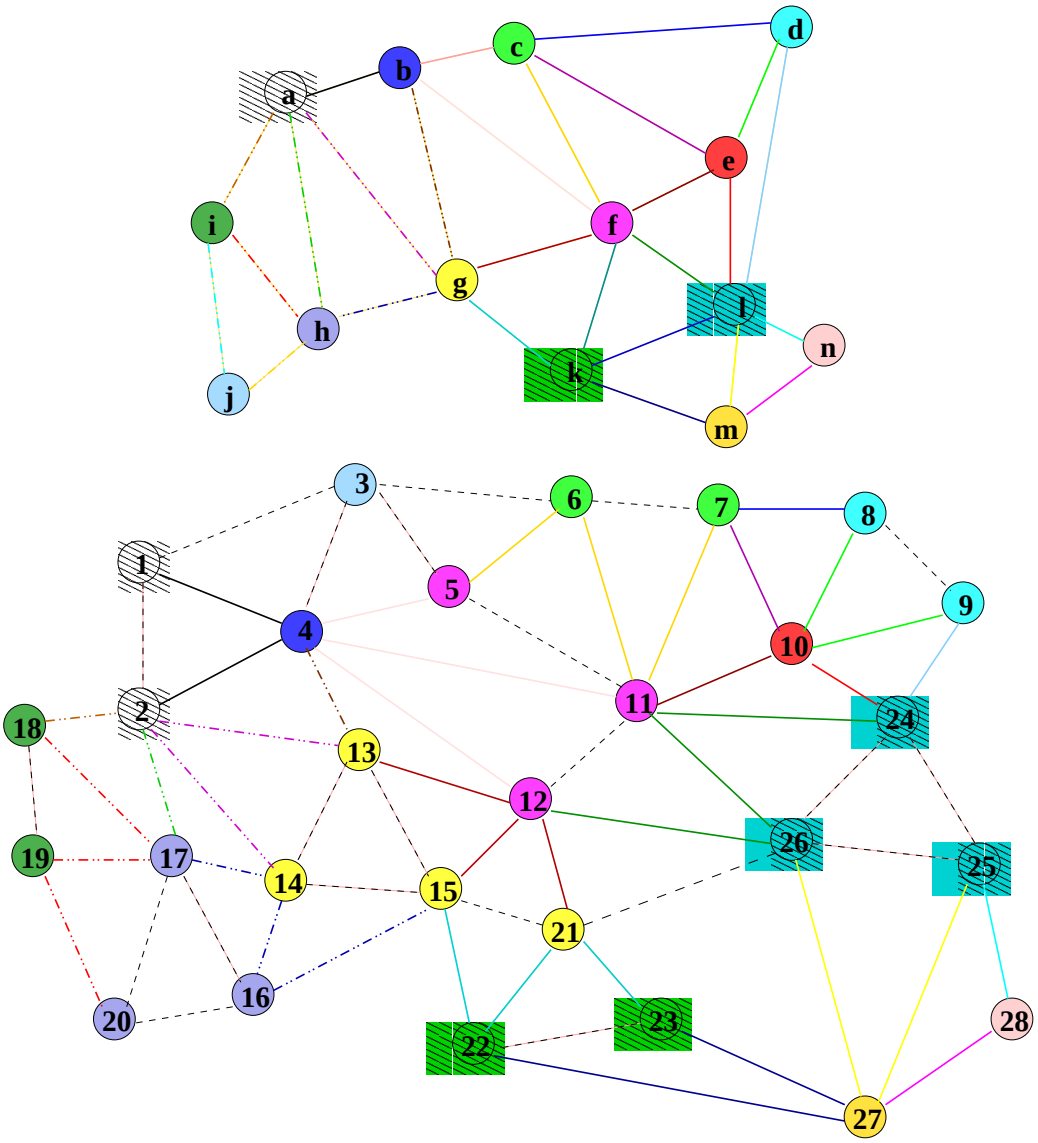


Figura 6.2: Solução obtida pela variante $AC_{100} + BL_{a+b}$ do algoritmo GCG para o problema teste I_7 com $\alpha = 0{,}5$

constatada através da Tabela 6.3, que apresenta os valores de f_1 obtidos pelo GCG nos casos (a), (b), (c) e (d) para $\alpha = 0{,}1$. Assim como aconteceu com a busca local, observa-se pelos gráficos das Figuras 6.3 e 6.4 que o comportamento do algoritmo GCG é semelhante nos quatro casos avaliados. Considerando-se os 36 problemas teste (12 intâncias e três valores de α), pode-se observar pelos resultados da Tabela 6.3 que a melhor solução é encontrada em 15 ocasiões pela combinação do caso (a), em 16 ocasiões no caso (b), em 14 ocasiões no caso (c) e em 35 ocasiões no caso (d), confirmando-se a superioridade da estratégia $AC_{100} + BL_{a+b}$. Os tempos totais de processamento dos procedimentos GRASP avaliados ($AC_1 + BL_a$, $AC_1 + BL_{a+b}$, $AC_{100} + BL_a$ e $AC_{100} + BL_{a+b}$) foram muito próximos, exceto para o problema teste

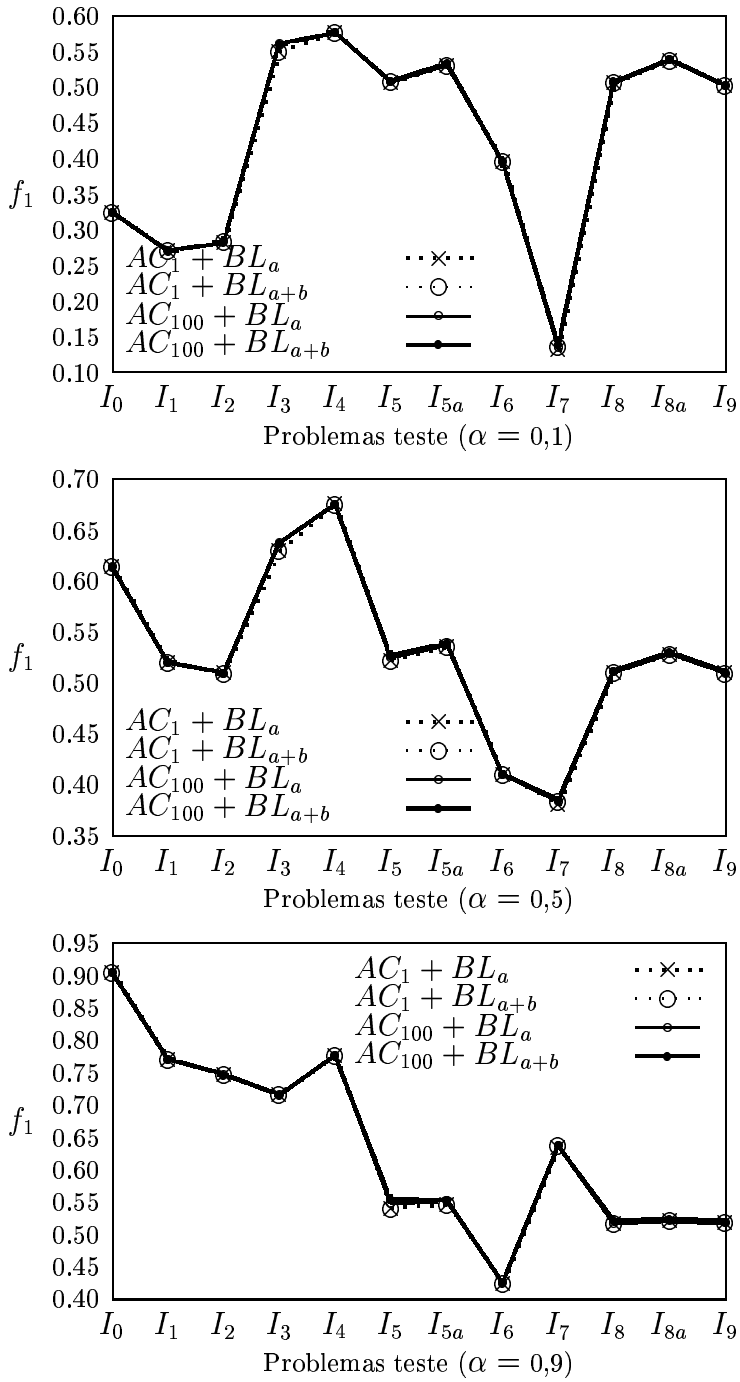


Figura 6.3: Valores de f_1 relativos às soluções obtidas por *GCG* para todos os problemas teste

I_9 (Figura 6.4).

A análise acima efetuada do comportamento das quatro variantes do algoritmo *GCG* testadas para 200 iterações (AC_1+BL_a, $AC_{100}+BL_a$, AC_1+BL_{a+b} e $AC_{100}+BL_{a+b}$) permitiu reafirmar a vantagem de utilização do algoritmo *GCG* usando a versão AC_{100} do algoritmo construtivo e a versão BL_{a+b} da busca local. Em

	GCG			
Problema teste	$AC_1 + BL_a$	$AC_{100} + BL_a$	$AC_1 + BL_{a+b}$	$AC_{100} + BL_{a+b}$
I_0 ($\alpha = 0{,}1$)	0,3240	0,3240	0,3240	0,3240
I_1	0,2707	0,2707	0,2707	0,2707
I_2	0,2818	0,2818	0,2818	0,2818
I_3	0,5497	0,5603	0,5497	0,5603
I_4	0,5766	0,5766	0,5766	0,5766
I_5	0,5071	0,5071	0,5071	0,5086
I_{5a}	0,5300	0,5306	0,5300	0,5337
I_6	0,3955	0,3955	0,3955	0,3955
I_7	0,1316	0,1353	0,1358	0,1395
I_8	0,5059	0,5061	0,5059	0,5071
I_{8a}	0,5374	0,5378	0,5374	0,5388
I_9	0,5020	0,5021	0,5020	0,5024
I_0 ($\alpha = 0{,}5$)	0,6142	0,6142	0,6142	0,6142
I_1	0,5202	0,5202	0,5202	0,5202
I_2	0,5098	0,5098	0,5098	0,5098
I_3	0,6293	0,6375	0,6294	0,6375
I_4	0,6754	0,6754	0,6754	0,6754
I_5	0,5220	0,5246	0,5220	0,5271
I_{5a}	0,5351	0,5373	0,5351	0,5389
I_6	0,4107	0,4101	0,4102	0,4102
I_7	0,3806	0,3834	0,3837	0,3855
I_8	0,5101	0,5108	0,5101	0,5124
I_{8a}	0,5279	0,5285	0,5279	0,5302
I_9	0,5097	0,5095	0,5097	0,5106
I_0 ($\alpha = 0{,}9$)	0,9045	0,9045	0,9045	0,9045
I_1	0,7698	0,7698	0,7698	0,7698
I_2	0,7465	0,7465	0,7465	0,7465
I_3	0,7147	0,7147	0,7147	0,7147
I_4	0,7761	0,7761	0,7761	0,7761
I_5	0,5408	0,5481	0,5407	0,5543
I_{5a}	0,5463	0,5518	0,5463	0,5532
I_6	0,4252	0,4253	0,4252	0,4254
I_7	0,6342	0,6363	0,6365	0,6367
I_8	0,5166	0,5170	0,5166	0,5210
I_{8a}	0,5202	0,5205	0,5207	0,5245
I_9	0,5174	0,5179	0,5174	0,5193

Tabela 6.3: Valores de f_1 obtidos pelas variantes (a) $AC_1 + BL_a$, (b) $AC_{100} + BL_a$, (c) $AC_1 + BL_{a+b}$ e (d) $AC_{100} + BL_{a+b}$ do algoritmo GCG.

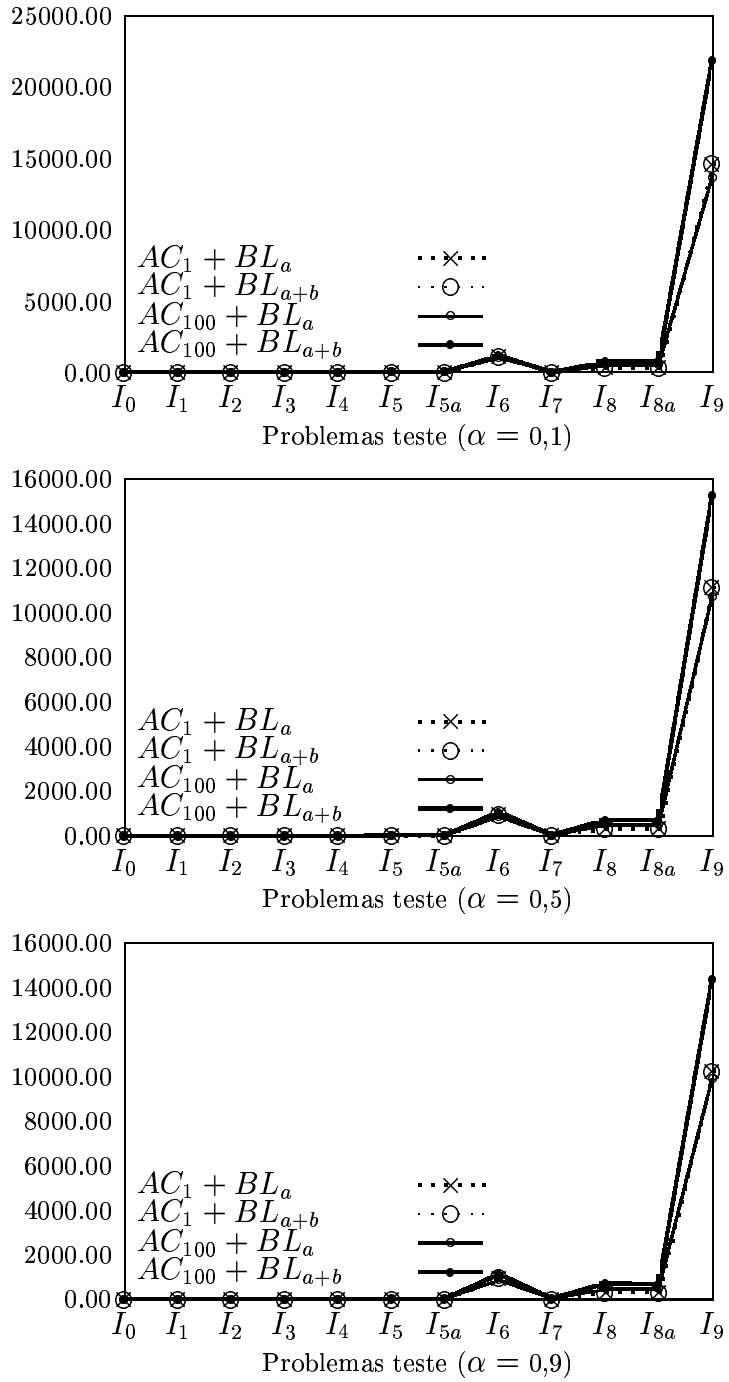


Figura 6.4: Tempo total de processamento do procedimento *GCG* para todos os problemas teste

cada iteração GRASP do *GCG* com $AC_{100} + BL_{a+b}$, são construídas 100 soluções e apenas uma (a melhor delas) é utilizada como inicial para a busca local. No caso $AC_1 + BL_{a+b}$, uma nova solução é construída e utilizada como inicial para a busca local em cada iteração GRASP. Na variante $AC_1 + BL_{a+b}$ são construídas em 100 iterações GRASP o mesmo número de soluções geradas em apenas uma iteração do

GRASP com a variante $AC_{100} + BL_{a+b}$. Devido à maior diversificação de soluções iniciais no caso $AC_{100} + BL_{a+b}$, melhores soluções são obtidas por esta variante do GCG com 200 iterações. Além disso, os tempos totais de processamento das duas variantes são muito próximos.

A fim de avaliar o comportamento do GCG com $MaxIter$ igual ao número total de soluções iniciais em 200 iterações da variante $AC_{100} + BL_{a+b}$, a variante $AC_1 + BL_{a+b}$ foi executada para o problema teste I_7 com $MaxIter = 20000$. Em ambos casos foram gerados 20000 soluções iniciais, mas o número de buscas locais aumentou de 200 para 20000. Foram realizados testes para $\alpha = 0{,}1$, $\alpha = 0{,}5$ e 0,9. No caso $\alpha = 0{,}5$, a melhor solução obtida pela variante $AC_1 + BL_{a+b}$ com 20000 iterações (em relação à solução mais adequada representada na Figura 4.5) tem 26 dos 28 vértices de N_2 corretamente associados. Apenas os vértices 5 e 21 não foram identificados. O valor de f_1 para essa solução é 0,3869, aproximadamente 0,4% maior do que o valor da solução encontrada pela variante $AC_{100} + BL_{a+b}$ executada para 200 iterações e com $f_1 = 0{,}3855$ (Figura 6.2). Observa-se que embora a melhoria no valor da solução seja pequena, a nova solução reconhece corretamente dois vértices a mais do que a anterior. Entretanto, o tempo total de processamento do GCG para o problema teste I_7 é de 216,56 segundos, mesmo com esse elevado número de iterações. O tempo total de processamento da variante $AC_{100} + BL_{a+b}$ com 200 iterações para o mesmo problema teste é de 21,78 segundos, quase dez vezes mais rápido que no caso de 20000 iterações da variante $AC_1 + BL_{a+b}$. Observa-se que embora o número de buscas locais seja 100 vezes maior (aumentando de 200 para 20000), o tempo total de processamento é multiplicado apenas por 9,94. O aumento no número de buscas locais não compromete significativamente o tempo total de execução e permite melhorar a qualidade das soluções.

Os gráficos apresentados na Figura 6.5 mostram os valores de f_1 relativos às melhores soluções obtidas pelo algoritmo GCG (usando AC_{100} e BL_{a+b}) ao longo de 1000 iterações realizadas, considerando-se os problemas teste I_6 e I_7 e $\alpha = 0{,}1$. Através da análise desses gráficos, pretende-se avaliar o impacto do número de iterações GRASP na qualidade da solução final obtida por GCG. Os problemas teste I_6 e I_7 foram escolhidos para essa análise por serem obtidos a partir de exemplos reais e pela quantidade relativamente grande de vértices e de arestas de seus grafos. No caso do problema teste I_6 (grafos maiores), o algoritmo GCG gera rapidamente (em menos de 50 iterações) uma solução cujo valor está a menos de 0,005% do valor

da melhor solução encontrada (obtida posteriormente na iteração 473). Já no caso do problema teste I_7 (grafos menores), o algoritmo GCG encontra a melhor solução na iteração 170. Neste caso, o valor da melhor solução encontrada pelo GRASP (em menos de 200 iterações) é maior en 6,2% do que aquele da solução encontrada pela busca local. Embora o algoritmo GRASP encontre rapidamente uma boa solução, esta normalmente pode ser sucessivamente melhorada através da realização de um maior número de iterações GRASP.

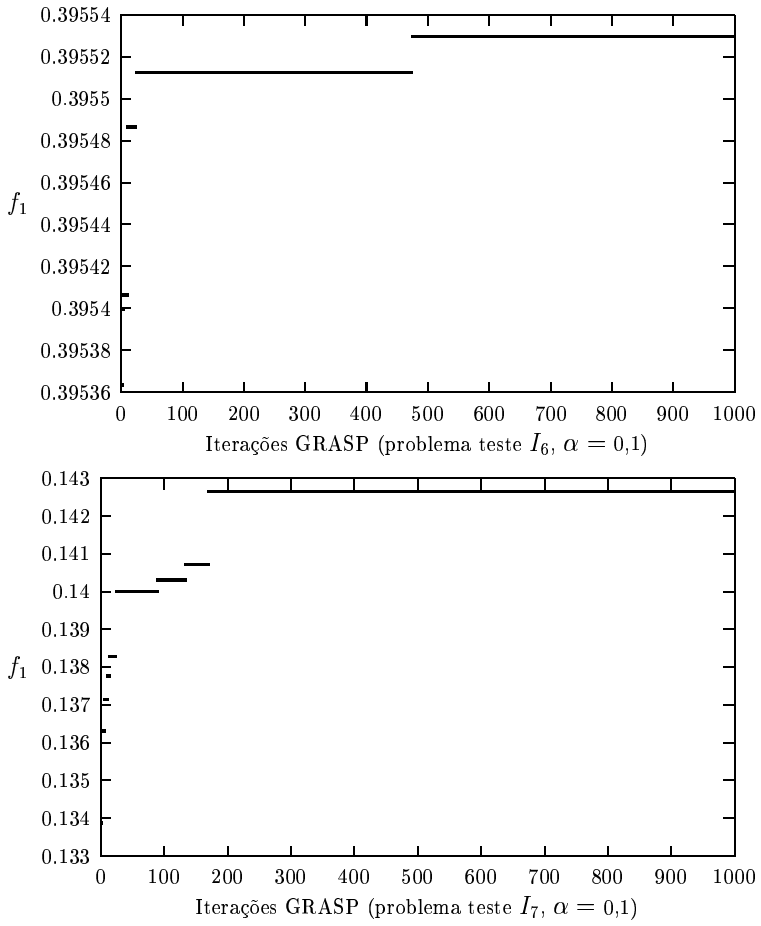


Figura 6.5: Valores de f_1 relativos à melhor solução obtida ao longo de 1000 iterações do GCG para os problemas teste I_6 e I_7

6.3 Conclusões

O algoritmo GCG permite melhorar as soluções obtidas pelos algoritmos construtivo e de busca local para todos os problemas teste. Como os demais, este algoritmo é sensível à variação de valor do parâmetro α. Para os grafos do problema teste I_7 com $\alpha = 0{,}9$, o algoritmo GCG obtém uma solução que reconhece corretamente apenas

sete das 28 regiões da imagem. Para $\alpha = 0{,}1$, o número de regiões identificadas cresce para 21. A solução obtida para $\alpha = 0{,}5$ aumenta para 24 o número de regiões corretamente reconhecidas.

Os experimentos realizados com o problema teste I_7 mostraram que o algoritmo GCG com 200 iterações de $AC_{100} + BL_{a+b}$ foi capaz de gerar uma solução muito próxima daquela conhecida como a mais adequada em termos do reconhecimento da imagem supersegmentada. Com a solução da variante $AC_1 + BL_{a+b}$ do algoritmo GCG para 20000 iterações, foi possível reconhecer quase a totalidade das regiões (apenas duas regiões não foram identificadas) através do aumento de buscas locais, sem onerar em demasia o tempo total de processamento. Boas soluções são obtidas mesmo com um número relativamente pequeno de iterações GRASP. Os melhores resultados em relação ao reconhecimento da imagem supersegmentada foram aqueles obtidos pela variante $AC_1 + BL_{a+b}$ do algoritmo GCG com 20000 iterações. De acordo com estes experimentos, esta é a melhor versão do algoritmo GCG.

Capítulo 7

Conclusões

Estudou-se nesta tese o problema de reconhecimento de cenas em imagens complexas. A motivação para o estudo deste problema surgiu de uma aplicação na área de reconhecimento de imagens médicas. Nesta aplicação, pretende-se efetuar o reconhecimento de estruturas tridimensionais do cérebro humano, a partir de imagens obtidas por ressonância magnética.

Várias são as abordagens propostas na literatura para o tratamento de problemas de reconhecimento de imagens através de diferentes técnicas de correspondência de grafos. Nesta tese, o problema de reconhecimento é tratado como a determinação da correspondência ótima entre grafos que representam a imagem a ser reconhecida e um modelo conhecido (atlas).

Propõe-se inicialmente uma formulação original como um problema de otimização em variáveis 0-1 para a determinação da correspondência ótima entre um grafo-imagem e um grafo-modelo. Diferentes funções objetivo (baseadas nos valores de similaridade entre nós e entre arestas) são propostas e investigadas. Mostra-se que uma destas funções é mais apropriada, pelo menos para os exemplos considerados. Esta função é utilizada para os desenvolvimentos algorítmicos posteriores. Entretanto, o modelo e as técnicas de solução propostas independem da função objetivo considerada. Estes podem ser facilmente estendidos ou adaptados, de forma a considerar outras funções objetivo mais adequadas, que venham a ser identificadas por especialistas no problema de reconhecimento de cenas. Ao longo da tese são descritos experimentos computacionais sobre 36 problemas teste, definidos por 12 pares de grafos imagem-atlas e três variantes da função objetivo escolhida.

Devido à complexidade computacional do problema de correspondência ótima

de grafos, diversos algoritmos aproximados foram propostos para sua resolução. O primeiro é uma heurística construtiva randomizada, que gera soluções viáveis. Este algoritmo pode ser aplicado a partir de múltiplas sementes iniciais utilizadas pelo gerador de números pseudo-aleatórios empregado, sendo retida a melhor solução construída ao longo das diferentes tentativas.

A segunda heurística é um procedimento de busca local aplicado às soluções construídas pelo algoritmo randomizado. São propostas e utilizadas duas diferentes estruturas de vizinhança. De forma similar aos métodos de descida em vizinhanças variáveis (VND), a segunda vizinhança só é explorada a partir de ótimos locais em relação à primeira. Os algoritmos de busca local permitem melhorar sistematicamente as soluções construídas pelo algoritmo construtivo. Tomando-se como exemplo o caso do problema teste I_7 com $\alpha = 0{,}5$, observa-se que o algoritmo construtivo (versão AC_{100}, correspondente à seleção da melhor solução obtida após 100 tentativas, cf. Seção 4.3.2) obtém uma solução de custo 0,3757 e reconhece corretamente 7 das 28 regiões da imagem. O algoritmo completo de busca local utilizando as duas vizinhanças (versão BL_{a+b}, cf. Seção 5.5) melhora o valor desta solução para 0,3806, mas o número de regiões corretamente reconhecidas permanece o mesmo.

A terceira e mais elaborada heurística é uma implementação da metaheurística GRASP, combinando os dois procedimentos anteriores de construção e de busca local. Mostra-se que a heurística GRASP encontra boas soluções subótimas para problemas associados a grafos com centenas de nós e milhares de arestas. Embora a qualidade das soluções produzidas por esta heurística melhore com o número de iterações, poucas iterações (algumas centenas) são suficientes para melhorar significativamente a solução encontrada pela combinação de uma aplicação do algoritmo construtivo seguida de busca local.

No caso do problema teste I_7 com $\alpha = 0{,}5$ e considerando-se a execução de 200 iterações da versão $AC_{100} + BL_{a+B}$, a heurística GCG descrita na Seção 6.1 encontra em 21,78 segundos uma solução de valor 0,3855 que reconhece corretamente 24 regiões da imagem. Finalmente, mostra-se que a execução de um maior número de iterações GRASP utilizando a versão mais simples (AC_1) do algoritmo construtivo permite melhorar ainda mais estes resultados. Considerando-se o mesmo problema teste I_7 com $\alpha = 0{,}5$, a versão $AC_1 + BL_{a+b}$ do algoritmo GCG encontra uma solução de valor 0,3869 e reconhece corretamente 26 regiões após a realização de

20.000 iterações em 216,56 segundos.

Conforme mencionado anteriormente, todos estes algoritmos propostos podem ser facilmente adaptados para outras funções objetivo que sejam eventualmente mais apropriadas ou para outras formulações que envolvam outros tipos de restrições e condições. Devido à sua reduzida complexidade, todos os algoritmos são rápidos.

Extensões tanto do modelo quanto das heurísticas são propostas como trabalhos futuros. Pretende-se investigar mais detalhadamente o efeito da incorporação da Condição 2.3 à função objetivo, considerando-se problemas teste mais elaborados. Pretende-se também investigar extensões do algoritmo GRASP, baseadas por exemplo na estratégia de intensificação por reconexão de caminhos [54]. Outra extensão deste trabalho consiste na comparação sistemática dos resultados obtidos pela heurística GRASP com aqueles obtidos por outras heurísticas encontradas na literatura, tais como o algoritmo genético descrito em [52] e o algoritmo probabilístico proposto em [11]. Estes estudos envolverão não apenas os problemas teste considerados nesta tese, mas também aplicações reais provenientes principalmente do reconhecimento de imagens faciais.

Bibliografia

[1] E. H. L. Aarts e M. Verhoeven. Local search. Em *Annotated bibliographies in combinatorial optimization*, editado por M. Dell'Amico, F. Maffioli e S. Martello, págs. 163–180. Wiley, 1997.

[2] E. Bengoetxea, P. Larrañaga, I. Bloch, A. Perchant e C. Boeres. Inexact graph matching by means of estimation of distribution algorithms. *Pattern Recognition*, 35:2867–2880, 2002.

[3] C. Berge. *Graphs*. Prentice-Hall, 1983.

[4] R. E. Blake. Partitioning graph matching with constraints. *Pattern Recognition*, 27:439–446, 1994.

[5] I. Bloch. Fuzzy spatial relationships: A few tools for model-based pattern recognition in aerial images. *Proceedings of SPIE*, 2955:141–152, 1996.

[6] I. Bloch. Fuzzy relative position between objects in image processing: A morphological approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:657–664, 1999.

[7] I. Bloch. On fuzzy distances and their use in image processing under imprecision. *Pattern Recognition*, 32:1873–1895, 1999.

[8] B. G. Buchanan e E. H. Shortliffe. *Rule-based expert system - The MYCIN experiments of the Stanford heuristic programming project*. Addison-Wesley, 1984.

[9] H. Bunke e B. T. Messmer. Recent advances in graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 11:169–203, 1997.

[10] V. Cantoni, L. Cinque, C. Guerra, S. Levialdi e L. Lombardi. 2-D object recognition by multiscale tree matching. *Pattern Recognition*, 31:1443–1454, 1998.

[11] R. Cesar, E. Bengoetxea e I. Bloch. Inexact graph matching using stochastic optimization techniques for facial feature recognition. Em *International Conference in Pattern Recognition, ICPR 2002*, Quebec, 2002.

[12] L. J. Chipman e H. S. Ranganath. A fuzzy relaxation algorithm for matching imperfectly segmented images to models. Em *Proceedings of the SoutheastCon Region 3 Conference of the Institute of Electrical and Electronics Engineers*, volume 1, págs. 128–136, Birmingham, 1992.

[13] A. D. J. Cross e E. R. Hancock. Relational matching with stochastic optimization. Em *Proceedings of the IEEE Computer Society International Symposium on Computer Vision*, págs. 365–370, Ascona, 1995.

[14] A. D. J. Cross e E. R. Hancock. Inexact graph matching with genetic search. *Lectures Notes in Computer Science*, 1121:150–159, 1996.

[15] A. D. J. Cross e E. R. Hancock. Graph matching with a dual-step EM algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:1236–1253, 1998.

[16] A. D. J. Cross, R. Myers e E. R. Hancock. Convergence of a hill-climbing genetic algorithm for graph matching. *Pattern Recognition*, 33:1863–1880, 2000.

[17] A. D. J. Cross, R. C. Wilson e E. R. Hancock. Genetic search for structural matching. *Lectures Notes in Computer Science*, 1064:514–525, 1996.

[18] A. D. J. Cross, R. C. Wilson e E. R. Hancock. Inexact graph matching using genetic search. *Pattern Recognition*, 30:953–970, 1997.

[19] Y. El-Sonbaty e M. A. Ismail. A new algorithm for subgraph optimal isomorphism. *Pattern Recognition*, 31:205–218, 1998.

[20] T. A. Feo e M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

[21] T. A. Feo e M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[22] P. Festa e M. G. C. Resende. GRASP: An annotated bibliography. Em *Essays and surveys in metaheuristics*, editado por C. C. Ribeiro e P. Hansen, págs. 325–367. Kluwer Academic Publishers, 2001.

[23] A. M. Finch, R. C. Wilson e E. R. Hancock. Symbolic graph matching with the EM algorithm. *Pattern Recognition*, 31:1777–1790, 1998.

[24] M. Gabrani e O. J. Tretiak. Surface-based matching using elastic transformations. *Pattern Recognition*, 32:87–97, 1999.

[25] J. C. Gee. On matching brain volumes. *Pattern Recognition*, 32:99–111, 1999.

[26] D. Geman e S. Geman. Stochastic relaxation, gibbs distributions and bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[27] F. Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49:231–255, 1994.

[28] F. Glover, J. P. Kelly e M. Laguna. Genetic algorithms and tabu search: Hybrids for optimization. *Computers and Operations Research*, 22:111–134, 1995.

[29] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[30] E. R. Hancock e J. Kittler. Discrete relaxation. *Pattern Recognition*, 23:711–733, 1990.

[31] J. Holland. *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor, 1975.

[32] E. Horowitz e S. Sahni. *Fundamentals of computer algorithms*. Computer Science Press, 1978.

[33] S. Kirkpatrick, C. D. Gellat e M. P. Vecchi. Optimisation by simulated annealing. *Science*, 220:671–680, 1983.

[34] P. Larrañaga, Y. Yurramendi, R. H. Murga e C. M. H. Kuijpers. Structure learning of bayesian networks by genetic algorithms: A performance analysis of control parameters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18:912–926, 1996.

[35] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics-Doklady*, 10:707–710, 1966.

[36] J. F. Mangin, V. Frouin, J. Régis, I. Bloch, P. Belin e Y. Samsin. Towards better management of cortical anatomy in multi-modal multi-individual brain studies. *Physica Medica*, XII:103–107, 1996.

[37] J. F. Mangin, J. Regis, I. Bloch, V. Frouin, Y. Samson e J. Lopez-Krahe. A MRF based random graph modelling the human cortical topography. Em *Computer Vision, Virtual Reality and Robotics in Medicine*, editado por N. Ayache, págs. 177–183, Nice, 1995.

[38] D. Marr. *Vision*. Freeman, 1982.

[39] H. Maître, I. Bloch, H. Moissinac e C. Gouinaud. Cooperative use of aerial images and maps for the interpretation of urban scenes. Em *Automatic Extraction of Man-Made Objects from Aerial and Space Images Proceedings*, editado por A. Gruen, O. Kuebler e P. Agouris, págs. 297–306, Ascona, 1995.

[40] B. T. Messmer e H. Bunke. A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20:493–504, 1998.

[41] B. T. Messmer e H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32:1979–1998, 1999.

[42] H. Moissinac, H. Maître e I. Bloch. Urban aerial image understanding using symbolic data. *Proceedings of SPIE*, 2315:310–321, 1994.

[43] H. Moissinac, H. Maître e I. Bloch. Markov random fields and graphs for uncertainty management and symbolic data fusion in a urban scene interpretation. *Proceedings of SPIE*, 2579:298–309, 1995.

[44] R. Myers e E. R. Hancock. Genetic algorithms for ambiguous labelling problems. *Pattern Recognition*, 33:685–704, 2000.

[45] R. Myers e E. R. Hancock. Least-commitment graph matching with genetic algorithms. *Pattern Recognition*, 34:375–394, 2000.

[46] R. Myers, R. C. Wilson e E. R. Hancock. Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:628–635, 2000.

[47] N. K. Nilsson. *Principles of artificial intelligence*. Morgan Kaufmann, 1980.

[48] A. Pearce, T. Caelli e W. F. Bischof. Rulegraphs for graph matching in pattern recognition. *Pattern Recognition*, 27:1231–1247, 1994.

[49] A. Perchant. *Morphisme de graphes d'attributs flous pour la reconnaissance structurelle de scènes (Application à la reconnaissance de structures anatomiques saines et pathologiques sur des IRM).* Tese de Doutorado, École Nationale Supérieure des Télécommunications, Paris, 2000.

[50] A. Perchant e I. Bloch. A new definition for fuzzy attributed graph homomorphism with application to structural shape recognition in brain imaging. Em *Proceedings of 16th IEEE Conference on Instrumentation and Measurement Technology*, págs. 402–410, Veneza, 1999.

[51] A. Perchant e I. Bloch. Semantic spatial fuzzy attribute design for graph modeling. Em *Proceedings of 16th IEEE Conference on Instrumentation and Measurement Technology*, págs. 1801–1906, Veneza, 1999.

[52] A. Perchant, C. Boeres, I. Bloch, M. Roux e C. C. Ribeiro. Model-based scene recognition using graph fuzzy homomorphism solved by genetic algorithm. Em *Proceedings of the 2nd IAPR-TC-15 Workshop on Graph-based Representations*, págs. 61–70, Hansdorf, 1999.

[53] H. S. Ranganath e L. J. Chipman. Fuzzy relaxaton approach for inexact scene matching. *Image and Vision Computing*, 10:631–640, 1992.

[54] M. G. C. Resende e C. C. Ribeiro. Greedy randomized adaptive search procedures. Em *State-of-the-Art Handbook of Metaheuristics*, editado por F. Glover e G. Kochenberger. Kluwer Academic Publishers, a ser publicado.

[55] K. Rohr. Extraction of 3D anatomical point landmarks based on invariance principles. *Pattern Recognition*, 32:3–15, 1999.

[56] A. Rosenfeld. Fuzzy graphs. Em *Fuzzy sets and their applications to cognitive and decision processes*, editado por L. A. Zadeh, K. S. Fu, K. Tanaka e M. Shimura, págs. 77–95. Academic Press, New York, 1975.

[57] A. Rosenfeld, R. Hummel e S. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man and Cybernetics*, 6:420–433, 1976.

[58] A. Sanfeliu e K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13:353–362, 1983.

[59] L. Schrage. A more portable FORTRAN random number generator. *ACM Transactions on Mathematical Software*, 5:132–138, 1979.

[60] L. G. Shapiro e R. M. Haralick. Organization of relational models for scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:595–602, 1982.

[61] L. G. Shapiro e R. M. Haralick. A metric for comparing relational descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:90–94, 1985.

[62] M. Singh, A. Chatterjee e S. Chaudhury. Matching structural shape description using genetic algorithm. *Pattern Recognition*, 30:1451–1462, 1997.

[63] P. N. Suganthan, H. Yan, E. K. Teoh e D. P. Mital. Optimal encoding of graph homomorphism energy using fuzzy information aggregation operators. *Pattern Recognition*, 31:623–639, 1998.

[64] M. L. Williams, R. C. Wilson e E. R. Hancock. Deterministic search for relational graph matching. *Pattern Recognition*, 32:1255–1271, 1999.

[65] R. C. Wilson, A. N. Evans e E. R. Hancock. Relational matching by discrete relaxation. *Image and Vision Computing*, 13:411–421, 1995.

[66] R. C. Wilson e E. R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analisys and Machine Intelligence*, 19:634–648, 1997.

[67] A. K. C. Wong e M. You. Entropy and distance of random graphs with application to structural pattern recognition. *IEEE Transactions Analysis and Machine Intelligence*, 7:599–609, 1985.

[68] A. K. C. Wong, M. You e S. C. Chan. An algorithm for graph optimal monomorphism. *IEEE Transactions on Systems, Man and Cybernetics*, 20:628–636, 1990.

[69] E. K. Wong. Model matching in robot vision by subgraph isomorphism. *Pattern Recognition*, 25:287–303, 1992.

[70] A. L. Yuille, P. Stolorz e J. Utans. Statistical physics, mixtures of distributions and the EM algorithm. *Neural Computation*, 6:334–340, 1994.