UNIVERSIDADE FEDERAL FLUMINENSE

GUSTAVO SILVA SEMAAN

Algoritmos Heurísticos para o Problema de Particionamento de Grafos com Restrições de Capacidade e Conexidade

NITERÓI-RJ
2010

UNIVERSIDADE FEDERAL FLUMINENSE

GUSTAVO SILVA SEMAAN

Algoritmos Heurísticos para o Problema de Particionamento de Grafos com Restrições de Capacidade e Conexidade

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Orientador:
Prof. Luiz Satoru Ochi, D.Sc.

NITERÓI-RJ
2010

Algoritmos Heurísticos para o Problema de Particionamento de Grafos com Restrições de Capacidade e Conexidade

Gustavo Silva Semaan

Dissertação de Mestrado submetida ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do título de Mestre. Área de concentração: Otimização Combinatória e Inteligência Artificial.

Aprovada por:

_____

Prof. Luiz Satoru Ochi, IC-UFF
Presidente

_____

Prof. José Andre Moura Brito, ENCE-IBGE

_____

Profa. Loana Tito Nogueira, IC-UFF

_____

Prof. Nelson Maculan Filho, COPPE-UFRJ

_____

Prof. Virgilio José Martins Ferreira Filho, COPPE-UFRJ

Niterói, 02 de Fevereiro de 2010.

Aos meus pais Bill e Isa,
ao meu irmão Felipe
e a minha noiva Débora.

Agradecimentos

A Deus por me iluminar e me acompanhar rumo a meus objetivos.

Aos meus pais William e Isabel, ao meu irmão Felipe e minhas avós Zélia e Cecília, por todo o amor, carinho e por estarem sempre presentes em minha vida.

À minha noiva Débora pelo amor, carinho, paciência e por estar incondicionalmente ao meu lado em todos os momentos de minha vida.

Ao professor Luiz Satoru Ochi pela confiança depositada, desde minha inscrição para seleção no programa de mestrado da UFF, ao apoio e orientações durante as disciplinas e na dissertação.

Ao professor José André de Moura Brito pela atenção, amizade e por sua orientação, fundamental para o desenvolvimento desse trabalho.

Aos amigos de república Carlos, Bruno e Felipe, que me acolheram durante as dificuldades e, juntos, formamos um lar em Niterói.

Ao professor Carlos Rodrigo Dias pela amizade, orientação na graduação, e por ser o maior incentivador do prosseguimento de meus trabalhos científicos, inclusive para minha inscrição no programa de mestrado.

À professora Melise Paula pela amizade, incentivo, apoio e carinho.

Aos professores do Centro de Ensino Superior de Juiz de Fora, em especial ao prof. Marco Antônio Pereira Araújo, profa. Alessandreia Marta de Oliveira Julio, prof. Allan Fonseca da Silva e José Honório Glanzmann.

Aos funcionários do CAEd / UFJF, em especial ao Eurico, Roberta, Gilson e Manuel pelo companheirismo, apoio e amizade.

Aos professores, funcionários e alunos do Instituto de Computação da Universidade Federal Fluminense, pelo apoio e ensinamentos.

Resumo

O presente trabalho de dissertação propõe algoritmos para a resolução de um particular problema real de agrupamento com restrições de capacidade e conexidade. Assim como em outros problemas de agrupamento, tal problema pode ser mapeado em um problema de particionamento de grafos em subgrafos disjuntos, estando cada subgrafo associado a um *cluster*. Acrescenta-se ainda, que no processo de particionamento do grafo são consideradas as restrições de conexidade e capacidade. Para a resolução deste problema foram propostos algoritmos heurísticos (Evolutivos e GRASPs) e, para a realização de experimentos, foram utilizadas instâncias reais obtidas em Censos Demográficos e Pesquisas Socioeconômicas.

Palavras-chave: Algoritmo Evolutivo; GRASP; Particionamento de grafos; clusterização; regionalização.

Abstract

The present work proposes algorithms for the resolution of a real capability and conexity graph partition problem. As in other clustering problems, this problem can be mapped by a graph partition into subtrees, each one of these subtrees will be associated to one cluster. Moreover, in the graph partition process, it must be considered the restrictions of this problem. For the resolution of this problem, this work proposes heuristics algorithms (Evolutionary and GRASP) and, for the experiments analyses, were used real instances of Demographic Census and Socio-economic Research.

Keywords: Evolutionary Algorithm, GRASP; Graph Partitioning; Clustering; Regionalization

Siglas e Abreviações

AE	:	Algoritmo Evolutivo
AGM	:	Árvore Geradora Mínima
GRASP	:	*Greed Randomized Adaptive Search Procedure*
AZP	:	*Automatic Zoning Procedure*
LRC	:	Lista Restrita de Candidatos
AE	:	Algoritmo Evolutivo
K	:	Quantidade de clusters
BL	:	Busca Local
MUT	:	Mutação
Cross	:	Cruzamento (*crossover*)
APA	:	Área de Ponderação Agregada
APOND	:	Área de Ponderação
Const	:	Procedimento Construtor

Sumário

SIGLAS E ABREVIAÇÕES VIII

LISTA DE FIGURAS XI

LISTA DE ALGORITMOS XIII

LISTA DE TABELAS XIV

1 INTRODUÇÃO 1

1.1 Problema de Agrupamento com Restrições de Capacidade e Conexidade 1
1.2 Objetivos do Trabalho 2
1.3 Objetivos Específicos 2
1.4 Estrutura do Trabalho 3

2 PROBLEMA DE AGRUPAMENTO 4

2.1 Motivação 4
2.2 Definição do Processo de Agrupamento 6
2.3 Análise da Qualidade do Agrupamento 9
2.4 Problema de Agrupamento com Restrições de Capacidade e Conexidade 10
2.4.1 Abordagem utilizando Automatic Zoning Procedure 11
2.4.2 Abordagem baseada na construção da Árvore Geradora Mínima 12
2.5 Problemas Abordados 17

3 ALGORITMOS 20

3.1 Algoritmos Genéticos 20
3.2 GRASP 24
3.3 Heurísticas e Procedimentos Implementados 25
3.3.1 Representação de uma Solução 25
3.3.2 Procedimentos para Construção de Soluções Iniciais 27
3.3.3 Procedimentos de Busca Local 36
3.3.4 Procedimento de Mutação 46
3.3.5 Procedimentos de Cruzamento 47
3.3.6 Procedimento de Seleção 48
3.4 Algoritmos Heurísticos Propostos 50
3.4.1 Algoritmos Evolutivos Propostos 50
3.4.2 GRASP 52

4 RESULTADOS COMPUTACIONAIS 53

4.1 Instâncias Utilizadas 53
4.1.1 Áreas de Ponderação 54
4.1.2 Municípios 55
4.2 Normalização dos Dados 55

4.3 ALGORITMOS....56
4.3.1 Algoritmos Evolutivos....57
4.3.2 GRASP....58
4.4 PARÂMETROS DOS ALGORITMOS....59
4.5 EXPERIMENTOS REALIZADOS....60
4.5.1 Experimentos com Procedimentos Construtores....60
4.5.2 Experimentos com os Algoritmos Propostos....64
4.5.3 Análise Probabilística....72
5 CONCLUSÕES E TRABALHOS FUTUROS....79
5.1 TRABALHOS FUTUROS....80
REFERÊNCIAS....83

Lista de Figuras

Figura 2.1: Exemplo de agrupamento de figuras. 5
Figura 2.2: Representação de similaridades intraclusters e interclusters [Larose, 2006]. 6
Figura 2.3: Exemplo de cálculo do Custo$_e$. 14
Figura 2.4: Representação de subregiões e clusters através de grafos [Assunção et al., 2006]. 19
Figura 3.1: Exemplo de cruzamento de um ponto 21
Figura 3.2: Exemplo de cruzamento de dois pontos 21
Figura 3.3: Exemplo de mutação 22
Figura 3.4: Representação de uma solução. 26
Figura 3.5: Exemplo de aplicação do procedimento construtor 1. 29
Figura 3.6: Seleção de dois vértices para o procedimento construtor 2. 30
Figura 3.7: Identificação da subárvore principal na AGM. 31
Figura 3.8: Formação dos grupos de vértices na subárvore principal. 32
Figura 3.9: Exemplo de execução do procedimento construtor 2 33
Figura 3.10: Exemplo de execução do procedimento construtor 2 com fator aleatório. 34
Figura 3.11: Exemplo de execução do procedimento construtor 2 regenerando a solução. 35
Figura 3.12: Exemplo da aplicação do procedimento de busca local 1 38
Figura 3.13: Exemplo de possibilidades de migração de vértices 38
Figura 3.14. Exemplo de migração de vértice entre clusters da Busca Local 1. 39
Figura 3.15: Exemplo de execução do Procedimento de Busca Local 2. 41
Figura 3.16. Exemplo de execução do Procedimento Busca Local 4. 42
Figura 3.17: Grafo utilizado no exemplo de demonstração do procedimento construtor 3. 43
Figura 3.18: Exemplo de execução do procedimento construtor 3. 44
Figura 3.19: Procedimento Busca Local 6. 45
Figura 3.19: Exemplo de migração da primeira versão do procedimento de mutação. 47
Figura 3.20: Exemplo de aplicação do procedimento de mutação. 47
Figura 3.21: Exemplo de execução do procedimento de roleta [Glover and Kochenberger, 2002]. 49
Figura 4.1: Relação de instâncias utilizadas nos experimentos. 54
Figura 4.2: Legenda utilizada nos experimentos. 56
Figura 4.3: Algoritmos evolutivos que utilizam somente a AGM. 57
Figura 4.4: Algoritmos evolutivos que utilizam também o grafo original. 58
Figura 4.5: GRASPs que utilizam somente a AGM. 58
Figura 4.6: GRASPs que utilizam também o grafo original. 58
Figura 4.7: Outras configurações para os Algoritmos Evolutivos 59
Figura 4.8: Outras configurações para os GRASPs 59
Figura 4.9: Obtenção de soluções válidas dos procedimentos construtores por instância. 62
Figura 4.10: Obtenção de soluções válidas dos procedimentos construtores por fator de ajuste. 62

Figura 4.11: Obtenção de soluções válidas das instâncias por fator de ajuste. *63*
Figura 4.12: Quantidade de soluções por algoritmo. *68*
Figura 4.13: Quantidade de soluções (AGM ou o grafo original). *68*
Figura 4.14: Quantidade de soluções dos AEs e GRASPs. *69*
Figura 4.15: Quantidade de soluções por procedimento construtor. *69*
Figura 4.16: Tempo de execução dos AEs e GRASPs. *70*
Figura 4.17: Tempo de Execução dos algoritmos (AGM e o grafo original). *70*
Figura 4.18: Tempo de execução dos algoritmos por procedimento construtor utilizado. *70*
Figura 4.19: Aptidão por quantidade de clusters. *72*
Figura 4.21: Probabilidade acumulada alvo fácil instância 4. ..... *75*
Figura 4.22: Probabilidade acumulada alvo médio instância 4. ..... *75*
Figura 4.23: Probabilidade acumulada alvo difícil instância 4. ..... *76*
Figura 4.25: Probabilidade acumulada alvo fácil instância 13. ..... *77*
Figura 4.26: Probabilidade acumulada alvo médio instância 13. ..... *77*
Figura 4.27: Probabilidade acumulada alvo difícil instância 13. ..... *78*
Figura 4.28: Probabilidade acumulada por categoria de soluções – instância 13. ..... *78*

Lista de Algoritmos

Algoritmo 1: Algoritmo AZP (Automatic Zoning Procedure) [Neves, 2003]. 12
Algoritmo 2: Algoritmo genético tradicional. 22
Algoritmo 3: Algoritmo GRASP (Greed Randomized Adaptive Search Procedure). 25
Algoritmo 4: Algoritmo do procedimento construtor 1. 28
Algoritmo 5: Algoritmo evolutivo proposto. 51

Lista de Tabelas

Tabela 4.1: Aptidão, gap e tempo de execução por fator de ajuste. *64*
Tabela 4.2: Resultados dos AEs para a obtenção de 3 clusters..... *66*
Tabela 4.3: Resultados dos GRASPs para a obtenção de 3 clusters. *67*
Tabela 4.4: Aptidão por instância para 3, 5 e 8 grupos. *71*
Tabela 4.5: Aptidões dos algoritmos e desvio padrão para as instâncias 4 e 13. ..... *74*

Capítulo 1

Introdução

1.1 Problema de Agrupamento com Restrições de Capacidade e Conexidade

O presente trabalho de dissertação traz uma proposta de resolução para um particular problema real de agrupamento com restrições de capacidade e conexidade. Tal problema pode ser mapeado em um problema de particionamento de grafos em k subgrafos disjuntos que corresponderão aos k clusters. Acrescenta-se ainda, que no processo de particionamento do grafo são consideradas as duas restrições acima.

A restrição de conexidade (ou contigüidade) implica que, em cada cluster, todos os vértices tomados dois a dois, devem ser vizinhos ou deve existir pelo menos um caminho entre estes dois vértices, passando apenas por vértices pertencentes ao mesmo *cluster*. Ou seja, tais vértices devem pertencer a um mesmo subgrafo.

Já a restrição de capacidade é definida em função de um dos atributos dos vértices, de forma que o somatório desse atributo nos vértices pertencentes a um mesmo *cluster* seja maior ou igual a um limite inferior pré-estabelecido.

Dentre as várias abordagens encontradas na literatura, para a resolução do problema de agrupamento com a restrição de conexidade, destacam-se o algoritmo AZP (*Automatic Zoning Procedure*) encontrado em [Assunção *et al.*, 2006] e um algoritmo baseado na utilização da *AGM* (Árvore Geradora Mínima) [Neves, 2003].

Este problema de agrupamento aparece, por exemplo, em situações nas quais a regionalização geográfica torna-se necessária, isto é, nos casos dos censos demográficos e de pesquisas socioeconômicas [Openshaw, 1977].

No contexto do censo demográfico, por exemplo, os objetos podem estar associados, a domicílios, a setores censitários, a áreas de ponderação e a municípios. Considerando o particular caso das áreas de ponderação, para a formação dos *clusters*, são considerados ainda as restrições de conexidade (contiguidade), de forma que estas áreas sejam constituídas por regiões geograficamente limítrofes, e de homogeneidade, segundo um conjunto de *p* atributos associados às características populacionais e de infra-estrutura conhecidas denominados indicadores de áreas de ponderação.

Dessa forma, ao se implementar qualquer algoritmo heurístico ou formulação para este problema, é possível associar as informações relativas à contigüidade das regiões, bem como as informações dos totais associados a cada um dos p atributos e as distâncias d_{ij} a um grafo conexo $G = (E, V)$. Essas distâncias representam o grau de homogeneidade entre os vértices (objetos) *i* e *j* em função dos valores de seus atributos associados. E cada vértice $i \in V$ do grafo corresponde a um objeto e contém os seus atributos, inclusive o atributo utilizado para a validação da restrição de capacidade.

Este trabalho, em seu estudo de caso, utiliza instâncias referentes aos problemas de Criação de Áreas de Ponderação Agregadas e Agregados de Municípios [Censo Demográfico, 2000; Silva *et al.*, 2004], obtidos a partir dos dados da amostra do Censo Demográfico de 2000 e da Contagem Populacional de 2007.

1.2 Objetivos do Trabalho

O presente trabalho, tem como objetivo, propor algoritmos heurísticos para a resolução de um problema de particionamento de grafos com restrições de capacidade e conexidade. Para isto, implementou-se no decorrer desta dissertação, um conjunto de algoritmos evolutivos e algoritmos GRASP.

Considerando estes algoritmos, foram realizados vários experimentos computacionais, utilizando um conjunto de instâncias reais obtidas do censo demográfico de 2000 e da contagem populacional de 2007. Ao se realizar tais experimentos, tem-se por objetivo avaliar a robustez, a escabilidade e a eficiência dos algoritmos.

1.3 Objetivos Específicos

Os objetivos específicos deste trabalho foram:

1) Efetuar uma revisão da literatura sobre os problemas de particionamento de grafos, em particular, os problemas onde são consideradas as restrições de capacidade e de conexidade.
2) Realizar uma breve revisão de algoritmos genéticos, evolutivos e do GRASP, bem como uma descrição detalhada das técnicas utilizadas e dos procedimentos implementados nesse trabalho.

3) Implementar os algoritmos heurísticos para resolver o problema abordado, conforme os conceitos de algoritmos evolutivos e do GRASP.
4) Comparar e avaliar o desempenho dos algoritmos heurísticos propostos, mediante a utilização de um conjunto instâncias reais.

1.4 Estrutura do Trabalho

O presente trabalho está dividido em cinco capítulos, incluindo a introdução. O capítulo 2 apresenta uma breve descrição do problema geral de agrupamento, sua definição, complexidade, requisitos desejáveis para obtenção de bons resultados e medidas para avaliar a qualidade das soluções deste problema. Este capítulo aborda também o particular problema de agrupamento considerado neste trabalho, que agrega as restrições de capacidade e conexidade.

Além disso, é feita uma revisão da literatura sobre o problema geral de agrupamento, problema de agrupamento com restrição de capacidade, conexidade, e a apresentação dos problemas utilizados nos estudos de caso, quais sejam: Criação de Áreas de Ponderação Agregadas e de Agregados de Municípios.

No Capítulo 3 são apresentados os algoritmos propostos para a resolução do problema de agrupamento abordado nesta dissertação. Tais algoritmos foram desenvolvidos a partir do estudo de Algoritmos Evolutivos e do GRASP.

O Capítulo 4 traz informações sobre as instâncias utilizadas, sobre as versões de Algoritmos Evolutivos e GRASPs propostos bem como a análise e avaliação dos resultados em relação a obtenção de soluções válidas e de soluções de boa qualidade. Por fim, o Capítulo 5 apresenta as conclusões do trabalho e sugestões para o desenvolvimento e pesquisa de trabalhos futuros.

Capítulo 2
Problema de Agrupamento

O presente capítulo traz uma descrição geral do problema de agrupamento, também conhecido como problema de clusterização, suas definições, complexidade, requisitos desejáveis para obtenção de bons resultados e medidas para avaliar a qualidade das soluções deste problema. Em seguida, define-se um particular problema de agrupamento que agrega restrições de capacidade e conexidade, com uma revisão da literatura e sugestões de técnicas para sua resolução. E complementando o capítulo, são apresentados dois estudos de caso para aplicação dos algoritmos propostos neste trabalho, quais sejam: os Problemas de Criação de Áreas de Ponderação Agregadas e de Agregados de Municípios.

2.1 Motivação

A teoria de Jean Piaget, especialista em aprendizado infantil, diz que a ação de agrupar objetos é própria do ser humano, e pode ser verificada claramente já nos seus primeiros anos de vida. Ela afirma que a mente humana é dotada de estruturas cognitivas denominadas esquemas, que organizam os eventos e determinam como estes são assimilados pelo organismo e classificados em grupos. Estes esquemas são utilizados no processamento e na identificação de estímulos [Soares, 2004].

Diante a um estímulo, o organismo tenta associá-lo a um esquema existente, podendo ser capaz tanto de diferenciá-lo quanto generalizá-lo, em relação ao conjunto de esquemas existentes. Além disso, pode-se ampliar a rede de informações registradas nestes esquemas, enriquecendo o conjunto de esquemas existentes através dos processos de assimilação e acomodação. Quanto mais rico o conjunto de esquemas existentes, maior capacidade de reconhecimento e diferenciação de objetos.

Um exemplo clássico desta teoria relacionada aos processos de assimilação e acomodação é, ao apresentar uma imagem de um cavalo a uma criança e questioná-la sobre o que é, obter como resposta tratar-se de um cachorro. Esta resposta indica que o esquema mais próximo ao estímulo apresentado (um cavalo) é o esquema que a criança possui de um

cachorro. Assim, gradativamente novos estímulos são assimilados e acomodados em sua mente.

A Figura 2.1 apresenta um conjunto com 6 elementos e pede-se para formar grupos a partir das similaridades existentes. Dado um conjunto de elementos, é possível agrupá-los de várias formas, conforme seus atributos e o critério desejado. Neste exemplo, caso o critério seja o atributo *forma* das figuras, obtêm-se três grupos: quadrado, círculo e estrela. Se o critério utilizado for o atributo *cor* obtêm-se também três grupos: preto, cinza e branco. Já utilizando o atributo tipo de *borda* obtêm-se dois grupos: contínua e tracejada.

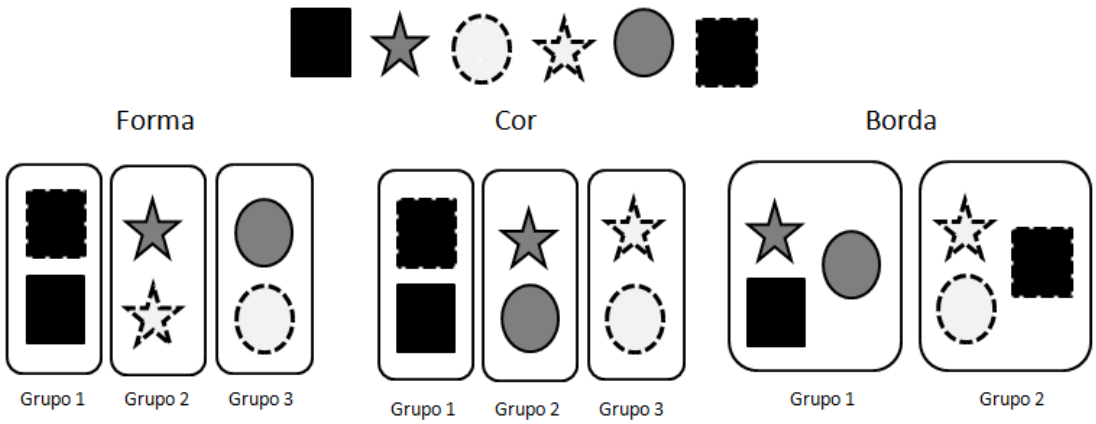


Figura 2.1: Exemplo de agrupamento de figuras.

A tarefa apresentada anteriormente é extremamente simples basicamente por duas razões: uma pequena massa de dados com apenas 6 objetos, objetos com dimensão 3 (forma, cor e borda) e o critério utilizado para a realização da *clusterização* leva em conta apenas uma das dimensões isoladamente.

Segundo [Soares, 2004; Han and Kamber, 2001] os sistemas perceptivos dos seres humanos são restritos para realização de algumas classificações e associações. É possível, por exemplo, obter grupos formados por objetos com 2 ou 3 atributos, porém, o sistema cérebro-visão é insuficiente para objetos com mais de 3 atributos.

2.2 Definição do Processo de Agrupamento

O processo de agrupar um conjunto em subconjuntos disjuntos (grupos, *clusters*) é denominado clusterização [Han and Kamber, 2001; Goldschimidt and Passos, 2005]. O objetivo é identificar *clusters* de forma a maximizar a similaridade entre os elementos de um mesmo *cluster* (*intraclusters*) e minimizar a similaridade entre elementos de *clusters* distintos (*interclusters*), conforme apresenta a Figura 2.2 [Han and Kamber, 2001; Larose, 2006; Goldschimidt and Passos, 2005].

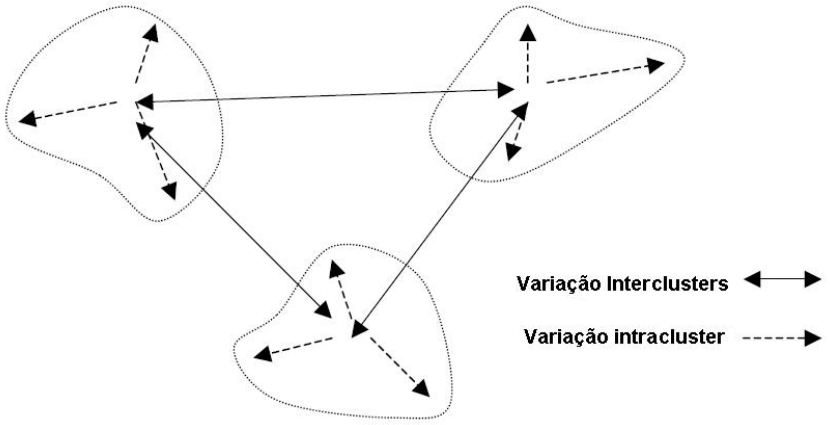


Figura 2.2: Representação de similaridades *intraclusters* e *interclusters [Larose, 2006]*.

Formalmente o problema pode ser descrito como, dado um conjunto com *n* objetos $X = \{x_1, x_2, ..., x_n\}$ no qual cada objeto x_i possui p atributos $x_i = (x_i^1, x_i^2, ..., x_i^p)$, que dimensionam as características do objeto, deve-se construir k partições C_i (*clusters*) a partir do conjunto X, respeitando as seguintes restrições [Han and Kamber, 2001; Ester *et al.*, 1995; Baum, 1986; Hruschka and Ebecken, 2001; Dias and Ochi, 2003]:

$$C_i \neq \varnothing \text{ para } i = 1,..,k$$

$$C_i \bigcap C_j = \varnothing \text{ para } i, j = 1,..,k \text{ e } i \neq j$$

$$\bigcup_{i=1}^{k} C_i = X$$

Embora grande parte das abordagens propostas para o problema de clusterização trabalhe com um número de *clusters* predefinido, em muitas aplicações reais não há o conhecimento prévio de qual é a quantidade ideal de clusters [Chiou and Lan, 2001] apud [Abramowitz, 1964]. Quando o número de *clusters* é definido previamente, o problema é conhecido como *Problema de k-clusterização* ou simplesmente por Problema de *Clusterização* (PC) [Fasulo, 1999], com a quantidade de soluções viáveis calculada a partir da seguinte Fórmula 1 [Chiou and Lan, 2001; Cole, 1998] *apud* [Liu, 1968].

$$N(n,k) = \frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n \tag{1}$$

Quando a quantidade ideal de clusters não é conhecida, ou seja, a quantidade de clusters a ser utilizada faz parte do problema, trata-se de um Problema de C*lusterização* Automática (PCA) [Doval *et al.,* 1999] e a quantidade de soluções viáveis é obtida pela Fórmula 2.

$$N(n) = \sum_{k=1}^{n}\frac{1}{k!}\sum_{i=0}^{k}(-1)^i \binom{k}{i}(k-i)^n \tag{2}$$

Tanto o PC quanto o PCA são classificados como problemas do tipo *NP-Completo*, embora o PCA seja de resolução muito mais difícil devido a um maior número de soluções viáveis.

Para ilustrar o crescimento da quantidade de soluções, [Dias, 2004] apresenta exemplos da aplicação das fórmulas 1 e 2. Considerando no PC a combinação de 10 elementos em 2 *clusters* e aplicando a fórmula 1, tem-se $N(10, 2) = 511$. Para demonstrar a dificuldade do PCA considerando apenas 10 elementos, N(10) utilizando a fórmula 2, obtêm-se 115.975 combinações diferentes, considerando que nesse caso a quantidade de *clusters* pode variar entre 1 e 10, inclusive.

Segundo [Baum, 1986; Han and Kamber, 2001; Agrawal *et al.*, 1998], a obtenção de clusters de boa qualidade, depende da observação das seguintes premissas por parte dos algoritmos de clusterização:

- Escalabilidade: deve ser escalável para trabalhar com qualquer quantidade de objetos e dimensões (atributos). A alta escalabilidade nesses algoritmos é necessária.
- Tipos de variáveis: muitas aplicações reais podem necessitar trabalhar com diversos tipos de dados, tais como: escalares em intervalos, binários, nominais (ou categóricos), ordinais, escalados em proporção, ou ainda combinações desses tipos de variáveis.
- Identificar *clusters* com formas diferenciadas: a utilização das medidas de distância Euclidiana e Manhattan, por exemplo, tendem a identificar clusters esféricos e de tamanho e densidade similares. É importante a identificação de *clusters* com formas fora dessas características. Assim, pode-se utilizar métodos baseados em densidade em que os ruídos nos dados são ignorados e é possível a obtenção de clusters com formatos arbitrários devido a distinção de regiões que possuam densidades diferentes.
- Mínimo de conhecimento na entrada dos parâmetros: Em alguns algoritmos, os resultados são bastante sensíveis aos parâmetros de entrada, tais como as configurações do algoritmo e o(s) arquivo(s) de dados. Freqüentemente, estes parâmetros são difíceis de determinar, em especial em grande volume de dados e com alta dimensionalidade, isto é, com muitos atributos.
- Habilidade para tratar ruídos: apesar da existência de dados perdidos, fora do intervalo, desconhecidos e errados em muitas bases de dados reais, estes ruídos não devem interferir na qualidade dos clusters obtidos; Alguns algoritmos são sensíveis e podem gerar soluções de baixa qualidade no que diz respeito à similaridade entre os objetos.
- Alta dimensionalidade: apesar de seres humanos serem capazes de avaliar a qualidade de soluções com até três dimensões, os algoritmos devem trabalhar com bases de dados com um variado número de atributos.
- Utilização de restrições: aplicações do mundo real podem necessitar agrupar objetos satisfazendo vários tipos de restrições. Deve ser possível encontrar *clusters* com boa qualidade e que satisfaçam as restrições existentes.
- Resultados úteis e de fácil interpretação: os resultados obtidos a partir da execução desses algoritmos devem ser de simples entendimento, compreensíveis e potencialmente úteis.

2.3 Análise da Qualidade do Agrupamento

A Análise de *clusters* tem sido amplamente utilizada em inúmeras aplicações, que podem ser mapeadas em problemas de clusterização, contribuindo nas mais diversas áreas, como: economia, epidemiologia, planejamento urbano, marketing, geografia, medicina, engenharias, bioquímica, telecomunicações entre outras [Nd and Han, 1994; Han and Kamber, 2001; Baum, 1986; Assunção *et al.*, 2006].

Conforme [Dias, 2004], outro aspecto importante a ser considerado em relação aos problemas de clusterização é como mensurar a similaridade entre objetos. Para isto é necessária a utilização de uma medida de similaridade, também conhecida como aptidão da solução, específica a cada problema de clusterização abordado. Esta medida é utilizada para verificar se dois objetos devem pertencer a um mesmo *cluster* ou devem estar separados, pertencendo a *clusters* distintos.

Um importante critério utilizado para verificar a similaridade entre dois objetos é a distância entre eles. Normalmente, tal verificação ocorre entre cada par de objetos e, quanto maior for a similaridade entre eles, menor será a distância. Algumas das medidas de distância mais utilizadas, considerando atributos escaláveis, são a distância *Euclideana*, a distância de *Manhattan*, e a distância de *Minkowski* [Han and Kamber, 2001; Cole, 1998; Dias, 2004].

A distância *Euclideana* considera a distância *d* entre dois objetos X_i e X_j no espaço *p*-dimensional, conforme apresenta a Equação 3. Os atributo pode ser ponderado por um peso específico *w* relativo a sua importância e, dessa forma, distância pode ser apresentada pela Equação 4.

$$d(X_i, X_j) = \sqrt[2]{\sum_{l=1}^{p} \left(x_{il} - x_{jl} \right)^2} \tag{3}$$

$$d(X_i, X_j) = \sqrt[2]{\sum_{l=1}^{p} w_l \left(x_{il} - x_{jl} \right)^2} \tag{4}$$

A distância *Manhattan* ou *city block*, apresentada na Equação 5, corresponde ao somatório das diferenças entre todos os p atributos de dois elementos X_i e X_j. Esta medida, porém, não é indicada para instâncias em que existe correlação entre atributos.

$$d(X_i, X_j) = \sum_{l=1}^{p} \left| x_{il} - x_{jl} \right| \quad (5)$$

Tanto a distância Euclideana quanto as distâncias de Manhattan e Minkowski satisfazem os requisitos matemáticos [Han and Kamber, 2001]:

- $d(X_i, X_j) \geq 0$: a distância sempre será um valor não negativo.
- $d(X_i, X_i) = 0$: a distância de um objeto e si mesmo é 0.
- $d(X_i, X_j) = d(X_j, X_i)$: A distância é simétrica entre os objetos.
- $d(X_i, X_j) \leq d(X_i, X_h) + d(X_h, X_j)$: a distância direta entre os objetos *i* e *j* não será maior que a distancia entre esses objetos e um objeto *h* qualquer.

Já a distância *Minkowski*, apresentada na Equação 6, é uma generalização das distâncias *Euclideana* e *Manhattan,* e necessita de valores positivos e inteiros para q. Assim como a distância *Euclideana,* as distâncias *Manhattan* e *Minkowski* também suportam a aplicação de pesos aos atributos.

$$d(X_i, X_j) = \sqrt[q]{\sum_{l=1}^{p} \left| x_{il} - x_{jl} \right|^q} \quad (6)$$

2.4 Problema de Agrupamento com Restrições de Capacidade e Conexidade

Diversos problemas de *clusterização* podem ser mapeados em um problema de particionamento de grafos em subgrafos (*clusters*) disjuntos [Doval *et al.,* 1999; Dias, 2004;

Boley, 1999; Dias and Ochi, 2003; Karypis and Kumar, 1998; Maheshwari and Shen, 1998]. Alguns desses problemas são mais específicos e possuem restrições tais como: a capacidade mínima por *cluster* [Shieh and May, 2001; Scheuerer, 2006], a conexidade [Assunção *et al.*, 2006; Assunção et al., 2002; Neves, 2003], entre outras.

O problema abordado neste trabalho de dissertação corresponde a um particular problema de particionamento de grafos, e considera tanto a restrição de conexidade quanto a restrição de capacidade [Semaan *et al.*, 2008; Semaan *et al.*, 2009]. O trabalho [Brito *et al.*, 2004] propõe uma formulação de programação inteira para a resolução deste problema.

A restrição de conexidade implica que em cada cluster, todos os vértices tomados dois a dois, devem ser vizinhos ou deve existir pelo menos um caminho entre estes dois vértices, passando apenas por vértices pertencentes ao mesmo *cluster*.

A restrição de capacidade é função de um limite inferior pré-estabelecido, com base em um dos atributos dos vértices. Assim, o somatório desse atributo nos vértices pertencentes a um mesmo *cluster* deve satisfazer um limite inferior.

Dentre as várias abordagens encontradas na literatura para a resolução do problema de clusterização com a restrição de conexidade, destacam-se o algoritmo AZP (*Automatic Zoning Procedure*) encontrado em [Assunção *et al.*, 2006] e um algoritmo baseado na utilização da *AGM* (Árvore Geradora Mínima) [Neves, 2003].

2.4.1 Abordagem utilizando *Automatic Zoning Procedure*

Segundo [Neves, 2003; Ramos, 2002] o AZP teve sua primeira versão proposta na década de 70 e utiliza a estrutura de vizinhança de objetos espaciais para garantir que a restrição de conexidade seja satisfeita. Este algoritmo atua basicamente na realocação de objetos entre os *clusters* buscando a minimização de uma função objetivo. O Algoritmo 1 apresenta um pseudo-código simplificado do algoritmo AZP. Conforme [Neves, 2003] este algoritmo deu origem a um sistema de zoneamento automático denominado *ZDES*. Além disso, [Alvanides *et al.*, 2002] propôs melhorias objetivando o aumento da qualidade das soluções através da redução da função objetivo.

Passo 1: Gerar uma partição aleatória dos n objetos em k regiões.

Passo 2: Construir uma lista de k regiões

Passo 3: Selecionar aleatoriamente uma região k_i da lista de regiões e retirá-la da lista de regiões.

Passo 4: Identificar uma lista de objetos vizinhos à região k_i que podem ser realocados em k_i, sem destruir a contiguidade interna da região doadora.

Passo 5: Retirar aleatoriamente um objeto da lista de objetos vizinhos; se ocorrer melhoria na função objetivo, incluir o objeto em k_i e retornar ao Passo 4. Senão repetir o passo 5 até exaurir a lista de objetos.

Passo 6: Se a lista de regiões não estiver vazia, retornar ao passo 3 para selecionar outra região e repetir os passos de 4 a 6.

Passo 7: Repetir os passos de 2 a 6 até não haver mais realocações que provovam melhoria.

Algoritmo 1: Algoritmo AZP (*Automatic Zoning Procedure*) [Neves, 2003].

2.4.2 Abordagem baseada na construção da Árvore Geradora Mínima

A abordagem apresentada em [Assunção *et al.*, 2006] é constituída por duas etapas, em que a primeira é responsável pela construção da AGM do grafo associado ao problema. Já a segunda etapa consiste na obtenção de *clusters* a partir do particionamento da AGM, através da remoção de um subconjunto de arestas.

Em particular, no presente trabalho, optou-se pela abordagem baseada na construção da AGM. Os objetos correspondem aos vértices de um grafo e o algoritmo proposto particiona-o sucessivamente, conforme a quantidade de *clusters* especificada. A vantagem dessa abordagem é o fato da adjacência espacial (contigüidade) exigida pelo problema ser atendida naturalmente, tendo em vista que uma árvore é um grafo conexo e qualquer aresta removida da árvore produz duas subárvores também conexas.

Considerando os atributos de cada vértice, são calculadas todas as distâncias d_{ij} entre vértices *i* e *j* adjacentes, utilizando a Equação 7. Estas distâncias representam o grau de dissimilaridade entre dois vértices, em função dos valores de seus *p* atributos associados.

$$d(X_i, X_j) = \sqrt[2]{\sum_{l=1}^{p} \left(x_{il} - x_{jl} \right)^2} \quad (7)$$

2.4.2.1 Problema da Construção da Arvore Geradora Mínima

Segundo [Ferreira et al., 2007] o problema da construção da AGM é um dos mais importantes da otimização combinatória e que consiste em: a partir de um grafo $G = (E, V)$ para o qual cada aresta $e \in E$ possui um custo $c_e \in \mathbb{R}^+$, obter a árvore formada por |V|-1 arestas de E, de forma a minimizar a Equação 8, em que o custo C_e de cada aresta que compõe a árvore é obtido através da Equação 7.

Os algoritmos mais conhecidos da literatura são os de *Kruskal* [Cormen, 2002; Gersting, 2004] e de *Prim* [Cormen, 2002; Gersting, 2004], que além de possuírem uma complexidade polinomial para o pior caso, são utilizados para auxiliar na resolução de vários outros problemas de otimização mais complexos. Neste trabalho foi utilizado o algoritmo de *Kruskal.*

$$min \sum_{e \in T} C_e \quad (8)$$

A primeira etapa desse algoritmo consiste em ordenar, de forma crescente, as arestas do grafo submetido conforme seus custos. Em seguida, a árvore T é iniciada vazia. Conforme a ordenação das arestas, a cada iteração a inserção de uma aresta em T é avaliada, uma vez que não pode ocorrer ciclos em T. As inserções ocorrem até que a árvore T tenha |V|-1 arestas, considerando |V| a quantidade de vértices do grafo submetido ao algoritmo. Ao final da execução, o algoritmo retornará uma árvore conexa T com menor custo conforme a Equação 8.

2.4.2.2 Particionamento da Árvore Geradora Mínima

A segunda etapa consiste em particionar a AGM mediante a remoção de k-1 arestas dentre as |V|-1 arestas disponíveis, para a obtenção de k subgrafos conexos (*clusters*). Em [Neves, 2003] é observado que, com objetivo de obter melhores soluções, ou seja, soluções mais homogêneas, é necessário utilizar formas distintas de cálculo para o custo dos clusters

associados aos k subgrafos. Enquanto na etapa de construção, o custo das arestas do grafo submetido ao algoritmo é calculado através da Equação 7, durante a etapa de particionamento do grafo, utiliza-se a Equação 9 para a obtenção de seu valor.

O processo de particionamento consiste em, selecionado um *cluster* T_i para divisão, buscar a aresta existente entre vértices desse *cluster* que possuir maior $Custo_e$ e removê-la da árvore. Para isto, deve-se remover cada uma das arestas de T_i e aplicar a Equação 6 em cada um dos novos clusters obtidos T_i^1 *e* T_i^2. Em seguida, com base nos valores da função obtidos em cada um dos novos clusters, deve-se avaliar, através da Equação 9, o custo de remoção de cada aresta. O valor do custo de remoção da aresta corresponde a diferença entre a função objetivo do cluster atual e a soma da função objetivo de cada um dos novos clusters. Quanto mais homogêneos forem os dois novos *clusters*, maior será o valor de $Custo_e$.

A Equação 10 representa a soma dos quadrados dos desvios no espaço dos atributos em relação à média de todos os atributos dos vértices em um determinado cluster.

$$Custo_e = f(T_i) - (f(T_i^1) + f(T_i^2)) \tag{9}$$

$$f = \sum_{j=1}^{m}\sum_{i=1}^{n}(x_{ij} - \overline{x_j})^2 \quad \text{em que} \quad \overline{x_j} = \frac{\sum_{i=1}^{n} x_{ij}}{n} \quad \text{para } m \text{ atributos, } n \text{ objetos.} \tag{10}$$

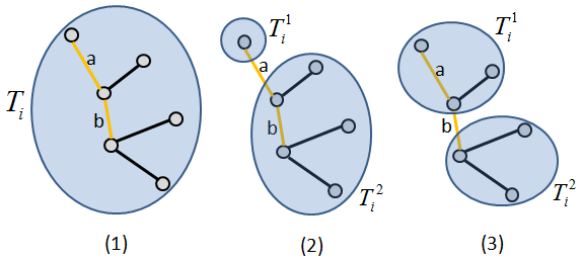


Figura 2.3: Exemplo de cálculo do *Custo*$_e$.

Para ilustrar a utilização da equação 9, a Figura 2.3 apresenta a divisão de um cluster formando duas soluções distintas, considerando a remoção das arestas *a* e *b*. A figura 2.3 (1) apresenta a solução formada com um *cluster* associado à subárvore T_i, que possui o valor de aptidão 10, obtido pela utilização da equação 10. A figura 2.3 (2) apresenta uma solução em que, a remoção da aresta *a* formou 2 *clusters*, associados as subárvores T_i^1 e T_i^2, com

aptidões 5 e 4, respectivamente. Dessa forma, utilizando-se a equação 9 obtém-se valor 1 para essa aresta. Já a figura 2.3 (3) as subárvores T_i^1 e T_i^2 possuem respectivamente aptidões 5 e 2, e a aresta *b* possui valor 3. Uma vez que deve-se maximizar o custo da aresta removida, a figura 2.3 (3) apresenta a solução mais interessante desse exemplo, com aptidão 7 enquanto a solução da figura 2.3 (2) possui aptidão 9.

Em [Assunção *et al.*, 2006] foi proposto um algoritmo denominado SKATER (*Spatial 'K'luster Analysis by Tree Edge Removal*). Este algoritmo utiliza a técnica baseada em AGM e trabalha com três critérios possíveis para obtenção de soluções, quais sejam:

- Através de uma quantidade de *clusters que*, no contexto da ferramenta, são denominados conglomerados;
- Pela capacidade de cada conglomerado, independente da quantidade de conglomerados construídos;
- Por um valor máximo de n conglomerados considerando uma capacidade mínima Cap informada.

Considerando os critérios existentes no SKATER é possível afirmar que o algoritmo não atende plenamente a demanda do problema de clusterização com restrições de contigüidade (conexidade) e capacidade, uma vez que não é possível determinar e garantir simultaneamente, tanto a quantidade de *clusters* quanto a capacidade mínima de cada *cluster*. Embora tal algoritmo não garanta a geração de uma solução cuja a restrição de capacidade seja satisfeita, a utilização de penalidades para soluções inválidas pode ser interessante tanto para auxiliar nos ajustes de configurações do algoritmo, quanto para obter soluções interessantes, mesmo quando a restrição de capacidade não seja atendida. Desta forma é possível a obtenção de soluções que não atendam esta restrição, porém, que satisfaça a restrição de contigüidade e que a quantidade de *clusters* pré-estabelecida também seja respeitada.

2.4.2.3 Formulação de Programação Inteira pela Abordagem do Particionamento da Árvore Geradora Mínima

Para a resolução do problema abordado nessa dissertação, a formulação matemática apresentada no trabalho [Brito et al., 2004] é descrita a seguir. Nessa formulação são consideradas as seguintes notações e restrições:

V = Quantidade de vértices.

K = Quantidade de *clusters*.

P_j = População associada a cada vértice j.

E*= Arestas da Árvore Geradora Mínima.

X_i = Variável Inteira que determina a quantidade de vértices que estão no i-ésimo *cluster*.

Y_{ij} = Variável Binária que assume valor 1 se o j-ésimo vértice está associado ao i-ésimo *cluster*.

e^i_{mn} = Variável Binária: assume valor 1 se a aresta $(m,n) \in E^*$ está ativa no i-ésimo *cluster*.

t^i_{mn} = Variável Binária: assume valor 1 se a aresta $(m,n) \in E^*$ está ativa no i-ésimo *cluster*.

D_{mn} = Medida que representa o grau de homogeneidade entre dois clusters vizinhos.

Função Objetivo: Está associada ao critério de soma mínima. Ou seja, o valor desta função representa a soma do melhor subconjunto de arestas ativas associadas aos valores D_{mn}.

Restrições 1 e 2: A quantidade de vértices associados a cada um dos clusters deve ser igual ao total de vértices da instância.

Restrição 3: Garante que um vértice estará associado a exatamente um cluster.

Restrição 4: A soma do atributo referente à capacidade associada a cada um dos vértices, em cada um dos *clusters,* deve ser maior ou igual a capacidade mínima pré-estabelecida.

Restrição 5: Garante que cada uma das arestas associadas à árvore geradora que contém todos os vértices só poderá estar em no máximo um *cluster*. Ou seja, se dois vértices são vizinhos e estão em um mesmo cluster, então e^i_{mn} = 1.

Restrição 6: A quantidade de arestas associadas a cada um dos *clusters* deve ser igual a quantidade de vértices em cada um dos *clusters* menos um.

Restrições 7 a 10: Estas restrições garantem que se uma aresta do conjunto *E** estiver ativa, ou seja, pertencer a uma subárvore associada a um cluster, então os vértices *m* e *n* desta aresta também farão parte do cluster.

$$\text{Minimizar} \sum_{i=1}^{K} \sum_{\forall (m,n) \in E^*} e_{mn}^{i} . D_{mn}$$

$$(1)\ \sum_{i=1}^{K} X_i = V$$

$$(2)\ \sum_{j=1}^{V} Y_{ij} = X_i, \quad i = 1,..,K$$

$$(3)\ \sum_{i=1}^{K} Y_{ij} = 1, \quad j = 1,..,V$$

$$(4)\ \sum_{j=1}^{V} Y_{ij} . P_j \geq TotPop, \quad i = 1,..,K$$

$$(5)\ \sum_{j=1}^{K} e_{mn}^{i} \leq 1, \quad \forall (m,n) \in E^*$$

$$(6)\ \sum_{\forall (m,n) \in E^*} e_{mn}^{i} = X_i - 1, \quad i = 1,..,K$$

$$(7)\ t_{mn}^{i} \leq Y_{im} \quad i = 1,..,K, \quad \forall (m,n) \in E^*$$

$$(8)\ t_{mn}^{i} \leq Y_{in} \quad i = 1,..,K, \quad \forall (m,n) \in E^*$$

$$(9)\ Y_{im} + Y_{in} - 1 \leq t_{mn}^{i} \quad i = 1,..,K, \quad \forall (m,n) \in E^*$$

$$(10)\ e_{mn}^{i} \leq t_{mn}^{i} \quad i = 1,..,K, \quad \forall (m,n) \in E^*$$

$$X_i \in Z_+, \quad Y_{ij} \in (0,1), \quad t_{mn}^{i} \in (0,1)$$

2.5 Problemas Abordados

Entre os casos em que se utiliza o conceito de regionalização geográfica, podem-se citar os censos demográficos e pesquisas socioeconômicas [Openshaw, 1977]. No caso do censo demográfico, os objetos podem estar associados, por exemplo, a domicílios, setores censitários, áreas de ponderação (APOND) e a municípios. Este trabalho, em seu estudo de

caso, utiliza instâncias referentes aos problemas de Criação de Áreas de Ponderação Agregadas e Agregados de Municípios [Censo Demográfico, 2000; Silva et al., 2004].

Para a formação dos conglomerados (*clusters*), são considerados ainda restrições de conexidade, de forma que estas áreas sejam constituídas por regiões geograficamente limítrofes, e de homogeneidade, segundo um conjunto de *p* atributos associados às características populacionais e de infra-estrutura conhecidas. Estas variáveis, que serão representadas por x_s, $s = \{1,..,p\}$, são denominadas indicadores de áreas.

Desta forma, para o desenvolvimento de qualquer algoritmo heurístico ou formulação para este problema, é possível associar as informações relativas à contigüidade das regiões, bem como as informações dos totais associados a cada um dos p atributos e as distâncias d_{ij}, a um grafo $G = (E, V)$. Cada vértice $i \in V$ do grafo corresponde a uma região e seus atributos, inclusive o atributo capacidade, utilizado para validação da restrição de capacidade.

Além disso, se duas regiões i e j são vizinhas, existe uma aresta $e=(i,j)$ cujo valor da distância é d_{ij}, obtido através da aplicação da Equação 7. Os conglomerados serão representados pelos *clusters*.

A Figura 2.4 ilustra o grafo que representa o(s) *cluster*(s) em três estágios: (1) Grafo contendo a vizinhança entre subregiões; (2) Arvore Geradora Mínima construída a partir do grafo original; (3) Árvore após o particionamento da AGM com a formação de *clusters*, representados utilizando cores distintas.

Considerando o Problema de Criação de Áreas de Ponderação Agregadas, uma APOND é definida como uma unidade geográfica formada por agrupamentos mutuamente exclusivos de setores censitários, sendo estes, por sua vez, formados por conjuntos de domicílios. Esta unidade é utilizada para se estimar informações para a população e, também, os níveis geográficos mais detalhados da base operacional, sendo desenvolvidos como forma de atender às demandas por informações em níveis geográficos menores que os municípios [Censo Demográfico, 2000; Silva *et al.*, 2004]. As Áreas de Ponderação Agregadas (APAs) são formadas por agrupamentos mutuamente exclusivos de APONDs.

De forma análoga ao Problema de Criação de Áreas de Ponderação Agregadas, serão utilizadas instâncias referentes a municípios objetivando a formação de conglomerados de municípios, novamente considerando as restrições de capacidade e contigüidade. O Capítulo 4 apresenta informações sobre as instâncias utilizadas, bem como sobre os experimentos realizados.

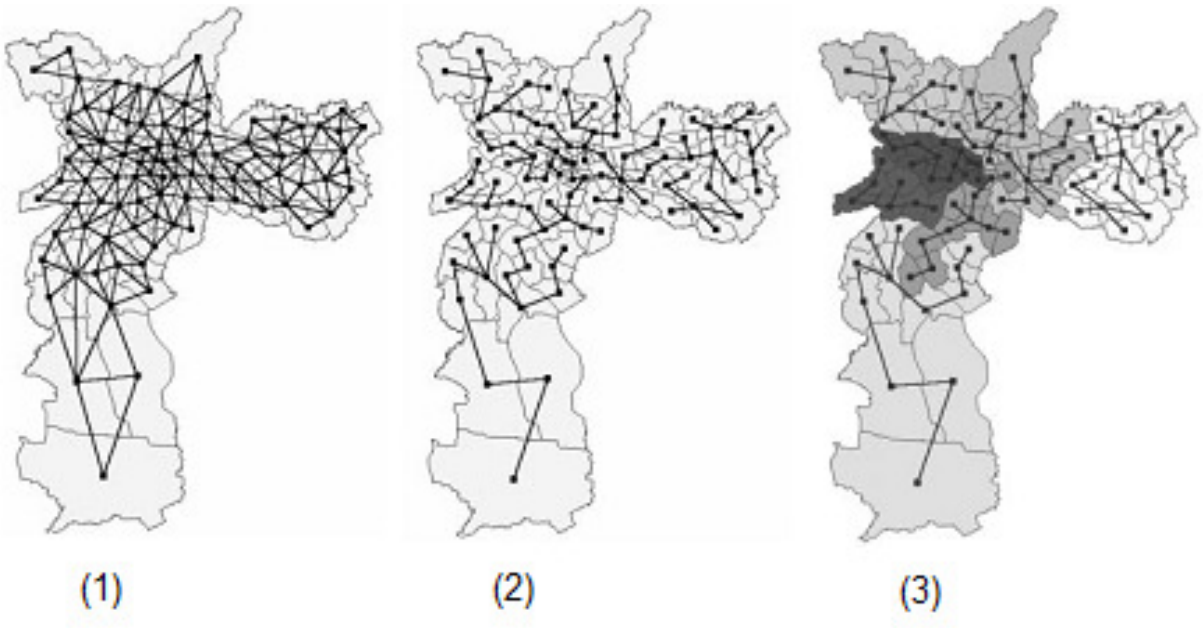


Figura 2.4: Representação de subregiões e clusters através de grafos [Assunção *et al.*, 2006].

Capítulo 3
Algoritmos

O presente capítulo traz uma descrição dos algoritmos heurísticos propostos para a resolução do problema de clusterização abordado nesta dissertação. Tais algoritmos foram desenvolvidos a partir do estudo de Algoritmos Genéticos, Algoritmos Evolutivos e do GRASP (*Greedy Randomized Adaptative Search Procedure*).

3.1 Algoritmos Genéticos

Os algoritmos genéticos são um ramo dos algoritmos evolutivos e foram introduzidos por [Holland, 1975]. Segundo [Dias, 2004; Linden, 2006], são técnicas heurísticas de otimização global que podem ser aplicadas para a solução de diversos problemas reais. Utilizam modelos computacionais baseados na teoria proposta por Charles Darwin sobre a evolução das espécies e nos princípios básicos da herança genética propostos por Gregor Mendel. Após [Holland, 1975], vários pesquisadores (vide [Goldberg, 1989; Beasle *et al.*, 1993]) tiveram importância fundamental para consolidação dos princípios desses algoritmos.

Termos como cromossomos e gens, comuns no estudo da genética, têm correspondentes no modelo computacional proposto para simular processos evolutivos. Esses algoritmos também buscam trabalhar de forma similar à teoria biológica da evolução das espécies, em que os cromossomos mais aptos sobrevivem e geram descendentes com características a eles semelhantes, devido à hereditariedade [Linden, 2006; Beasle *et al.*, 1993].

Esses algoritmos trabalham com uma população de indivíduos, ou seja, cromossomos que correspondem a um conjunto de soluções viáveis de um determinado problema. Cada indivíduo é avaliado conforme uma função objetivo, que determina o valor de sua aptidão. Em um problema de minimização, o indivíduo mais apto é o que possui o menor valor de aptidão, enquanto em um problema de maximização o mais apto possui o maior valor de aptidão. Através da avaliação de suas aptidões, as melhores soluções da população são selecionadas

para aplicação dos operadores genéticos de cruzamento e mutação. A seção 3.3.4 apresenta duas formas de seleção de indivíduos, quais sejam: método do torneio e o método da roleta.

O cruzamento de soluções realiza trocas de partes dos cromossomos, objetivando a obtenção de novas soluções de melhor qualidade. A freqüência de execução deste procedimento também está condicionada a uma probabilidade em que um valor real gerado aleatoriamente deve ser inferior à probabilidade submetida como parâmetro ao algoritmo.

Embora existam várias versões de cruzamentos entre soluções, a versão utilizada nesse trabalho é a de um ponto [Linden, 2006; Beasle *et al.*, 1993]. Nessa versão, são selecionados dois cromossomos e uma posição inicial (ponto de corte) dos cromossomos, a partir da qual, ocorre a troca de gens entre as soluções, formando duas novas soluções conforme ilustra a Figura 3.1. Já Figura 3.2 apresenta um exemplo do cruzamento de dois pontos, no qual duas posições dos cromossomos são aleatoriamente selecionadas e ocorre a troca de gens entre essas posições. Em ambos os exemplos as soluções S1 e S2 são substituídas pelas soluções S1” e S2”.

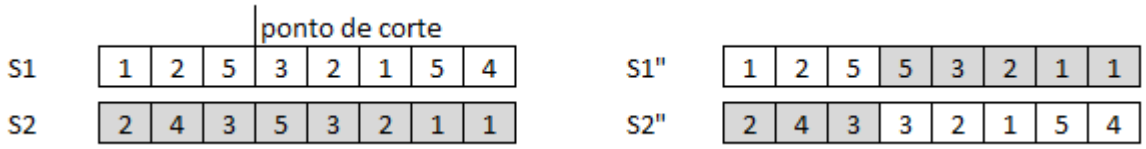


Figura 3.1: Exemplo de cruzamento de um ponto

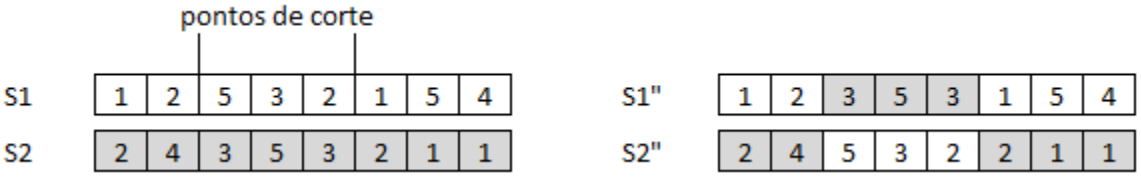


Figura 3.2: Exemplo de cruzamento de dois pontos

O procedimento de mutação atua como fator de perturbação, evitando que a solução fique estagnada em ótimos locais e regenerando possíveis soluções que tenham sido eliminadas pelos outros procedimentos. A execução deste procedimento, assim como para o procedimento de cruzamento, está condicionada a uma probabilidade. Neste caso o procedimento realiza trocas de gens entre soluções de forma aleatória, conforme apresenta a Figura 3.3.

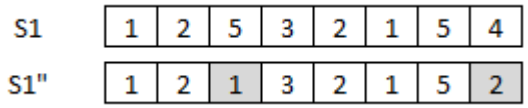


Figura 3.3: Exemplo de mutação

A partir da utilização desses operadores é possível formar uma nova população composta por algumas soluções de melhor qualidade, quando comparadas com as soluções advindas das gerações anteriores. Em seguida, já com a nova população, o algoritmo inicia uma nova iteração, também conhecida como geração. O Algoritmo 2 apresenta um pseudocódigo simplificado do algoritmo genético tradicional.

```
i = 0
Iniciar a população P_0
Avaliar as soluções de P_0
Enquanto (Critério de Parada não satisfeito)
   i = i + 1
   Selecionar soluções em P_(i-1) para P_i
   Aplicar os operadores genéticos de cruzamento e mutação nas soluções de P_i
   Avaliar as soluções de P_i
Fim-Enquanto
```

Algoritmo 2: Algoritmo genético tradicional.

Segundo [Trindade and Ochi, 2006], apesar da comprovada eficiência dos Algoritmos Genéticos para diversos problemas de otimização, sua utilização em conjunto com técnicas de busca local ou mesmo com outras metaheurísticas como *Tabu Search* [Glover and Kochenberger, 2002; Glover, 1986], ILS (*Iterated Local Search*) [Baum, 1986; Glover and Kochenberger, 2002; Baxter, 1981] e VNS (*Variable Neighborhood Search*) [Mladenovic, 1995; Glover and Kochenberger, 2002; Mladenovic and Hansen, 1997] pode aumentar consideravelmente seu desempenho no que concerne à qualidade das soluções obtidas.

Assim, especificamente no caso desse trabalho, buscou-se a implementação de algoritmos que combinassem as idéias básicas de um algoritmo genético e procedimentos de busca local e de construção que pudessem colaborar para a produção de resultados de boa qualidade, tornando os algoritmos propostos ainda mais especializados para o problema submetido.

Embora no algoritmo genético tradicional a construção de soluções ocorra de forma aleatória, este trabalho apresenta três versões de procedimentos construtores para a minimização da função aptidão e formação de soluções válidas, que atendam as restrições existentes no problema abordado, além dos seis procedimentos de busca local. Tanto os procedimentos de construção quanto os de busca local são utilizados nos algoritmos genéticos e nos algoritmos GRASP desse trabalho. A seguir são apresentados os procedimentos de construção e de busca local:

- **Procedimento para construção de soluções:** A resolução do problema abordado nesse trabalho não se restringe a simples minimização de uma função objetivo, mas também leva em conta o atendimento das restrições de conexidade e de capacidade. Assim, em todos os procedimentos implementados a restrição de conexidade é garantida. Porém, apesar dos procedimentos atuarem na busca por soluções válidas, que atendam também a restrição de capacidade mínima por cluster, esta restrição pode não ser atendida e, em algumas situações, pode ser impossível a formação de soluções que atendam ambas as restrições. Os procedimentos construtores utilizam de heurísticas que realizam o particionamento da árvore geradora mínima (AGM) construída a partir do grafo associado ao problema. A seção 3.3.2 apresenta em detalhes os procedimentos que foram implementados nesse trabalho para construção de soluções iniciais.
- **Procedimentos de Busca Local:** foram desenvolvidos seis procedimentos de busca local, nos quais três atuam com base somente na AGM e os três demais utilizam também o grafo original, submetido ao algoritmo antes da construção da AGM. Basicamente, esses procedimentos realizam migrações de vértices entre *clusters*, objetivando a minimização da aptidão das soluções ou mesmo a tentativa de eliminação de penalidades ocasionadas pelo não cumprimento da restrição de capacidade mínima por *cluster*. A seção 3.3.3 apresenta detalhadamente os procedimentos de busca local que foram implementados nesse trabalho. Tais procedimentos foram aplicados para melhorar a qualidade das soluções obtidas através da minimização de suas aptidões e para regenerar soluções penalizadas.

3.2 GRASP

O GRASP (*Greed Randomized Adaptive Search Procedure*) é uma metaheurística proposta por [Feo and Resende, 1995], na qual cada iteração consiste em duas fases: construção de uma solução inicial e aplicação de uma busca local sobre esta solução, com finalidade de melhorá-la.

Na fase de construção, uma solução é iterativamente construída, elemento por elemento. Observando que em cada iteração dessa fase, utiliza-se uma lista de candidatos *(LC)* formada por todos os elementos que podem ser incorporados na solução parcial x_0 e que não provocam a inviabilidade do problema.

Definida a *LC,* deve-se avaliar todos os seus elementos através de uma função gulosa $g(.)$, que representa o custo de se adicionar um novo elemento $t \in LC$ na solução parcial x_0.

Com o objetivo de possibilitar uma maior variabilidade nas soluções obtidas, pode-se definir uma lista restrita de candidatos (LRC), formada pelos melhores elementos avaliados na *LC* através da função g, quais sejam, aqueles que, quando incorporados à solução parcial x_0, produzem um acréscimo mínimo (caso de minimização). No trabalho de [Feo and Resende, 1995] são propostos dois esquemas para a construção da LRC: (i) um inteiro k é fixado e os k melhores candidatos, ordenados na LC segundo algum critério, são selecionados para compor a LCR e (ii) em cada iteração da fase de construção, denota-se respectivamente por $\underline{g}$ e $\overline{g}$ o menor e maior acréscimo provocados pela inserção de um elemento $t \in LC$ na solução, segundo a função gulosa $g(.)$. A partir da utilização desta função e dos valores $\underline{g}$ e $\overline{g}$, define-se:

$$LRC = \{t \in LC \mid g(t) \leq \underline{g} + \alpha(\overline{g} - \underline{g})\}$$, onde $\alpha \in [0,1]$.

Quando $\alpha = 0$, o procedimento de construção torna-se *guloso*, pois a LCR tem apenas um elemento, e quando $\alpha = 1$, é produzida uma solução aleatória, tendo em vista que a LRC terá todos os elementos da LC.

Na fase de busca local, há uma tentativa de melhoria da solução inicial x_0 obtida na primeira fase, tendo em vista que esta solução não é necessariamente uma solução ótima para o problema. A busca local consiste na substituição da solução x_0 pela melhor solução $x_0^{'}$ encontrada em uma vizinhança de x_0. A solução obtida a partir do GRASP será a melhor das soluções obtidas em todas as iterações. Algoritmo 3 apresenta um pseudocódigo do algoritmo GRASP.

```
Aptidao_solucao_melhor. = ∞
i=0
Enquanto (i < Total_Iteracoes)
   i = i + 1;
   solucao = ConstruirSolucao(α)
   solucao = BuscaLocal(solucao)
   Se (Aptidao_solucao < Aptidao_solucao_melhor)
      solucao_melhor = solucao
   Fim-se
Fim-Enquanto
```

Algoritmo 3: Algoritmo GRASP (*Greed Randomized Adaptive Search Procedure*).

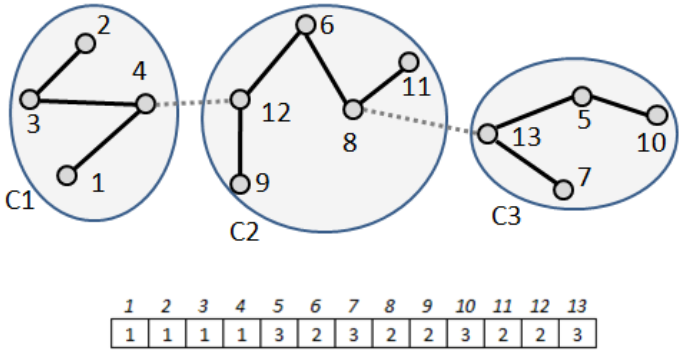
3.3 Heurísticas e Procedimentos Implementados

3.3.1 Representação de uma Solução

Segundo [Hartuv and Shamir, 1999] uma boa representação para o problema é extremamente importante, tendo impacto direto na performance do algoritmo em relação ao tempo para obtenção de soluções de boa qualidade.

Conforme a descrição encontrada em [Trindade and Ochi, 2006; Dias, 2004], a estrutura de representação adotada neste trabalho foi a *group-number*. Nessa estrutura, cada índice do vetor representa o vértice do grafo e o seu conteúdo, um número inteiro positivo entre 1 e k, representa o *cluster* ao qual o vértice pertence. A solução apresentada pela Figura 3.4 indica

que o grafo abaixo foi dividido em 3 *clusters*, em que o *cluster* C_1 possui os vértices 1,2,3 e 4, o *cluster* C_2 possui os vértices 6,8,9,11 e 12 e o *cluster* C_3 possui os vértices 5,7,10 e 13.



1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	1	1	3	2	3	2	2	3	2	2	3

Figura 3.4: Representação de uma solução.

Conforme apresentado no Capítulo 2, a capacidade mínima por *cluster*, associada ao *i-ésimo* atributo dos objetos, é uma restrição do problema abordado. Os algoritmos implementados nesta dissertação permitem que este valor seja informado como parâmetro de entrada ou que o próprio algoritmo calcule o valor da capacidade através da utilização da Equação 11. Nessa equação, |V| corresponde a quantidade de objetos a serem agrupados, x_{il} é o valor do *l*-ésimo atributo associado ao *i*-ésimo objeto, *k* é quantidade de clusters e β é um fator de ajuste.

$$Capacidade_{Min} = (\beta / k).\sum_{i=1}^{|V|} x_{il} \qquad (11)$$

Além deste vetor, a estrutura de dados utilizada para representar as soluções também possui um atributo referente a capacidade do *cluster*, obtida através do somatório do atributo associado a capacidade dos vértices do *cluster*. Assim é possível verificar quais *clusters* estão penalizados, ou seja, que possuem um valor inferior ao limite inferior pré-estabelecido e, conseqüentemente, quantas penalidades a solução possui.

3.3.2 Procedimentos para Construção de Soluções Iniciais

Os procedimentos de construção desenvolvidos neste trabalho geram os *clusters* através do particionamento de uma AGM T, obtida a partir do grafo G associado ao problema, em k subárvores (*clusters*). Ou seja, os k *clusters* são definidos considerando a remoção de exatamente $k\text{-}1$ arestas de T, sendo que a remoção de cada aresta produz duas novas subárvores que corresponderão a dois novos *clusters* definidos por T_i^1 e T_i^2.

Assim, utilizando a idéia de particionamento da AGM, foram implementadas três versões de procedimentos construtores de soluções iniciais. A primeira versão realiza o particionamento respeitando a restrição de conexidade, porém não considerando a restrição de capacidade. Ou seja, objetivando apenas a minimização da função aptidão das soluções. Já as duas outras versões, trabalham na formação de soluções que atendam as duas restrições do problema abordado neste trabalho.

3.3.2.1 Procedimento Construtor 1

Este procedimento consiste basicamente em duas etapas, executadas k – 1 vezes para a obtenção de k *clusters*:

- **Seleção do cluster a ser particionado**: o *cluster* que possuir o maior valor considerando a Equação 12 é selecionado para ser particionado, exceto na primeira iteração do procedimento, em que existe apenas um cluster (inicial) composto pela AGM. A equação utilizada representa o somatório do desvio quadrático em relação a um conjunto de atributos, isto é, as variáveis consideradas no problema.
- **Definição da aresta a ser removida:** após a seleção do *cluster* a ser particionado é necessário definir a aresta a ser removida, de forma a produzir dois novos clusters. Para isto, [Neves, 2003] utilizou-se um procedimento guloso, em que todas as possibilidades de remoção de arestas do *cluster* selecionado são executadas, objetivando a minimização da função objetivo f. O valor do custo de remoção da aresta é obtido através da Equação 13. Quanto maior o valor de $Custo_e$, mais homogêneos são os dois novos clusters obtidos, mediante a remoção da aresta.

$$f = \sum_{j=1}^{p} \sum_{i=1}^{n} (x_{ij} - \overline{x_j})^2 \quad \text{em que} \quad \overline{x_j} = \frac{\sum_{i=1}^{n} x_{ij}}{n} \quad \text{para } p \text{ atributos, } n \text{ objetos.} \tag{12}$$

$$Custo_e = f(T_k) - (f(T_k^1) + f(T_k^2)) \tag{13}$$

Com o objetivo de possibilitar a diversificação de soluções, este procedimento foi adaptado para não possuir comportamento guloso. Dessa forma, o algoritmo é capaz de obter soluções potencialmente interessantes, mas que seriam ignoradas pelo procedimento guloso. Para isso, utilizando-se a idéia do procedimento de construção do GRASP, em vez de selecionar sempre a aresta de maior custo, optou-se pela construção de uma lista restrita de candidatos (LRC). Para esse problema, a LRC será formada pelas *α* arestas que, se removidas, possibilitam a geração das *α* melhores soluções, que possuem menor valor da função objetivo f. Uma das arestas desta LRC é aleatoriamente selecionada e removida do cluster correspondente, formando dois novos clusters. Então, caso $\alpha=1$ o algoritmo torna-se guloso. Os passos para o Procedimento de Construtor 1 são apresentados no Algoritmo 4.

Inicializar grafo G *=T_0 considerando $T_{0 = AGM}$.
Calcular *Custo$_e$* para todas as arestas em T_0.
Formar a LRC com as α arestas que possuem os maiores valores *Custo$_e$*.
Particionar a AGM através da remoção aleatória de uma aresta da LRC.
Enquanto a quantidade de clusters pré-estabelecida não for alcançada
  Selecionar o cluster com maior valor da função objetivo *f*.
  Calcular o *Custo$_e$* para todas as arestas do cluster selecionado.
  Formar a LRC com as α arestas que possuem os maiores valores *Custo$_e$*.
  Particionar o cluster selecionado através da remoção aleatória de uma aresta da LRC.
Fim-Enquanto

Algoritmo 4: Algoritmo do procedimento construtor 1.

A Figura 3.5 apresenta a execução do procedimento de construtor 1 para a obtenção de 3 *clusters* passo a passo, considerando os comentários a seguir:

- **Passo 1:** todos os objetos de T_0 inicialmente pertencem ao *cluster* C_1. Esse *cluster* é selecionado para o particionamento.
- **Passo 2:** considerando α = 3, as três arestas do *cluster* selecionado que possuírem maior valor segundo a Equação 13, devem compor a lista restrita de candidatos. Essas arestas estão em destaque na Figura 3.5 referente a esse passo.
- **Passo 3:** uma das arestas da lista restrita de candidatos é aleatoriamente removida e, com isso, são formados dois novos *clusters*, C_1 e C_2.
- **Passo 4:** o *cluster* que possuir maior valor conforme a Equação 12 é selecionado para o particionamento.
- **Passo 5:** novamente as três arestas em destaque do *cluster* selecionado, que possibilitam a geração das melhores soluções, compõem a lista restrita de candidatos.
- **Passo 6:** uma das arestas da lista restrita de candidatos é removida aleatoriamente, formado dois novos *clusters*, C_2 e C_3.

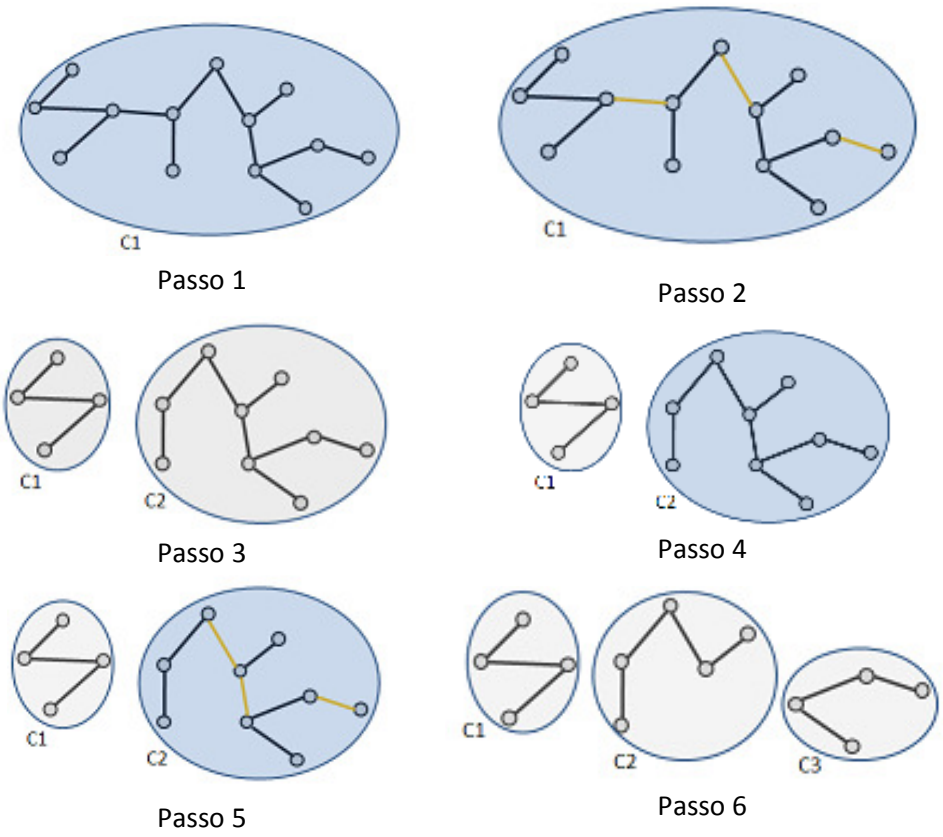


Figura 3.5: Exemplo de aplicação do procedimento construtor 1.

Embora exista a restrição de capacidade mínima, ou seja, um limite inferior para o somatório do atributo associado à capacidade dos vértices do cluster, a primeira versão do procedimento construtivo apenas indica os clusters que não satisfazem esta restrição, isto é, que estão penalizados. Os procedimentos de busca local são utilizados para a realização de possíveis migrações de vértices entre os *clusters,* de forma a regenerar soluções penalizadas e para reduzir a aptidão das soluções, melhorando sua qualidade.

3.3.2.2 Procedimento Construtor 2

A segunda versão de procedimento construtor trabalha na tentativa de obtenção de *clusters* que atendam a restrição de capacidade mínima. Para isso, esse procedimento identifica uma subárvore principal contida na AGM e particiona-a sucessivamente até a obtenção dos K clusters.

Para definição da subárvore principal devem ser selecionados, de forma aleatória, dois vértices A e B da AGM. Tal subárvore será formadas pelos vértices A e B e pelos os vértices que encontram-se no caminho existente na AGM entre esses vértices. A Figura 3.6 apresenta um grafo e dois vértices A e B em destaque, selecionados aleatoriamente.

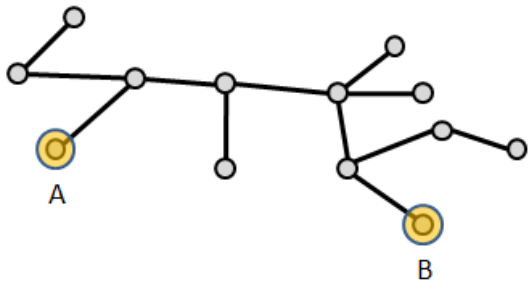


Figura 3.6: Seleção de dois vértices para o procedimento construtor 2.

Para a identificação dos vértices que encontram-se no caminho existente entre os vértices A e B, utilizou-se a Equação 14. A distância entre os vértices A e B conforme o caminho existente na *AGM*, é a quantidade de arestas existentes entre esses dois vértices. A Figura 3.7 ilustra a subárvore principal formada pelos vértices A e B e os vértices que estão no caminho entre A e B. Em seguida, é apresentado um exemplo da aplicação dessa equação para identificação dos vértices pertencentes a árvore principal.

$$Dist(A,B)-(Dist(A, v)+Dist(B, v))=0 \quad (14)$$

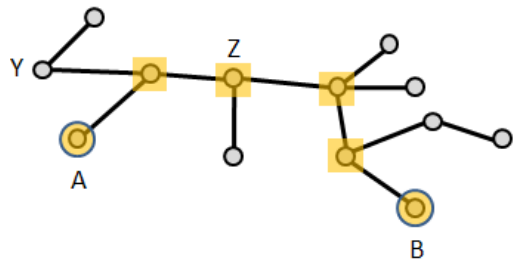


Figura 3.7: Identificação da subárvore principal na AGM.

- *Vértice Y* ***não*** **pertence** a árvore principal:

  Dist(A,B) - (Dist (A, Y) + Dist (B, Y)) = 0

  5 - (2 + 5) = 0

  - 2 = 0 (Falso)

- *Vértice Z* **pertence** a árvore principal:

  Dist(A,B) - (Dist (A, Z) + Dist (B, Z)) = 0

  5 - (2 + 3) = 0

  0 = 0 (Verdadeiro)

Os vértices que não pertencem a subárvore principal serão agrupados ao vértice pertencente a esta subárvore mais próximo, conforme a equação Dist. Dessa forma, o grafo ficará semelhante a um "*cordão*" e os vértices pertencentes a subárvore principal possuirão o somatório das capacidades dos vértices a eles agrupados. A Figura 3.8 (a) apresenta os números em um dos *clusters,* que indicam as capacidades dos vértices. Já a Figura 3.8 (b) apresenta os vértices exibidos na figura 3.8 (a) agrupados ao vértice pertencente a árvore principal mais próximo, com a capacidade total atualizada com o somatório das capacidades dos vértices a ele associados.

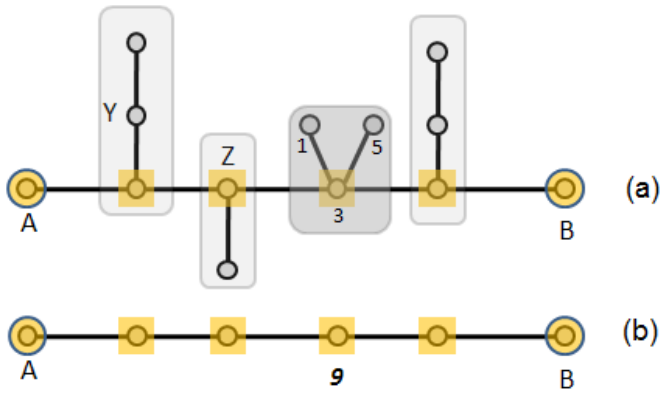


Figura 3.8: Formação dos grupos de vértices na subárvore principal.

O próximo passo é particionar a subárvore principal formando os *clusters*. Para isso, a partir de um dos vértices selecionados para a formação da subárvore principal (neste exemplo os vértices A e B), esse procedimento adiciona o grupo de vértices da subárvore principal adjacente a esse vértice. Além disso é necessário verificar se o somatório de sua capacidade é superior ao limite inferior pré-estabelecido. Dessa forma, quando esta restrição de capacidade mínima é satisfeita, o *cluster* está formado e um novo *cluster* é iniciado a partir dos vértices restantes na subárvore principal.

O particionamento da subárvore principal continua sua execução enquanto a quantidade pré-fixada de clusters não for obtida ou existir algum agrupamento de vértices na subárvore principal ainda não adicionado a um *cluster*. A Figura 3.9 apresenta a execução desse procedimento passo a passo, considerando a capacidade mínima de 7 unidades e a obtenção de 3 *clusters*. Os valores associados a cada um dos vértices representam as capacidades.

- **Passo 1:** início da execução do particionamento da subárvore principal. O cluster C1 é penalizado pois possui a capacidade 4 unidades, enquanto o limite inferior estabelecido é de 7 unidades. Sendo assim, para atender a restrição de capacidade, devem ser adicionados a este *cluster* mais vértices da subárvore.
- **Passo 2:** o grupo de vértices vizinhos ao cluster C1 que não está associado a nenhum cluster é adicionado a ele. Com um total de 9 unidades este cluster atende a restrição de capacidade mínima. Assim o *cluster* C1 é formado e o cluster C2 começa a ser construído. Com um total de 6 unidades este cluster é penalizado, e conseqüentemente, necessita da inclusão de mais vértices da subárvore para atender a restrição de capacidade.
- **Passo 3:** o grupo de vértices vizinhos ao cluster C2 que não está associado a nenhum cluster é adicionado a ele, definindo o cluster C2 com um total de 12 unidades, o que

implica na imediata satisfação da restrição de capacidade mínima. Finalmente, inicia-se a formação do cluster C3. Com um total de 7 unidades este cluster atende a restrição de capacidade, porém, uma vez que o problema solicita a formação de 3 clusters e ainda existe um grupo de vértices que não estão associados a nenhum *cluster* o algoritmo deve continuar a sua execução.

- **Passo 4:** Os vértices restantes, não associados a nenhum *cluster*, são adicionados ao cluster C3 e finalmente obtêm-se o número de *clusters* pré-fixado e o procedimento de construção é concluído.

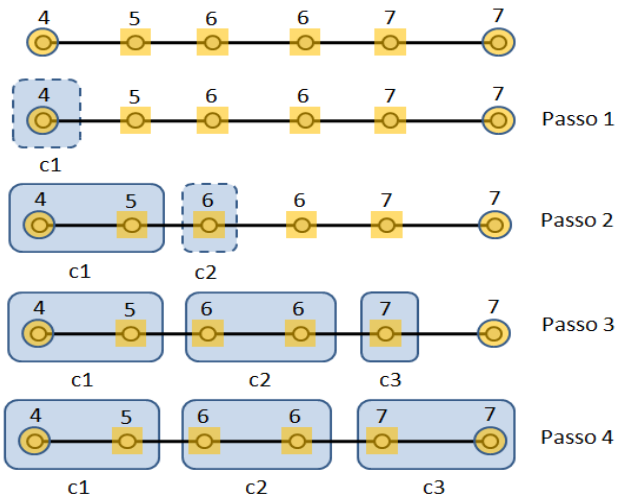


Figura 3.9: Exemplo de execução do procedimento construtor 2

Para geração de um conjunto soluções viáveis, isto é, que satisfaçam a restrição de capacidade, utiliza-se neste procedimento um parâmetro que assume valores aleatórios. De acordo com o valor que este parâmetro assume, o cluster pode receber novos vértices, ainda que a restrição de capacidade já tenha sido satisfeita. Além da obtenção de soluções válidas, cuja restrição de capacidade seja satisfeita, desta forma ocorre uma diversificação das soluções construídas. A Figura 3.10 apresenta um exemplo de obtenção de uma solução distinta devido a mais esta verificação, considerando a formação de 3 *clusters* e a capacidade mínima de 7 unidades:

- **Passo 1:** o grafo apresenta os grupos de vértices que fazem parte da subárvore principal antes do particionamento.

- **Passo 2:** apesar do cluster C1 já atender a restrição de capacidade mínima, ou seja, o somatório do atributo referente a capacidade dos vértices contidos no *cluster* C1 ser superior ao mínimo pré-estabelecido, um fator aleatório determina que o cluster ainda pode receber mais vértices da subárvore.
- **Passo 3:** com a restrição de capacidade do cluster C1 já atendida, sua formação depende apenas do fator aleatório desse procedimento. Nesse passo o fator aleatório confirma a formação desse cluster e, em seguida, o novo *cluster* C2 é iniciado no próximo grupo de vértices pertencentes a subárvore principal. O cluster C2, porém, não possui capacidade suficiente para atender a restrição.
- **Passo 4:** o próximo grupo de vértices é adicionado ao *cluster* C2. A capacidade desse cluster satisfaz a restrição mínima. Além disso, o fator aleatório indica se deve ocorrer a formação do *cluster* C2 e iniciar a construção do novo cluster, C3.
- **Passo 5:** o grupo de vértices vizinho ao cluster C2 que não está associado a nenhum cluster é selecionado para a formação do cluster C3. A capacidade desse cluster satisfaz a restrição. Como não existem mais grupos de vértices, independente do valor do parâmetro, o cluster C3 é formado.

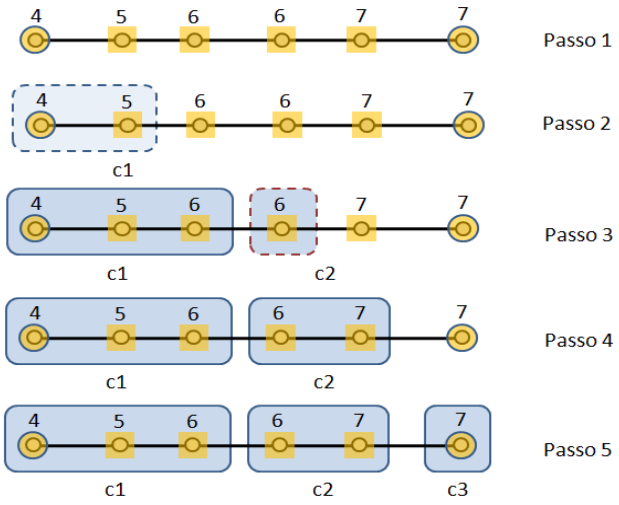


Figura 3.10: Exemplo de execução do procedimento construtor 2 com fator aleatório.

Ao final do procedimento, é possível a construção de uma solução para qual a quantidade de *clusters* seja inferior ao determinado previamente. Nesse caso, torna-se necessário regenerar a solução, mediante a seleção do *cluster* com maior capacidade

excedente para, em seguida, a terceira versão do procedimento construtivo, apresentado na seção 3.3.2.3, particioná-lo.

Essa operação ocorre até que a quantidade de *clusters* definida seja atingida ou até que não seja mais possível dividir o cluster em função da violação da restrição de capacidade. A Figura 3.11 apresenta uma situação na qual a regeneração da solução é necessária considerando $k=3$:

- **Passo 1:** nesse exemplo considera-se a solução com os c*lusters* C1 e C2 já formados, porém sendo necessária a formação de 3 *clusters*. Assim, para a regeneração da solução, torna-se necessário o particionamento de um dos clusters. Para isso é realizado o cálculo da capacidade total por cluster, o que determina qual cluster será particionado.
- **Passo 2:** Os *clusters* C1 e C2 possuem, respectivamente, capacidades de 15 e 20 unidades. Assim o *cluster* C2 é selecionado para ser particionado através do procedimento construtor 3.
- **Passo 3:** o procedimento construtor indica a aresta a ser removida para a formação de dois novos *clusters* C2 e C3. A aresta selecionada não pertence necessariamente a subárvore principal.
- **Passo 4:** ocorre o particionamento do cluster selecionado. A solução produzida possui o número de clusters pré-fixado e o procedimento finaliza sua execução.

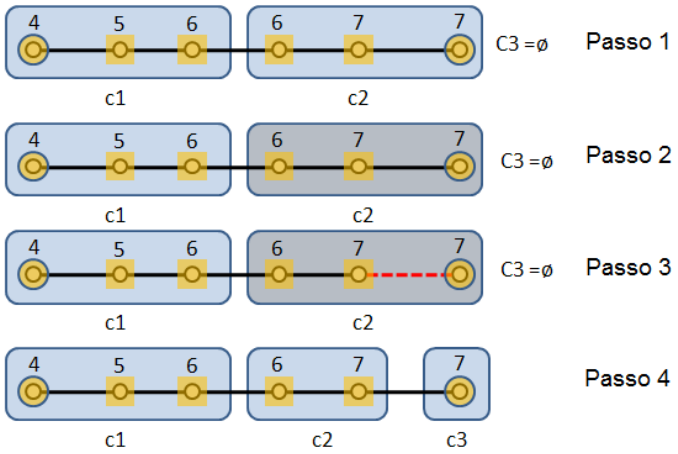


Figura 3.11: Exemplo de execução do procedimento construtor 2 regenerando a solução.

3.3.2.3 Procedimento Construtor 3

Assim como procedimento construtor 2, a terceira versão de procedimento construtor atua especificamente na tentativa de formação de soluções válidas em que, além da restrição de conexidade, seja satisfeita também a restrição de capacidade mínima por *cluster*. Embora essa versão possua outro objetivo em relação ao procedimento construtor 1, seus algoritmos são semelhantes. Esse procedimento consiste basicamente em duas etapas, que devem ser executadas k – 1 vezes para a obtenção de k *clusters*:

- **Seleção do cluster a ser particionado**: o *cluster* que possuir o maior valor considerando o somatório do atributo capacidade em seus objetos é selecionado para ser particionado, exceto na primeira iteração do procedimento, em que existe apenas um cluster inicial composto pela AGM.
- **Definição da aresta a ser removida:** após a seleção do *cluster* a ser particionado é necessário definir a aresta a ser removida. Para isso, assim como na primeira versão do procedimento construtor, é utilizado um procedimento guloso, em que todas as possibilidades de remoção de arestas do *cluster* selecionado são executadas. Porém, nessa versão, a partir do *cluster* selecionado, o objetivo é a formação de dois novos *clusters* que atendam a restrição de capacidade mínima e, além disso, que a capacidade de um dos novos *clusters* seja maximizada, o que aumenta a chance da formação de novos *clusters* sem penalidade nas iterações seguintes.

Assim como na primeira versão do procedimento construtivo, para possibilitar a diversificação de soluções, este procedimento foi adaptado para não possuir comportamento guloso. Porém, nesta versão, a LRC é formada pelas α arestas que, se removidas, possibilitam a geração das α soluções que atendam a restrição de capacidade mínima e, ao mesmo tempo, maximizem a capacidade de um dos clusters. A remoção de uma das arestas desta LRC também ocorre de maneira aleatória, formando dois novos clusters.

3.3.3 Procedimentos de Busca Local

Segundo [Dias, 2004], muitos trabalhos utilizam a técnica de busca local na tentativa da melhoria da qualidade das soluções. Ou seja, uma redução da função objetivo no caso de minimização e aumento da função objetivo em caso de maximização. Essa técnica consiste basicamente em, partindo de uma solução corrente, analisar a vizinhança de seus elementos,

objetivando a melhoria de sua qualidade. Nesse trabalho foram implementadas seis versões de procedimentos de busca local, que atuam na tentativa de eliminação de penalidades, referentes ao não atendimento da restrição de capacidade mínima por cluster, e na melhoria da solução com a minimização da função aptidão. É importante destacar que em todas as versões a restrição de conexidade do problema é satisfeita.

3.3.3.1 Procedimento Busca Local 1

O Procedimento de Busca Local 1 tem como foco principal a eliminação de penalidades. Para isso, ele utiliza uma lista de arestas removidas da AGM e uma lista de *clusters* penalizados. A partir da utilização dessas listas, o procedimento verifica a possibilidade de realizar a migração de vértices entre os *clusters*.

Para cada aresta da lista, identifica-se a quais *clusters* seus vértices pertencem. Em seguida é verificada a necessidade de migração de vértices entre esses *clusters*, o que deve ocorrer, se e somente se apenas um desses clusters estiver penalizado. Além disso, observa-se que uma vez que apenas a AGM é considerada neste procedimento, tornam-se necessárias as seguintes verificações para que a restrição de conexidade seja garantida, assim como em todos os demais procedimentos:

- *S*e o *cluster* possui apenas um vértice, esse não deve ser migrado, uma vez que o cluster ficaria vazio e a quantidade estabelecida de clusters necessários não seria respeitada;
- Se o vértice é adjacente a pelo menos dois vértices no cluster ele não deve ser migrado, uma vez que a sua remoção deixaria o cluster desconexo.

Conforme apresenta a Figura 3.12, considerando o cluster C2 penalizado, o *cluster* C1 possui apenas o vértice A e não deve ser migrado para o *cluster C2* pois o *cluster* C1 ficaria vazio. O vértice c, é adjacente a pelo menos dois vértices no cluster e pode ser migrado para o cluster C2, como ilustra a Figura 3.12.

Na Figura 3.13 (a) o Vértice B é adjacente a dois vértices pertencentes ao mesmo *cluster*, não podendo ser migrado para o cluster C2 pois a restrição de conexidade neste cluster tornaria a solução inválida, deixando o cluster C3 desconexo (Figura 3.13 (b)), e a geração de um novo cluster não atenderia ao problema de clusterização considerando $k=3$ (Figura 3.13 (c)).

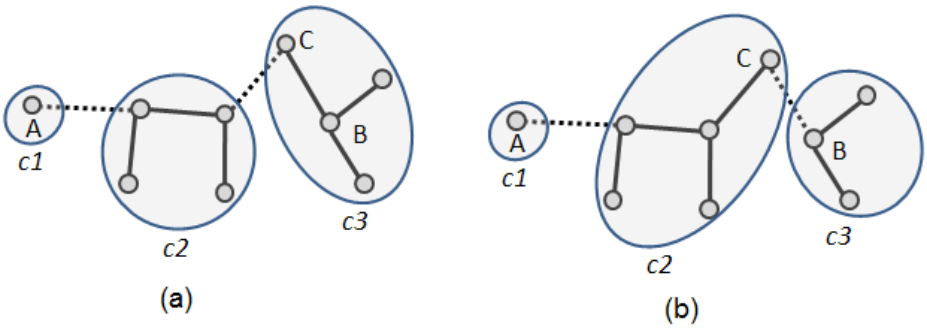


Figura 3.12: Exemplo da aplicação do procedimento de busca local 1

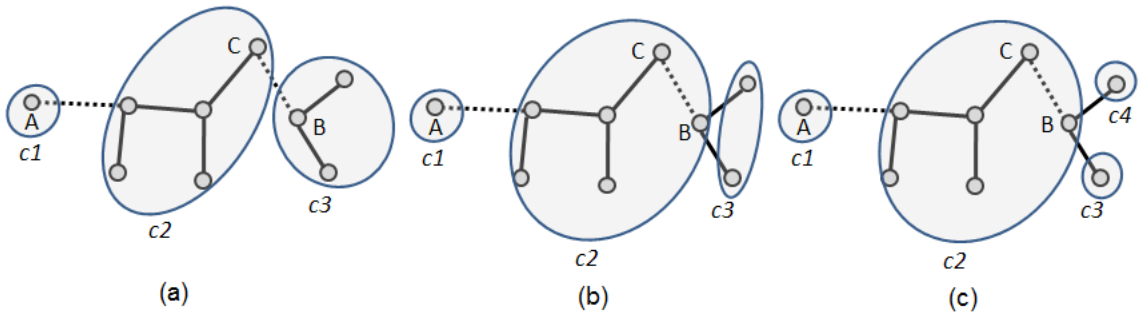


Figura 3.13: Exemplo de possibilidades de migração de vértices.

Caso seja confirmada a possibilidade de migração, o vértice da aresta removida que pertence ao *cluster* não penalizado migrará para o *cluster* penalizado. Essa operação não garante que o cluster penalizado se torne um cluster viável, que a migração do vértice faça com que o cluster de origem seja penalizado e nem mesmo a melhoria da qualidade da solução. Entretanto, a sua sistemática aplicação, pode auxiliar na produção de soluções válidas.

A Figura 3.14 apresenta um exemplo de migração de vértices considerando a capacidade mínima de 6 unidades e a capacidade dos vértices *a,b,c,d,e,f* e *g* serem respectivamente 5, 2, 3, 4, 5, 6 e 7. Na solução 1 o *cluster C1* encontra-se penalizado, possuindo como capacidade apenas 2 unidades.

Esse procedimento atua nas arestas removidas da AGM durante o seu particionamento e que possuam vértices no *cluster C1*. Ainda na Figura 3.14 (a) é possível observar que a aresta com vértices A e B foi removida para a formação dos clusters C1 e C2.

O próximo passo é verificar se as condições para a realização da migração são atendidas. Para isto, o procedimento de busca local 1, apresentado na Figura 3.12, deve ser executado. O vértice A pertence a um cluster não penalizado e, além disso, é adjacente a um

único vértice do mesmo cluster, atendendo assim aos requisitos para que a sua migração seja realizada. O vértice A é migrado do *cluster C2* para o *cluster C1.* A solução é recalculada e os *clusters C1 e C2* possuem ambos 7 e 7 unidades, atendendo à restrição de capacidade e eliminando a penalidade do cluster C1, conforme apresenta a Figura 3.14 (b).

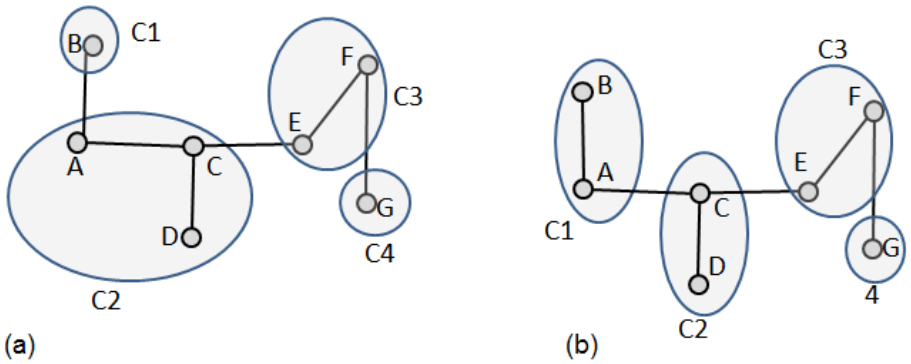


Figura 3.14. Exemplo de migração de vértice entre clusters da Busca Local 1.

3.3.3.2 Procedimento Busca Local 2

Assim como o Procedimento de Busca Local 1, esse procedimento também atua especificamente na tentativa de eliminação de penalidades, independente da melhoria da solução. Porém, nessa versão, ocorre a migração de vértices entre *clusters* considerando o grafo original e não apenas a AGM construída. Para isso, torna-se necessária a utilização de um procedimento para verificar se as restrições de conexidade entre os vértices internos aos *clusters* estão sendo satisfeitas, ou seja, se os vértices internos aos clusters estão conexos.

A primeira etapa do algoritmo consiste em selecionar aleatoriamente uma aresta $e=(i,j)$ do grafo original, antes da construção. E em seguida, são realizadas as seguintes verificações:

- Quando apenas um dos *clusters* em que os vértices da aresta selecionada pertence deve estar penalizado. Caso contrário, o algoritmo seleciona a próxima aresta também de forma aleatória;
- Se apenas um dos clusters estiver penalizado, este será candidato a receptor do vértice a ser migrado e o *cluster* não penalizado será candidato a doador. Observando que a migração do cluster doador será realizada, se esse não ficar penalizado.

- A migração do vértice para o *cluster* receptor deve ocorrer somente se os demais vértices do *cluster* doador forem conexos entre si, ou seja, atenderem a restrição de conexidade.

Caso estas verificações estejam atendidas, ocorre a migração do vértice do *cluster* doador para o *cluster* receptor. A Figura 3.15 (a) apresenta um exemplo da execução desse procedimento. As arestas em negrito correspondem a AGM construída e as demais arestas em conjunto com a AGM fazem parte do grafo original. Com o objetivo de ilustrar as situações que o procedimento deve tratar, são apresentadas algumas tentativas de migrações, tais como:

- Considerando apenas a AGM (arestas em negrito), a migração do vértice b, que pertence ao cluster C1, não seria possível, uma vez que não existem arestas da AGM entre este vértice e nenhum vértice de nenhum outro *cluster*. Porém, considerando o grafo original, existe uma aresta entre este vértice e o vértice que pertence ao *cluster* C2. Nesse exemplo é considerado que o *cluster* C2 esteja penalizado e que, além do cluster C1 não estar penalizado, a migração do cluster b não penalizaria este *cluster*. Além disso, pode-se observar que os demais vértices do cluster C1 são conexos entre si. Uma vez realizadas as verificações necessárias e considerando que os requisitos para a migração foram atendidos, ocorre a migração e o vértice b passa a pertencer ao cluster C2, conforme apresenta a Figura 3.15 (b).
- Apesar do vértice c, pertencente ao *cluster* C4, possuir conexões com os vértices dos clusters C2 e C3, por ser o único vértice do *cluster* ele não pode ser migrado, uma vez que o cluster C4 ficaria vazio e a solução não iria possuir a quantidade de clusters pré-estabelecida.
- Ainda considerando a Figura 3.15 (a), o vértice f, que possui conexão com o *cluster* C2 através do vértice e, não pode ser migrado uma vez que o cluster ficaria desconexo.

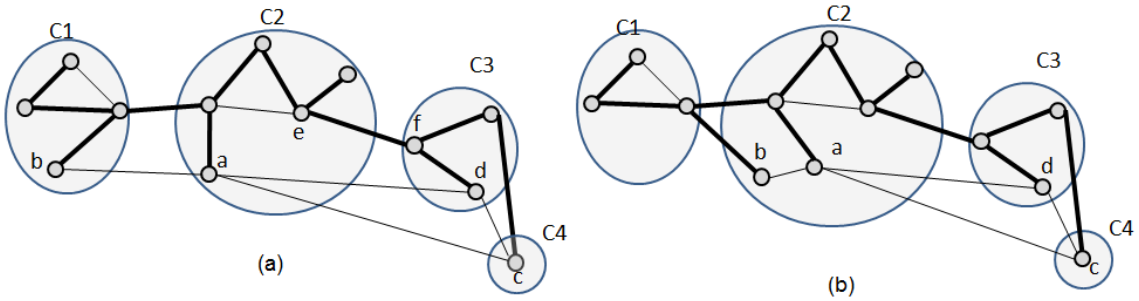


Figura 3.15: Exemplo de execução do Procedimento de Busca Local 2.

3.3.3.3 Procedimento Busca Local 3

O Procedimento de Busca Local 3 possui como único objetivo a minimização da aptidão da solução. Para isto, assim como o Procedimento Busca Local 2, ele atua na migração de vértices entre *clusters* considerando o grafo original e a AGM construída. Assim, torna-se novamente necessário executar o procedimento que verifica se a restrição de conexidade é satisfeita quando ocorre a migração de vértices.

Essa busca local possui algumas semelhanças com a Busca Local 2, sobretudo nas verificações necessárias para a realização da migração de um vértice. A primeira etapa do algoritmo consiste em selecionar uma aresta $e=(i,j)$ do grafo original aleatoriamente. Em seguida, as verificações abaixo realizadas:

- Os vértices da aresta selecionada devem estar em *clusters* distintos;
- Ambos os *clusters* serão candidatos tanto a receptor quanto a doador.
- Os clusters não devem estar penalizados;
- Assim como a segunda versão do procedimento de busca local, é necessário verificar se a migração do vértice penaliza o cluster doador. Caso isto ocorra, a migração não pode ser realizada. Nessa versão, essa verificação deve ocorrer em ambos os sentidos, ou seja, cada cluster pode atuar tanto como doador quanto como receptor de vértices.
- Mesmo com a migração do vértice para o *cluster* receptor, os demais vértices do *cluster* doador devem atender a restrição de conexidade;
- A realização da migração ocorre apenas quando há redução no valor da função objetivo.

3.3.3.4 Procedimento Busca Local 4

O Procedimento Busca Local 4 trabalha com as arestas removidas na geração dos clusters. Esta versão objetiva a busca por soluções de melhor qualidade através da união e do reparticionamento de *clusters* inicialmente construídos.

Os *clusters* aos quais os vértices das arestas removidas pertencem, selecionadas de forma aleatória, podem se unir em um *cluster* para, em seguida, este *cluster* ser particionado novamente através da aplicação do procedimento construtivo 1.

A Figura 3.16 apresenta a aplicação desse procedimento em um grafo de exemplo seguido dos comentários. Na figura 3.16(a), se um valor real gerado aleatoriamente for inferior à probabilidade submetida como parâmetro ao algoritmo, a aresta removida e(A,B) é selecionada. Os vértices dessa aresta pertencem respectivamente aos *clusters* C1 e C2. Já na figura 3.16(b), os *clusters* C1 *e* C2 são unidos, seus elementos fazem parte de um mesmo *cluster* (C2) e, temporariamente, um dos clusters ficará vazio (C1). Após a união desses dois *clusters*, deve ocorrer a seleção da aresta a ser removida do cluster formado na busca por soluções de melhor qualidade, o que ocorre através da utilização do procedimento construtor 1.

Nesse exemplo, após o reparticionamento do *cluster* C2*,* foram determinados os novos *clusters* C1 e C2 possuindo os vértices A , B e C , D respectivamente, conforme apresenta a Figura 3.16 (c). O procedimento considera ainda a aresta removida antes da execução do procedimento, evitando que ocorra a remoção da mesma aresta e, com isto, garantindo a formação de soluções diferentes.

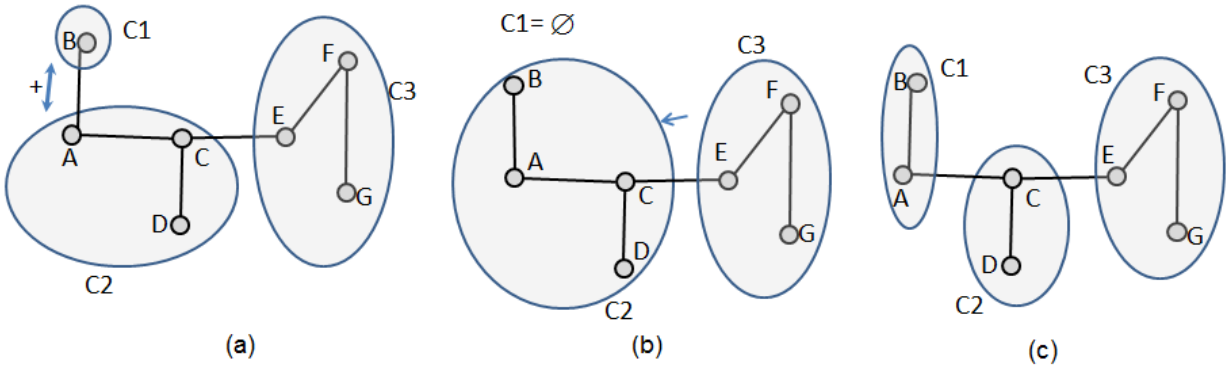


Figura 3.16. Exemplo de execução do Procedimento Busca Local 4.

3.3.3.5 Procedimento Busca Local 5

O procedimento Busca Local 5 é semelhante ao Procedimento Busca Local 4, mas é aplicado com o objetivo de obter novas soluções não penalizadas, ou seja, que atendam a restrição de capacidade mínima. Após a união de dois *clusters*, que ocorre de forma aleatória conforme a lista de arestas removidas, seleciona-se para ser particionado o cluster que possuir a maior capacidade. Já a seleção da aresta a ser removida consiste em um procedimento guloso, que implica em considerar todas as possibilidades de remoção de arestas no cluster selecionado.

O procedimento forma uma LRC composta pelas α arestas que, se removidas, definem *clusters* que atendam a restrição de capacidade mínima e, simultaneamente, possuam a maior diferença, em valor absoluto, entre o somatório das capacidades dos *clusters* obtidos, conforme apresenta a Equação 15. Nesta equação, *getTotal()* retorna o somatório das capacidades de um *cluster* Ci e *getCluster()* retorna o cluster em que está contido o vértice v.

Em seguida, uma das arestas da LRC é aleatoriamente selecionada e removida, provocando o particionamento do *cluster* e determinando a quais *clusters* esses vértices pertencerão. A Figura 3.17 apresenta um exemplo dessa versão de busca local. Nessa figura os números representam o atributo capacidade dos vértices e as letras as arestas candidatas a remoção.

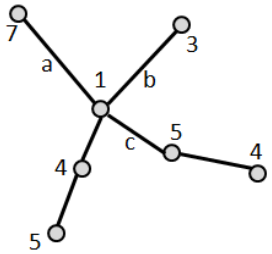


Figura 3.17: Grafo utilizado no exemplo de demonstração do procedimento construtor 3

$$CustoMut3_e(x, y) = | getTotal(getCluster(x)) - getTotal(getCluster(y)) | \quad (15)$$

Com base na Figura 3.17, seguem exemplos de arestas candidatas a remoção. Já a Figura 3.18 apresenta soluções obtidas com a remoção das arestas.

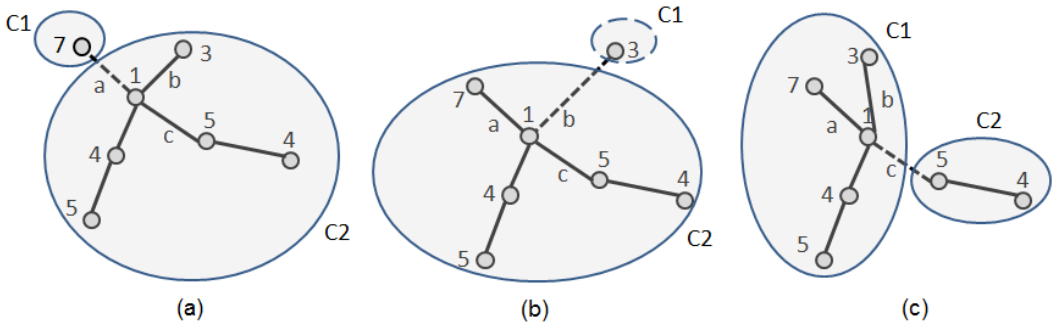


Figura 3.18: Exemplo de execução do procedimento construtor 3.

- Aresta *a* na Figura 3.18 (a): C1 possui capacidade 7 e C2 possui capacidade 22. Ambos os clusters atendem a capacidade mínima. Utilizando a Equação 15 se obtém o valor 15.
- Aresta *b* na Figura 3.18 (b): C1 possui capacidade 3 e C2 possui capacidade 26. *Cluster* C1 não atende a capacidade mínima.
- Aresta *c* na Figura 3.18 (c): C1 possui capacidade 20 e C2 possui capacidade 9. Ambos os *clusters* atendem a capacidade mínima. Utilizando a Equação 15 se obtém o valor 11.

Dentre os exemplos avaliados, a remoção da aresta *b* iria gerar um cluster com penalidade. Assim, apenas as arestas *a* e *c* seriam adicionadas na LRC, uma vez que o objetivo é maximizar o valor da Equação 15. Uma das arestas da LRC é aleatoriamente selecionada e removida, formando 2 *clusters*.

3.3.3.6 Procedimento Busca Local 6

A Busca Local 6 objetiva a minimização da aptidão da solução e, para isso, realiza migrações de vértices entre *clusters,* com base no o grafo original. Os passos desse procedimento são apresentados no Algoritmo 6 e a Figura 3.19 apresenta sua execução, passo a passo.

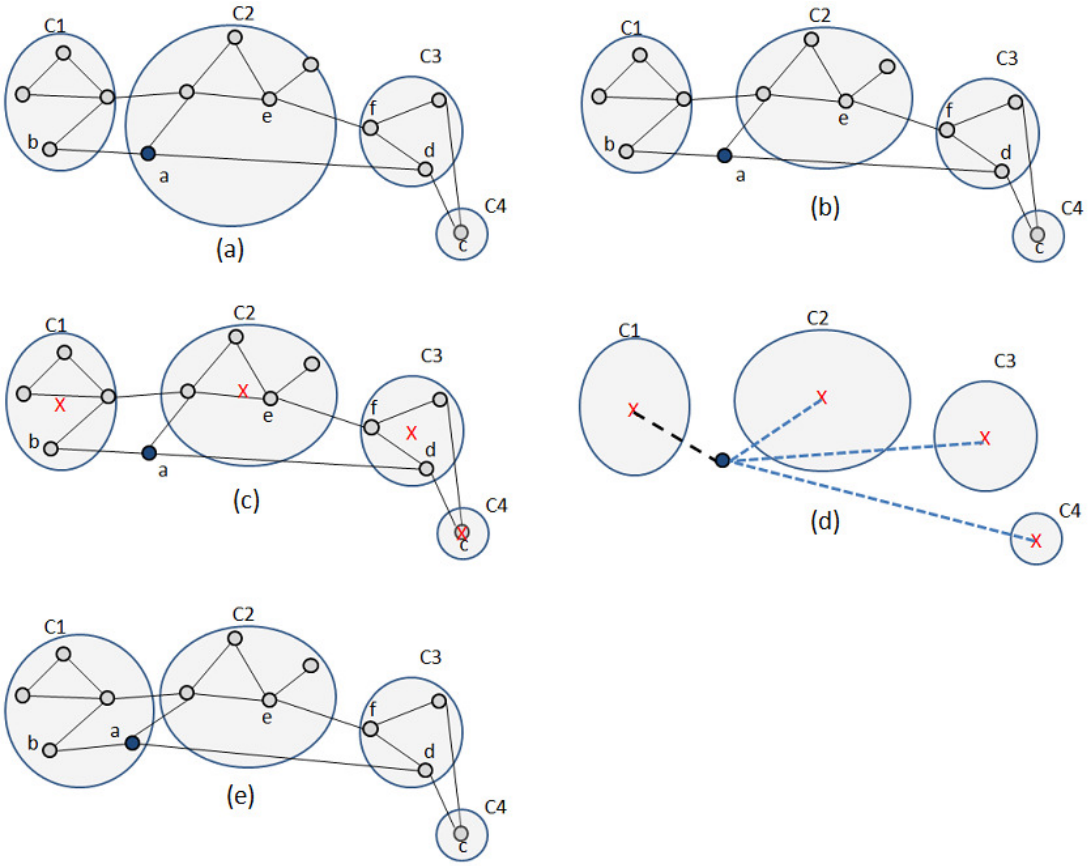


Figura 3.19: Procedimento Busca Local 6.

- **Passo 1:** o procedimento ordena os vértices de forma decrescente através da soma de seus atributos. Iterativamente, conforme a ordenação realizada, cada vértice é selecionado como candidato para a migração. No exemplo apresentado na Figura 3.22 (a) o vértice a foi selecionado.
- **Passo 2:** uma vez selecionado o vértice a, é necessário verificar se pode ocorrer a migração. Para isso, a primeira verificação consiste em analisar se a restrição de conexidade continua sendo satisfeita no cluster C2, caso o vértice a seja migrado. Nesse exemplo o vértice a pode ser migrado e a Figura 3.22 (b) mostra o vértice isolado, ou seja, não associado a nenhum cluster.
- **Passo 3:** após o vértice candidato estar isolado são calculados os centróides de cada cluster. Cada cluster possui a média da soma de seus atributos. Na Figura 3.22 (c) o caractere “X” ilustra, simbolicamente, a posição correspondente aos valores calculados para cada cluster.

- **Passo 4:** após o cálculo dos centróides avalia-se a qual cluster o vértice candidato é mais semelhante. A Figura 3.22 (d) exibe linhas tracejadas que simbolizam a distância (dissimilaridade) entre o vértice candidato e cada um dos clusters. A linha que conecta o vértice ao cluster C1 está em destaque e indica que o vértice é mais semelhante a esse cluster.
- **Passo 5:** uma vez identificado que o cluster C1 é o mais próximo do vértice, verifica-se se algum vértice pertencente a este cluster possui conexão com o vértice a, ou seja, se a restrição de conexidade é satisfeita. Nesse exemplo existe conexão entre os vértices a e b, o que confirma a possibilidade de migração.
- **Passo 6:** o objetivo desse procedimento é a minimização da aptidão da solução. Desta forma, a migração é realizada somente se a houver a melhoria da qualidade da solução.

3.3.4 Procedimento de Mutação

No Procedimento Mutação a migração de vértices entre clusters é semelhante ao do procedimento de busca local 1, porém com o objetivo de perturbar a solução, evitando que esta fique estagnada em ótimos locais e regenerando possíveis soluções que tenham sido eliminadas pelos outros procedimentos. Para isso, conforme uma probabilidade submetida como parâmetro ao algoritmo, um dos vértices de cada aresta removida pode ser migrado. Para a execução desse procedimento**,** torna-se necessário verificar se o vértice pode ser migrado, conforme as condições apresentadas no Procedimento de Busca Local 1. A Figura 3.19 apresenta um exemplo deste procedimento de mutação, conforme os passos a seguir:

- **Passo 1:** de forma aleatória uma aresta da lista de arestas removidas para a formação de *clusters* é selecionada. Neste exemplo trata-se da aresta que conecta os vértices A e B.
- **Passo 2:** o vértice A é migrado para o *cluster* em que se encontra o vértice B. Novamente ocorre a seleção entre as arestas da lista de arestas removidas para o particionamento de clusters. A aresta que conecta os vértices C e E é selecionada.
- **Passo 3:** O vértice E pertencente a aresta selecionada no passo anterior é migrado para o *cluster* em que o vértice C pertence.

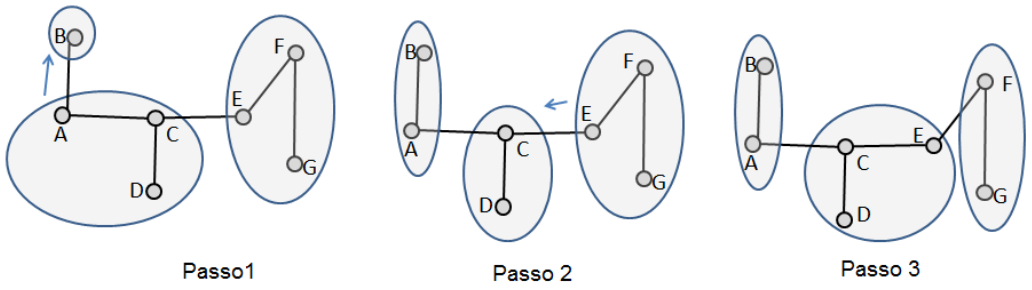


Figura 3.19: Exemplo de migração da primeira versão do procedimento de mutação.

3.3.5 Procedimentos de Cruzamento

Nesse trabalho, o ponto de corte para a realização do cruzamento entre as soluções corresponde a uma das arestas removidas (tracejadas na Figura 3.20) na construção dos *clusters*. Para isso é necessária a identificação de uma aresta comum, que tenha sido removida nas soluções candidatas ao cruzamento. Além disso, o número de *clusters* existentes entre as partes identificadas pelo ponto de corte devem ser equivalentes. Sem essa verificação é possível a obtenção de soluções inválidas, para as quais as quantidades de *clusters* não correspondam com o fixado previamente. A Figura 3.20 apresenta a execução do cruzamento entre as soluções S_1 e S_2 e a geração das soluções $S_1"$ e $S_2"$.

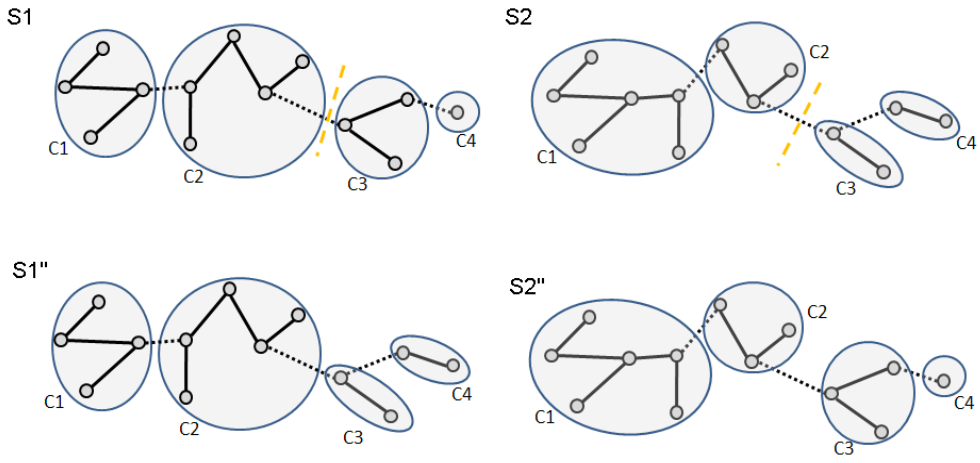


Figura 3.20: Exemplo de aplicação do procedimento de mutação.

3.3.6 Procedimento de Seleção

O Procedimento de Seleção deve simular o mecanismo de seleção natural das espécies biológicas, em que pais mais aptos geram mais filhos. Porém, é necessário também que pais menos aptos possam gerar descendentes. No modelo computacional isto significa privilegiar soluções com melhores aptidões e, ao mesmo tempo, não ignorar soluções que possuam aptidões ruins. Isto pode ser explicado pelo fato de uma solução ruim poder possuir características interessantes e que podem contribuir para a formação de soluções melhores.

No contexto desse trabalho, pode-se exemplificar esta situação através de uma solução obtida considerando k = 3, em que um cluster satisfaz as restrições impostas pelo problema e possui aptidão interessante, mas que ao mesmo tempo, os demais clusters não sejam tão interessantes do ponto de vista da qualidade da solução, possuindo valores altos, ou mesmo não atendendo as restrições de do problema abordado. Dessa forma, uma solução melhor pode ser obtida caso os vértices do *cluster* de boa qualidade sejam mantidos unidos em um mesmo *cluster* e os demais vértices sejam reorganizados nos outros *clusters*.

Nos Algoritmos Genéticos as iterações também podem ser denominadas gerações. Cada geração G_i é composta pelas soluções resultantes da aplicação dos operadores genéticos e demais procedimentos adicionais nas soluções selecionadas da geração anterior G_{i-1}. Segundo [Beasle et al., 1993] apud [Dias, 2004] existem várias estratégias para a seleção, sendo as mais utilizadas a seleção pelo método do torneio e o pelo método da roleta. Nesse trabalho utilizou-se esses dois métodos de seleção.

Além do procedimento de seleção utilizado, foi implementado um procedimento de Elitismo, com o objetivo de garantir que a melhor solução obtida seja armazenada e inserida na população seguinte (G_{i+1}). As soluções selecionadas formarão a população da geração G_{i+1} e serão submetidas aos operadores genéticos e demais procedimentos adicionais.

Para o problema abordado nesse trabalho, são armazenadas as melhores soluções obtidas com e sem penalidade. No início de cada iteração, a melhor solução é inserida na população objetivando, através dos demais procedimentos como buscas locais, mutações e cruzamento, a melhoria de sua qualidade.

Quando não há uma solução sem penalidade, cuja restrição de capacidade mínima por cluster seja satisfeita, a melhor solução obtida até o momento é inserida na população. Ao final

de sua execução, o algoritmo possuirá a melhor solução penalizada e/ou a melhor solução sem penalidade.

3.3.6.1 Seleção pelo Método da Roleta

Este método foi proposto inicialmente por [Assunção et al., 2006] e a idéia base é que a seleção seja relacionada com a aptidão (qualidade) das soluções [Glover and Kochenberger, 2002]. Basicamente, deve-se atribuir para as soluções da população na geração G_n probabilidades de permanecer na população da geração seguinte (G_{n+1}) proporcionais as suas aptidões.

Sua denominação deve-se a analogia com o jogo de roleta, em que cada solução possui uma área na roleta proporcional à sua aptidão. Para a formação da população da geração seguinte (G_{n+1}) esse procedimento deve ser executado na quantidade equivalente ao tamanho da população, uma vez que em cada execução apenas uma solução é selecionada.

A Figura 3.21 apresenta o exemplo utilizado por [Glover and Kochenberger, 2002] considerando uma população formada pelas soluções 1, 2, 3, 4 e 5 com aptidões 32, 9, 17, 17 e 25, respectivamente. Para ilustrar uma roleta com base nesse exemplo, a solução 1 possui na roleta dessa figura área no intervalo $]0,32]$, a solução 2 no intervalo $]32,41]$, e assim por diante. Ainda com Base na Figura 3.21, para a seleção de uma solução, obtém-se o número aleatório, neste exemplo o número 13. Conforme a seta apresentada nesta figura, considerando a obtenção do número aleatório, indica a seleção da solução 1 para a próxima geração.

Em problemas cujo objetivo é a minimização da função de aptidão, uma vez que a melhor solução é a que possuir menor valor de aptidão, esse procedimento deve ser adaptado e áreas maiores devem ser atribuídas às soluções que possuem valores de aptidão menores.

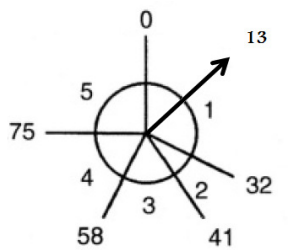


Figura 3.21: Exemplo de execução do procedimento de roleta
[Glover and Kochenberger, 2002].

3.3.6.2 Seleção pelo método do Torneio

Conforme [Glover and Kochenberger, 2002], o procedimento de seleção por torneio, assim como o procedimento de seleção por roleta, também é associado ao valor da aptidão das soluções. Esta seleção consiste em, a partir de um parâmetro ℓ submetido ao algoritmo, formar uma lista de candidatos de tamanho ℓ com soluções obtidas, de forma aleatória, da população na geração corrente (G_n). Após a formação dessa lista, a melhor solução nela contida é inserida na população da próxima geração (G_{n+1}).

Dessa forma, assim como a seleção utilizando o método da roleta, o procedimento de Torneio deve ser executado a quantidade de vezes equivalente ao tamanho da população. A mesma solução pode ser selecionada e inserida na lista de candidatos mais de uma vez, o que permite que todas as soluções possam ser selecionadas, inclusive as soluções que possuem valores de aptidão altos, ou seja, soluções de baixa qualidade conforme o problema abordado.

3.4 Algoritmos Heurísticos Propostos

Os algoritmos heurísticos propostos nessa dissertação são o resultado de combinações dos procedimentos apresentados na seção 3.3. Conforme já observado, tais procedimentos foram implementados a partir do estudo de conceitos de algoritmos genéticos, algoritmos evolutivos e do GRASP.

3.4.1 Algoritmos Evolutivos Propostos

O Algoritmo 5 apresenta uma versão de algoritmo evolutivo que utiliza os procedimentos apresentados na seção 3.3. As versões desses algoritmos nesse trabalho possuem características específicas, tais como:

- Utilização de um algoritmo auxiliar para a construção de uma AGM a partir do grafo submetido ao problema antes da construção das soluções iniciais.
- Os procedimentos construtores de soluções iniciais atuam na construção soluções de forma a minimizar o valor da função de avaliação para a solução utilizada no trabalho ou mesmo

objetivando a formação de soluções válidas, que atendam tanto a restrição de conexidade quando o de capacidade mínima por cluster.

- Em todos os procedimentos a restrição de conexidade é garantida. Já restrição de capacidade mínima por cluster pode não ser satisfeita em algumas situações. Porém, procedimentos de busca local atuam nessas soluções, tentando eliminar eventuais penalidades através da migração de vértices. O objetivo é que a restrição de capacidade mínima por *cluster* seja satisfeita para todos os *clusters*.
- O método de seleção utilizado pelo Cruzamento é específico e necessita de validações que garantam a construção de soluções quem atendam a restrição de conexidade e que possuam a quantidade de *clusters* especificada.
- Apesar do cálculo da função de avaliação de soluções possuir um alto custo computacional, ela foi utilizada após a execução de cada procedimento do algoritmo de forma a garantir que a melhor solução obtida a qualquer momento fosse armazenada.
- Como critério de parada do algoritmo pode ser utilizado o número de gerações ou tempo de execução.

```
i = 0;
Construção da AGM de G* utilizando algoritmo de Kruskal
Iniciar a população P_0 através de uma das versões de Procedimento Construtor
Avaliar a população P_0
Enquanto não(Critério de Parada)
   i = i + 1;
   Cruzamento()
   Avaliar soluções de P_i
   Mutações() //diversas versões
   Avaliar soluções de P_i
   BuscasLocais() // seis versões
   Avaliar soluções de P_i
   Selecionar soluções em P_i - 1 para P_i
   Elitismo()
Fim-Enquanto
```

Algoritmo 5: Algoritmo evolutivo proposto.

3.4.2 GRASP

Nesse trabalho foi implementado um procedimento de filtro no GRASP, cujo objetivo é, a partir de um parâmetro η submetido ao algoritmo, executar η vezes o procedimento construtor de soluções antes da aplicação da fase de busca local. Dessa forma, apenas a melhor solução obtida nas η execução do procedimento construtor é filtrada e será submetida à segunda fase desse algoritmo, a busca local. Para os experimentos realizados no Capítulo 4 foram formados algoritmos combinando os procedimentos de construção de soluções iniciais e de buscas locais apresentados nesse capítulo.

Capítulo 4
Resultados Computacionais

O presente capítulo apresenta um conjunto de resultados computacionais obtidos a partir da aplicação dos algoritmos propostos nesta dissertação, considerando o problema de agrupamento com restrições de capacidade e conexidade. Todas as implementações foram feitas em Linguagem Ansi C, utilizando como ambiente de desenvolvimento o NetBeans 6.5. A execução dos experimentos foi realizada em um computador com um processador Intel Centrino Core 2 Duo 64 bits de 2.4 GHz, com 4 GB de memória RAM e sistema operacional Windows Vista.

4.1 Instâncias Utilizadas

Com a finalidade de avaliar os algoritmos propostos neste trabalho, utilizou-se um conjunto de instâncias obtidas a partir dos dados da amostra do censo demográfico de 2000 e da Contagem Populacional 2007. Para cada instância foram utilizados dois arquivos do tipo texto. O primeiro arquivo contém a informação de vizinhança entre os objetos (APONDs e municípios) e o segundo contém os atributos associados a estes objetos, inclusive o atributo relacionado com a capacidade.

O arquivo referente a vizinhança é utilizado para determinar a relação entre os vértices do grafo G (suas arestas) a partir do qual será construída a AGM *T*. Assim, cada objeto corresponde a um vértice do grafo ao qual também terá associado seus atributos. Tais atributos foram normalizados e utilizados no cálculo das distâncias d_{ij} nas arestas (i, j) de *G*.

Observa-se ainda, que para todas as instâncias utilizadas nesse trabalho, a variável total de pessoas também foi utilizada como variável de capacidade no processo de formação dos grupos. Assim, estipulou-se previamente, que a soma dos valores da variável total de pessoas associada aos agregados (grupos) não deveria ser inferior a um parâmetro *P* pré-estabelecido. A Figura 4.1 apresenta a relação de instâncias utilizadas.

	Código	Número de Objetos (Vértices)	Número de Vizinhanças (Arestas)
APOND	1	21	58
	2	61	286
	3	409	2020
	4	73	350
	5	14	46
Município	6	18	59
	7	89	363
	8	16	60
	9	57	236
	10	375	1769
	11	179	882
	12	74	357
	13	231	1172
	14	178	791
	15	121	567
	16	75	359
	17	114	502
	18	133	620
	19	195	868
	20	68	307
	21	181	843
	22	151	560
	23	86	388
	24	155	722
	25	461	2385
	26	285	1451

Figura 4.1: Relação de instâncias utilizadas nos experimentos.

4.1.1 Áreas de Ponderação

Nas instâncias referentes as áreas de ponderação (Figura 4.1), cada vértice representa uma área de ponderação e as arestas a vizinhança existente entre essas áreas. Os atributos utilizados nessas instâncias são: o total de casas, o total de domicílios, o total de pessoas, a soma dos rendimentos totais dos domicílios, a soma dos anos de estudo do responsável pelo domicílio, a soma dos rendimentos per-Capita dos domicílios e o número médio de anos de estudo do responsável pelo domicílio.

4.1.2 Municípios

As instâncias referentes aos municípios (Figura 4.1) possuem os seguintes atributos relacionados: a população total, a área plantada, total de pessoas com nível superior concluído e total de pessoas economicamente ativas. Nessas instâncias, os municípios correspondem aos vértices do grafo e as arestas representam as vizinhanças existentes.

4.2 Normalização dos Dados

Antes da realização dos experimentos desse trabalho, foi fundamental analisar os atributos que estão associados aos objetos e normalizá-los.

Os algoritmos de agrupamento geralmente utilizam duas estruturas de dados de entrada. A primeira representa os *n* objetos por meio de p atributos, através de uma matriz de ordem $n \times p$, em que as linhas correspondem aos objetos e as colunas aos valores de seus respectivos atributos (variáveis). A segunda estrutura corresponde a uma matriz $n \times n$ que contém as distâncias entre todos os objetos, que representam o grau de similaridade entre eles.

Porém, em diversas aplicações as magnitudes dos valores associados às características dos objetos podem traduzir diferentes estruturas de agrupamento. Por exemplo, considerando as variáveis idade (em anos), e altura (em metros), para um conjunto de indivíduos, verifica-se que o atributo idade teria mais relevância que o atributo altura, pelo fato da sua magnitude ser significativamente superior. Entretanto, se a altura for considerada em centímetros, haverá um grau de equilíbrio maior entre esses atributos. Dessa forma, antes da submissão das matrizes de atributos e de distâncias aos algoritmos propostos, torna-se necessária a padronização de seus dados.

Assim, o conjunto formado por n objetos, representado por $X=\{x_1, x_2, .., x_n\}$, em que cada objeto x_j possui p atributos $x_j=(x_j^1, x_j^2, ..., x_j^p)$ associados a valores originais devem ser convertidos a valores adimensionais. O processo de padronização dos dados consiste em calcular a média μ e o desvio padrão σ (Equação 1) dos valores associados a cada um dos atributos e, em seguida, aplicar a Equação 2 para obter os valores normalizados para cada atributo de cada objeto.

$$\sigma_h = \sqrt{\frac{\sum_{i=1}^{n} (x_1^h - \mu_h)^2}{n-1}} \quad \text{h = 1,..,p} \tag{1}$$

$$x_j^h = \frac{x_j^h - \mu_h}{\sigma_h} \text{, para o objeto } j \text{ e atributo } h \tag{2}$$

4.3 Algoritmos

Conforme o capítulo 3, os procedimentos e técnicas implementados neste trabalho de dissertação compõem um conjunto de algoritmos evolutivos e os algoritmos GRASP. E considerando que alguns destes procedimentos trabalham especificamente com a AGM e outros trabalham com o grafo original, tornou-se interessante a distinção de quatro grandes grupos de algoritmos: Algoritmos Evolutivos considerando apenas a AGM construída, Algoritmos Evolutivos considerando também o grafo original, Algoritmos GRASP considerando apenas a AGM construída e Algoritmos GRASP considerando também o grafo original. A Figura 4.2 apresenta uma legenda com as siglas utilizadas no decorrer desse capítulo para identificar os algoritmos e alguns de seus parâmetros.

Legenda	
AE	Algoritmo Evolutivo.
GRASP	Algoritmo GRASP.
AGM	Algoritmo que considera apenas a AGM.
K	Quantidade de clusters
Alfa	Tamanho da Lista Restrita de Candidatos.
BL	Busca Local concatenada com a Versão. Ex.: BL1 - Busca Local 1.
MUT	Mutação concatenada a versão. Ex.: MUT3 - Mutação 3.
Cross	Utilização do Cruzamento entre Soluções no AG.
Filtro	Utilização de Filtro no GRASP.
Const	Construtor concatenado com a versão. Ex.: C2 - Construtor 2.
S	Utilização de seleção no AG.

Figura 4.2: Legenda utilizada nos experimentos.

4.3.1 Algoritmos Evolutivos

As Figuras 4.3 e 4.4 apresentam, respectivamente, os algoritmos evolutivos considerando apenas a AGM e o grafo original. Além da combinação dos procedimentos de construção e busca local implementados nesse trabalho, devem ser informados aos algoritmos evolutivos, o critério de parada, quais versões dos procedimentos de construção e busca serão executados e os valores de alguns parâmetros:

- **Critério de parada:** pode ser utilizada a quantidade de gerações do algoritmo genético ou ser estipulado um tempo máximo de execução.
- **Tamanho da População:** quantidade de soluções por geração.
- **Procedimento Construtor:** todos os algoritmos devem especificar qual algoritmo construtor deve ser utilizado. Nesse trabalho foram implementadas três versões deste procedimento, sendo a restrição de conexidade atendida nas três.
- **Buscas Locais:** versões de procedimentos de Busca Local serão utilizadas pelo algoritmo.
- **Probabilidade para execução de Cruzamento e Mutação:** A execução destes procedimentos está condicionada a uma probabilidade, que corresponde a um valor real gerado aleatoriamente e que deve ser inferior a uma probabilidade submetida como parâmetro ao algoritmo.
- **Seleção utilizada:** Deve-se optar pelo método da roleta ou pelo torneio. No caso deste último, torna-se necessário informar também o tamanho da lista de soluções candidatas ao torneio.
- **Alfa:** Esse parâmetro é utilizado com objetivo de atribuir aos procedimentos construtores um comportamento semi-guloso, mediante a manipulação do tamanho da LRC (lista restrita de candidatos). O valor desse parâmetro será o limite superior da LRC considerando que, em situações específicas, os candidatos podem estar em quantidade inferior ao tamanho dessa lista.
-

Algoritmo Evolutivo (AGM)						
Sigla	**Const**	**BL1**	**BL4**	**BL5**	**MUT1**	**Cross**
AEAGM1	1	X	X	X	X	X
AEAGM2	2	X	X	X	X	X
AEAGM3	3	X	X	X	X	X

Figura 4.3: Algoritmos evolutivos que utilizam somente a AGM.

Algoritmo Evolutivo(Grafo original)				
Sigla	**Const**	**BL2**	**BL3**	**BL6**
AE1	1	X	X	X
AE2	2	X	X	X
AE3	3	X	X	X

Figura 4.4: Algoritmos evolutivos que utilizam também o grafo original.

4.3.2 GRASP

As Figuras 4.5 e 4.6 apresentam, respectivamente, os algoritmos GRASP considerando apenas a AGM e o grafo original. Além da combinação de procedimentos implementados, existem alguns parâmetros que devem ser informados, tais como:

- **Critério de parada:** pode ser utilizada a quantidade de iterações ou tempo de execução.
- **Buscas Locais:** quais versões de procedimentos de Busca Local serão utilizadas pelo algoritmo.
- **Alfa:** Esse parâmetro é definido de forma idêntica ao parâmetro Alfa dos Algoritmos Evolutivos.
- **Filtro:** Caso seja utilizado esse procedimento, deve ser especificada a quantidade de soluções que serão construídas na fase de construção, para que dentre essas, a melhor seja selecionada para a aplicação da busca local.

	GRASP (AGM)				
Sigla	**Const**	**BL1**	**BL4**	**BL5**	**Filtro**
GRASPAGM1	1	X	X	X	X
GRASPAGM2	2	X	X	X	X
GRASPAGM3	3	X	X	X	X

Figura 4.5: GRASPs que utilizam somente a AGM.

	GRASP (Grafo original)				
Sigla	**Const**	**BL2**	**BL3**	**BL6**	**Filtro**
GRASP1	1	X	X	X	X
GRASP2	2	X	X	X	X
GRASP3	3	X	X	X	X

Figura 4.6: GRASPs que utilizam também o grafo original.

Para os experimentos realizados nesse capítulo, a busca local do GRASP (2ª fase) foi executada a metade de vezes da quantidade de iterações ou do tempo de execução, conforme o parâmetro critério de parada submetido ao algoritmo. Essa proporção foi obtida empiricamente, após a realização de um conjunto de experimentos para um ajuste dos parâmetros.

4.4 Parâmetros dos algoritmos

Com base nos parâmetros descritos para cada grupo de algoritmos, foram estabelecidas as configurações apresentadas pelas Figuras Figura 4.7 e Figura 4.8. Além dessas configurações, é necessário informar aos Algoritmos qual será a quantidade *K* de clusters.

Algoritmo Evolutivo	
Tamanho da população	20
Quantidade de gerações	100
Probabilidade de ocorrência dos Cruzamentos	80%
Probabilidade de ocorrência das Mutações	5%
Capacidade mínima por *cluster* (fator de ajuste β conforme a Equação 3)	25%
Alfa	10

Figura 4.7: Outras configurações para os Algoritmos Evolutivos

Algoritmo GRASP	
Quantidade de Iterações	100
Quantidade de soluções construídas (Filtro)	20
Alfa	10

Figura 4.8: Outras configurações para os GRASPs

$$Capacidade_{Min} = (\beta / k).\sum_{i=0}^{V} v_i^x \qquad (3)$$

em que:

V é a quantidade de vértices.

v^x é o atributo referente a capacidade.

k é a quantidade de *clusters.*

β é um fator de ajuste.

4.5 Experimentos Realizados

Os experimentos realizados nesse trabalho têm como objetivo, através de execuções sistemáticas e análises probabilísticas, demonstrar a eficiência dos algoritmos propostos no que se refere à qualidade das soluções obtidas e a obtenção de soluções válidas (aquelas cuja as restrições do problema abordado são atendidas).

Com o objetivo de avaliar os procedimentos construtores um primeiro experimento considera três instâncias de diferentes tamanhos em quantidade de vértices e arestas (instâncias 4, 13 e 22) e três fatores de ajuste para o cálculo da capacidade mínima por *cluster* (de 30%, 50% e 70%).

Um segundo experimento considerou a aplicação de cada algoritmo apresentado na seção 4.3 dez vezes, em cada uma das instâncias apresentadas na Figura 4.1, para a obtenção de três *clusters*. A partir dos resultados obtidos são apresentadas analises avaliando os algoritmos em relação a obtenção de soluções de boa qualidade, soluções válidas e soluções de qualidade ruim. Além disso, foram avaliados também os tempos de execução dos algoritmos que utilizam apenas da AGM e o grafo original, as versões de procedimentos construtores bem como os Algoritmos (AEs e GRASPs).

Foi realizado uma análise de probabilidade empírica [Aiex et al., 2002] nas instâncias 4 e 13, selecionadas a partir de seus tamanhos (em quantidade de vértices e arestas). Inicialmente cada algoritmo foi executado cinquenta vezes em cada instância para obter informações como desvio padrão, média da aptidão por algoritmo e os alvos (fácil, médio e difícil). Em seguida cada algoritmo foi executado nas instâncias selecionadas cem vezes, tendo como o critério de parada o alcance ao alvo difícil ou o tempo máximo de execução predeterminado.

4.5.1 Experimentos com Procedimentos Construtores

Nos experimentos com procedimentos construtores foram utilizadas cinco versões. As versões 1, 2 e 3, apresentadas no Capítulo 3, utilizam o parâmetro Alfa 10, enquanto as versões 4 e 5 representam, respectivamente, os procedimento construtores 1 e 3, porém com parâmetro Alfa = 1 (comportamento guloso). Além disso, com o objetivo de analisar o impacto do fator de ajuste nos resultados, embora os experimentos realizados nas seção 4.5.2 e 4.5.3

tenham utilizado 25%, nesse experimento cada procedimento construtor foi executado cem vezes para cada instância utilizando os fatores de ajuste 30%, 50% e 70%.

Com o objetivo de analisar os resultados obtidos nesses experimentos foram utilizados gráficos com diferentes perspectivas, como obtenção de soluções válidas dos procedimentos por instância, por fator de ajuste e de instância por fator de ajuste.

A Figura 4.9 apresenta o gráfico da obtenção de soluções válidas por algoritmo construtor em cada instância. Nesses experimentos o procedimento construtor 4 não gerou nenhuma solução válida para nenhum dos fatores de ajuste. É possível observar também que o procedimento construtor 1 alcançou soluções válidas em apenas 2% das execuções para as instâncias 4 e 22 e em nenhuma execução para a instância 13. Já o procedimento construtor 2, embora tenha alcançado probabilidade acima de 90% para a instância 13, obteve baixas probabilidades para as demais instâncias. O procedimento construtor 3 e sua versão gulosa (5) obtiveram os melhores resultados para todas as instâncias, inclusive alcançando probabilidade de 100% nas instâncias 13 e 22.

Esse experimento indica que o algoritmo construtor 3 e sua versão gulosa (5) são mais eficientes na construção de soluções válidas que o procedimento construtor 2. É importante realizar essa comparação, uma vez que ambos os procedimentos trabalham na construção de soluções válidas para as duas restrições do problema abordado. Já o procedimento construtor 1, objetiva a formação de soluções de qualidade, sem garantir que a restrição de capacidade mínima por cluster seja satisfeita. Dessa forma os resultados nesse experimento apenas refletem o objetivo desse procedimento, e poucas soluções válidas são construídas.

A Figura 4.10 apresenta obtenção de soluções válidas por fator de ajuste. É importante novamente ressaltar que o aumento do fator de ajuste dificulta a obtenção de soluções válidas. Um elevado percentual de fator de ajuste pode, inclusive, tornar impossível a obtenção de soluções válidas. Ainda nessa figura observa-se que o procedimento construtor 1, além da baixa probabilidade de alcançar soluções válidas para o fator de ajuste 30%, possui probabilidade de 0% para os demais fatores. Para o procedimento construtor 2 a probabilidade de obtenção de soluções válidas é reduzida de acordo com o aumento do fator de ajuste. Já os procedimentos 3 e 5 se destacaram novamente, alcançando 100% de chance de obtenção de soluções válidas para os fatores de ajuste 30% e 50% e de 67% para o fator de ajuste 70%.

A Figura 4.11 apresenta a probabilidade de obtenção de soluções válidas por instâncias e fatores de ajuste. Novamente, observa-se que ocorre um decréscimo da probabilidade com o aumento do fator de ajuste, exceto para a instância 22 em que a probabilidade manteve-se a

mesma para os fatores de ajuste 50% e 70%. Além disso, para a instância 4 a probabilidade de obtenção de solução válida utilizando o fator de ajuste 70% é de 0%.

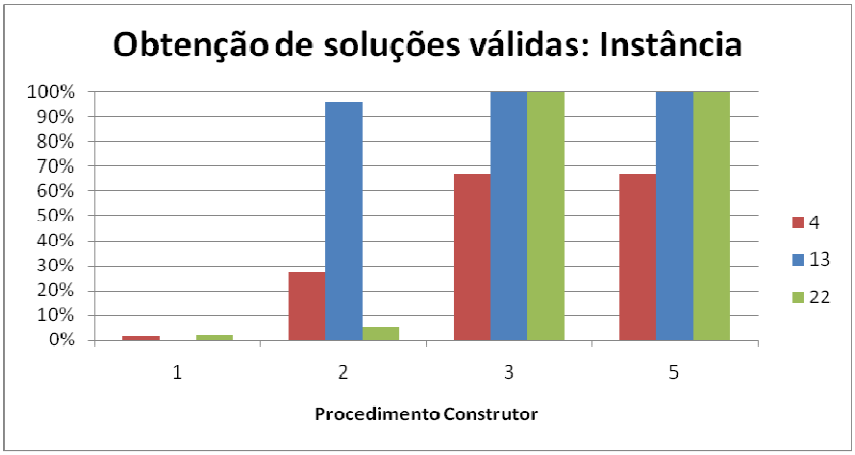


Figura 4.9: Obtenção de soluções válidas dos procedimentos construtores por instância.

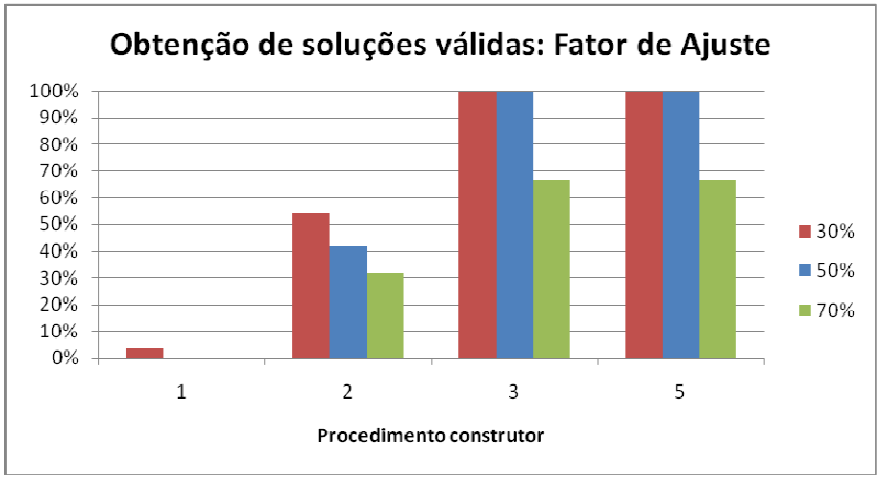


Figura 4.10: Obtenção de soluções válidas dos procedimentos construtores por fator de ajuste.

A Tabela 4.1 apresenta o valor da função aptidão (Fx), Gap (Equação 4) e o tempo de execução por algoritmo para cada instância, considerando apenas soluções válidas. É importante ressaltar que oito das nove melhores soluções desse experimento foram obtidas pelo procedimento construtor 3, e estão destacadas (em negrito e itálico) na Tabela 4.1.

$$Gap = 100 * \frac{solucao - solucao_{best}}{solucao_{best}} \quad (4)$$

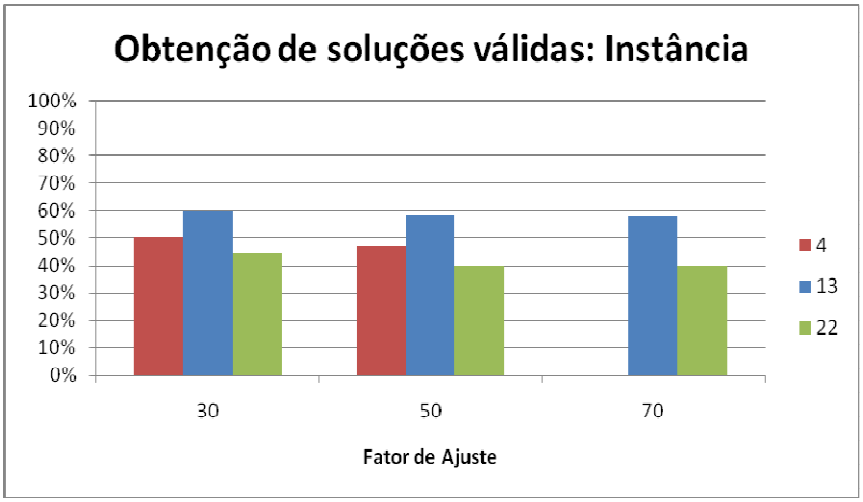


Figura 4.11: Obtenção de soluções válidas das instâncias por fator de ajuste.

Tabela 4.1: Aptidão, gap e tempo de execução por fator de ajuste.

Fator de Ajuste		Construtor	Instância	Fx	Gap	Tempo (s)
	30%	1	4	222,68	44%	1
		2	4	217,17	40%	1
		3	4	***154,91***	0%	1
		5	4	177,6	15%	1
		2	13	96,25	19%	1
		3	13	***80,99***	0%	1
		5	13	89,34	10%	1
		1	22	***37,6***	0%	1
		2	22	49,26	31%	1
		3	22	39,58	5%	1
		5	22	47,83	27%	1
	50%	2	4	217,17	26%	1
		3	4	***172,27***	0%	1
		5	4	174,04	1%	1
		2	13	98,01	15%	1
		3	13	***85,29***	0%	1
		5	13	93,84	10%	1
		3	22	***39,58***	0%	1
		5	22	43,21	9%	1
	70%	2	13	98,01	9%	1
		3	13	***90,08***	0%	1
		5	13	90,08	0%	1
		3	22	***36,32***	0%	1
		5	22	39,58	9%	1

4.5.2 Experimentos com os Algoritmos Propostos

A presente seção traz os resultados computacionais obtidos na execução de cada algoritmo proposto em cada instância utilizada nesse trabalho. As tabelas 4.2 e 4.3 mostram, a partir de dez execuções, as melhores soluções obtidas ($solucao_{best}$), o tempo de execução (em segundos) e o *Gap*. Somente soluções válidas são consideradas, ou seja, aquelas em que as restrições de conexidade e de capacidade mínima por cluster foram atendidas. Uma vez que o problema é de minimização, a solução com menor valor da função objetivo, considerando as

dez execuções, corresponderá à melhor solução. Nessas tabelas, as soluções cujo valor do *gap* foi inferior a 5% estão em negrito e sublinhadas. E as soluções cujo o gap está acima de 70% estão em negrito e com a célula da tabela em cinza.

A Figura 4.12 apresenta um gráfico com os quantitativos das soluções obtidas por algoritmo, classificadas conforme sua qualidade: M*elhores* (com Gap igual a 0%), *Interessantes* (com Gap maior do que 0% e menor ou igual a 5%) e *Ruins (Gap* acima de 70%). Uma vez que a Tabela 4.2 apresenta os melhores resultados obtidos pelos algoritmos em cada instância, o quantitativo máximo apresentado na Figura 4.12 é a quantidade de instâncias (26).

Ainda com base nessa figura, é possível observar que as soluções de boa qualidade, pertencentes as categorias *Melhores* e *Interessantes*, foram obtidas principalmente pelos algoritmos que consideram o grafo original. Além disso, nenhuma solução pertencente a categoria *Ruins* foi obtida por esses algoritmos. Em contrapartida, os algoritmos que consideram apenas a AGM são responsáveis por todas as soluções *ruins* obtidas nos experimentos. Observa-se também, que nesses algoritmos, a quantidade de soluções de boa qualidade obtidas é consideravelmente baixa em relação à de soluções ruins.

Tabela 4.2: Resultados dos AEs para a obtenção de 3 clusters

		Algoritmo Evolutivo (AE)											
		AE1		AE2		AE3		AEAGM1		AEAGM2		AEAGM3	
Instância	$solucao_{best}$	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)
1	10,38	***5%***	2	34%	1	***0%***	1	***5%***	2	**102%**	1	**102%**	1
2	56,99	19%	5	54%	4	***2%***	6	39%	2	28%	2	34%	2
3	733,11	***0%***	260	20%	201	***0%***	211	11%	208	21%	199	***0%***	241
4	97,05	24%	6	58%	4	***0%***	7	54%	3	**85%**	2	***5%***	2
5	2,37	15%	1	29%	1	15%	1	15%	1	26%	1	16%	1
6	9,68	***5%***	1	6%	1	***5%***	1	***5%***	1	8%	1	11%	1
7	20,41	***0%***	11	18%	11	***0%***	11	33%	4	**121%**	4	35%	4
8	6,21	***3%***	1	67%	1	***3%***	1	**86%**	1	**86%**	1	***0%***	1
9	7,86	9%	5	28%	5	9%	5	**92%**	1	67%	2	67%	2
10	51,02	13%	446	17%	390	***3%***	426	59%	192	57%	193	57%	178
11	41,32	***0%***	23	25%	24	23%	22	**79%**	8	**91%**	7	**73%**	7
12	30,29	***5%***	11	***0%***	9	51%	10	70%	3	**107%**	3	**78%**	3
13	40,62	24%	114	24%	143	***0%***	72	69%	46	59%	43	55%	47
14	42,52	***0%***	62	***0%***	53	***2%***	58	62%	27	37%	23	44%	23
15	34,29	***5%***	30	32%	25	***0%***	25	**124%**	9	**127%**	9	**94%**	9
16	25,81	***5%***	10	16%	9	***0%***	10	**85%**	3	40%	3	**80%**	3
17	20,09	***0%***	9	37%	9	16%	8	32%	2	43%	2	36%	2
18	60,87	***0%***	34	15%	30	***0%***	19	27%	9	56%	8	***0%***	11
19	47,68	***0%***	80	11%	78	***1%***	74	43%	29	68%	26	62%	27
20	115,00	2%	1597	3%	1011	0%	1478	**94%**	835	49%	461	60%	657
21	78,82	16%	75	32%	77	***0%***	65	**134%**	27	**85%**	24	52%	24
22	31,44	***0%***	36	11%	39	11%	35	***0%***	14	17%	15	***0%***	14
23	21,64	7%	12	11%	12	18%	13	59%	4	64%	4	39%	4
24	33,17	9%	48	7%	48	***0%***	47	***0%***	17	46%	15	52%	14
25	157,49	***5%***	852	6%	862	35%	865	**93%**	313	70%	318	**87%**	294
26	108,46	***0%***	209	20%	189	14%	188	**72%**	78	62%	79	68%	61

Tabela 4.3: Resultados dos GRASPs para a obtenção de 3 clusters.

		GRASP											
		GRASP1		GRASP2		GRASP3		GRASPAGM1		GRASPAGM2		GRASPAGM3	
Instância	$solucao_{best}$	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)	Gap	T(s)
1	10,38	26%	5	34%	4	14%	5	9%	5	**120%**	5	***0%***	5
2	56,99	21%	10	53%	5	***0%***	7	39%	6	22%	2	34%	4
3	733,11	***5%***	535	20%	263	***0%***	395	9%	500	24%	281	8%	452
4	97,05	45%	11	58%	6	***0%***	8	**79%**	8	**85%**	4	50%	5
5	2,37	11%	2	29%	1	***0%***	1	23%	1	18%	1	23%	1
6	9,68	***5%***	1	6%	1	***0%***	1	11%	1	***2%***	1	11%	1
7	20,41	13%	18	18%	15	***0%***	17	40%	9	**159%**	6	64%	8
8	6,21	67%	1	67%	1	40%	1	63%	1	61%	1	40%	1
9	7,86	***0%***	7	39%	7	***5%***	7	**148%**	4	67%	2	**83%**	4
10	51,02	13%	714	17%	504	***0%***	669	60%	430	58%	247	58%	342
11	41,32	***0%***	38	29%	29	9%	34	**71%**	18	**90%**	9	**72%**	15
12	30,29	***5%***	16	***0%***	11	7%	12	**83%**	7	**101%**	4	**91%**	6
13	40,62	20%	188	24%	179	23%	118	66%	103	**89%**	59	55%	93
14	42,52	17%	107	***0%***	69	***2%***	92	45%	62	40%	30	53%	45
15	34,29	***5%***	47	32%	32	***0%***	38	**91%**	24	**137%**	12	**103%**	18
16	25,81	9%	16	16%	12	***0%***	15	**101%**	7	43%	4	**93%**	6
17	20,09	9%	13	37%	11	29%	12	48%	6	48%	3	48%	5
18	60,87	***0%***	53	15%	39	8%	31	35%	24	58%	12	23%	22
19	47,68	***0%***	131	11%	100	***1%***	115	54%	66	**74%**	35	41%	57
20	115,00	2%	1764	3%	1298	0%	1693	**95%**	939	56%	569	**95%**	723
21	78,82	9%	123	32%	100	29%	99	65%	60	**90%**	30	**81%**	47
22	31,44	***0%***	64	16%	51	7%	57	20%	34	29%	19	20%	29
23	21,64	***0%***	20	11%	16	17%	19	52%	10	65%	6	50%	8
24	33,17	7%	81	7%	62	8%	74	47%	39	48%	21	42%	30
25	157,49	***0%***	1470	6%	1048	35%	1201	**72%**	680	**74%**	416	**86%**	668
26	108,46	***2%***	344	20%	249	17%	296	**71%**	170	66%	102	65%	148

A Figura 4.13 apresenta um gráfico que demonstra a eficiência dos algoritmos que utilizam o grafo original em relação aos algoritmos que consideram apenas a AGM. Nesse gráfico, os algoritmos que utilizam o grafo original alcançaram 41 soluções da categoria *Melhores (*$solucao_{best}$), 66 soluções da categoria *Interessantes* e nenhuma da categoria ruim, enquanto os algoritmos que utilizam apenas a AGM alcançaram apenas 7 soluções da categoria *Melhores*, 11 soluções da categoria *Interessantes* e 50 da categoria ruim. Dessa forma, os algoritmos que utilizam o grafo original além de possuir resultados melhores não

construíram soluções com *Gap* acima de 70%. Já os que utilizam apenas a AGM resultaram, essencialmente, em soluções da categoria *ruins*.

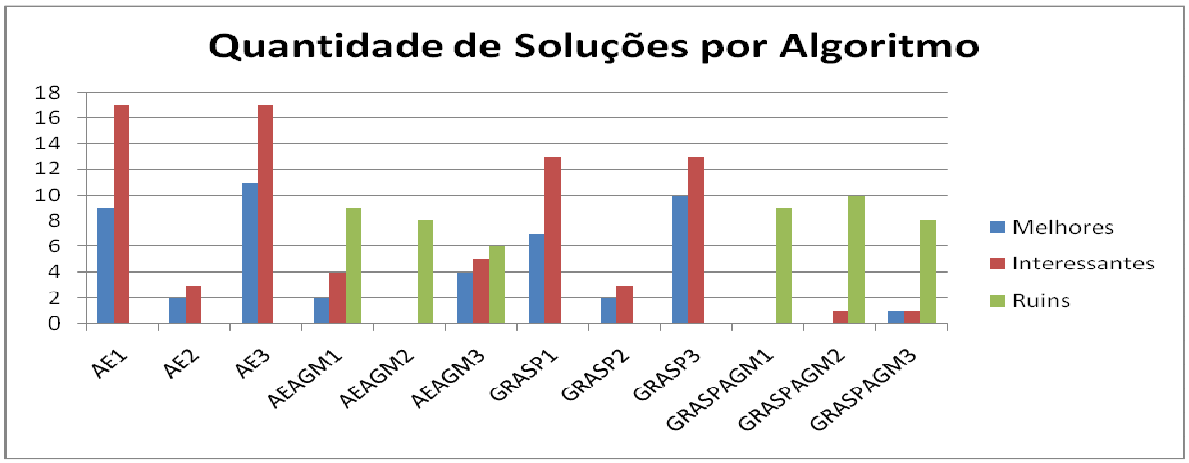


Figura 4.12: Quantidade de soluções por algoritmo.

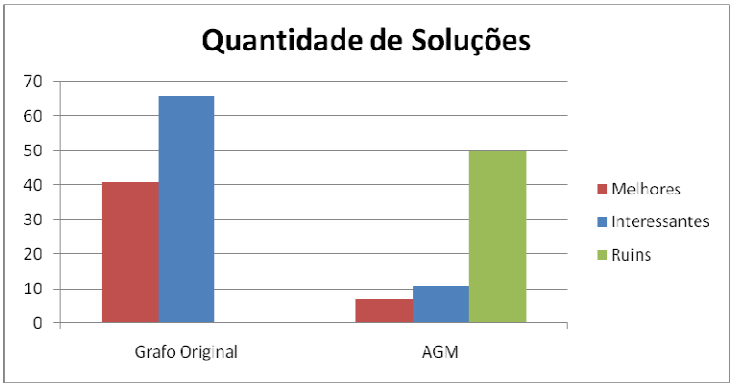


Figura 4.13: Quantidade de soluções (AGM ou o grafo original).

Novamente com base nos gráficos das Figuras 4.12 e 4.14 é possível observar a superioridade dos AEs em relação aos GRASPs, uma vez que os AEs alcançaram soluções de boa qualidade em maior quantidade (categorias Melhores e Interessantes) e soluções ruins em menor quantidade quando comparados com os algoritmos GRASP.

Em relação ao gráfico da figura 4.13, é possível observar que os algoritmos que utilizam o grafo original produzem soluções melhores do que os algoritmos que utilizam a AGM.

A Figura 4.15 apresenta um gráfico com os quantitativos de soluções por versão de procedimento construtor. É possível observar que o procedimento construtor 2 alcançou

poucas soluções de boa qualidade e que o procedimento construtor 3 destacou-se, produzindo uma maior quantidade de soluções de boa qualidade (Melhores e Interessantes) e menor quantidade das soluções ruins.

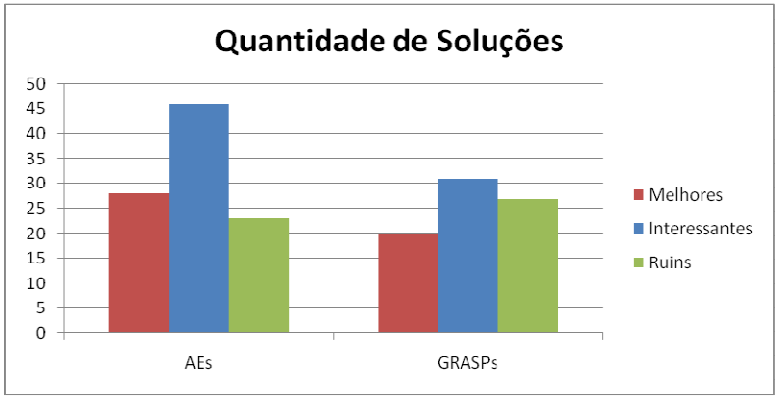


Figura 4.14: Quantidade de soluções dos AEs e GRASPs.

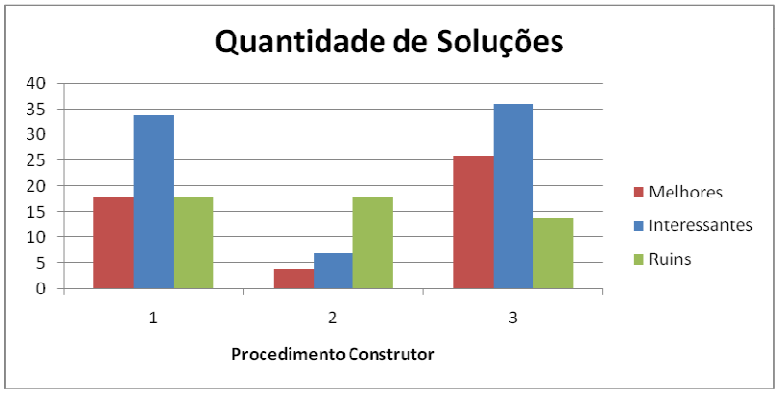


Figura 4.15: Quantidade de soluções por procedimento construtor.

Além das análises referentes aos resultados obtidos, é interessante estabelecer uma comparação entre os tempos de execução utilizados. Assim, a Figura 4.16 apresenta um gráfico com os tempos de execução médios dos AEs e dos GRASPs nas instâncias utilizadas. Os AEs obtiveram maior quantidade de soluções de boa qualidade com tempos de execução inferiores aos dos GRASPs.

Em relação ao comparativo realizado entre os algoritmos que consideram apenas a AGM e o grafo original (Figura 4.17), os algoritmos que utilizam o grafo original alcançaram

melhores resultados às expensas de tempos de execução consideravelmente superiores aos dos algoritmos que utilizam a AGM.

Concluindo a análise do tempo, o gráfico apresentado na Figura 4.18 indica que o tempo de execução dos algoritmos que utilizam o procedimento construtor 3 está compreendido entre os tempos médios de execução dos procedimentos 1 e 2. Sendo o procedimento construtor 2 o que consumiu o menor tempo em média.

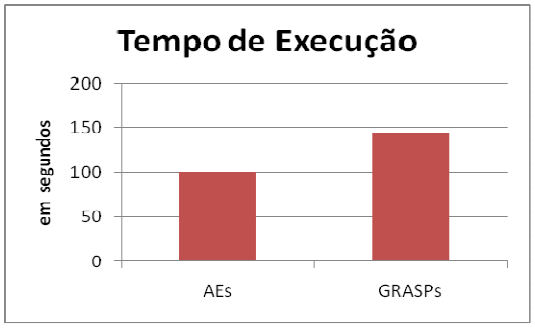


Figura 4.16: Tempo de execução dos AEs e GRASPs.

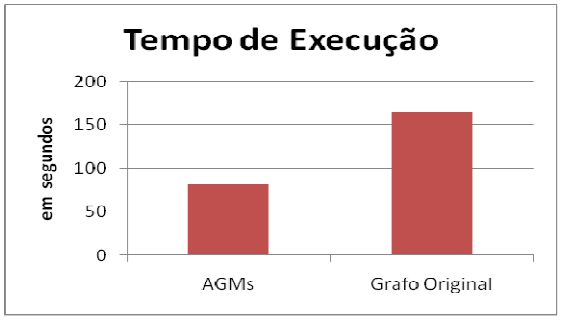


Figura 4.17: Tempo de Execução dos algoritmos (AGM e o grafo original).

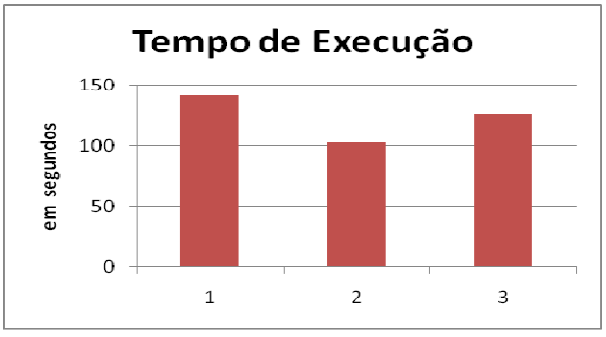


Figura 4.18: Tempo de execução dos algoritmos por procedimento construtor utilizado.

Uma vez que os experimentos realizados consideram apenas a obtenção de 3 clusters, torna-se importante a realização de experimentos com outros números de clusters para demonstrar o desempenho e a eficiência dos algoritmos. Assim, as instâncias utilizadas no experimento para avaliar os procedimentos construtores, 4, 13 e 22 foram submetidas a um novo experimento, em que cada algoritmo foi executado dez vezes em cada instância para a obtenção de soluções com 5 e 8 clusters.

A Tabela 4.4 apresenta as melhores aptidões obtidas por instância considerando soluções com 3, 5 e 8 clusters. Através dessa tabela, observa-se que à medida que o número de clusters é aumentado, há uma redução no valor da função aptidão. Um fato normalmente observado em problemas de agrupamento.

Tabela 4.4: Aptidão por instância para 3, 5 e 8 grupos.

		Total de Clusters		
		3	**5**	**8**
Instância	**4**	97,05	87,7	56,43
	13	40,62	31,31	27,36
	22	31,44	22,83	22,11

Para demonstrar a relação entre a aptidão da solução e a quantidade de clusters, as instâncias 4, 13 e 22 foram submetidas a um novo experimento, em que foram construídas e avaliadas soluções com o parâmetro K assumindo valores entre um e o total de vértices da instância. A Figura 4.19 apresenta o gráfico com as aptidões das soluções por quantidade de clusters. É importante ressaltar que nesse experimento a restrição de contigüidade foi atendida, porém, com o objetivo de apresentar todas as aptidões calculadas, a restrição de capacidade mínima por cluster foi ignorada. Esse experimento foi realizado utilizando apenas o procedimento construtor 3, ou seja, a solução não foi submetida a buscas locais, cruzamento, mutação ou qualquer outro procedimento.

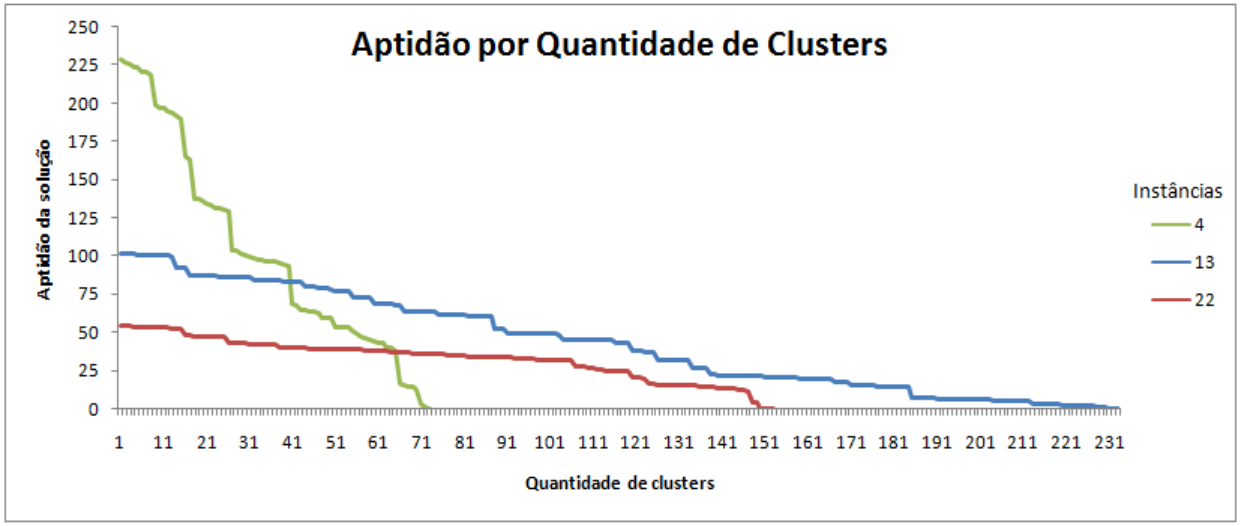


Figura 4.19: Aptidão por quantidade de clusters.

4.5.3 Análise Probabilística

A presente seção traz uma nova avaliação dos algoritmos mediante a realização de uma análise de probabilidade empírica. Para essa análise, foram definidos alvos fáceis, médios e difíceis associados, respectivamente, com a aptidão pior solução válida, com a média das aptidões e com a melhor aptidão obtida. Para tal análise, foram utilizadas as instâncias 4 e 13, selecionadas a partir de seus tamanhos em quantidade de vértices e arestas.

Após a determinação dos alvos, cada algoritmo foi executado 100 vezes em cada instância desse experimento. No instante em que o algoritmo alcança uma solução com valor de aptidão igual ou inferior ao alvo, seu tempo de processamento era inserido em uma lista *L* e, em seguida, inicia-se uma nova execução. Após a realização das execuções dos algoritmos, os tempos de processamentos são ordenados e, para cada tempo de processamento t_i obtêm-se a probabilidade $p_i=(i-\frac{1}{2})/100$.

Após o cálculo da probabilidade plotam-se os pontos $z_i=(t_i,p_i)$ um gráfico, o que estabelece uma distribuição de probabilidade empírica do tempo que cada algoritmo consome para alcançar o alvo pré-determinado.

Para cada instância são apresentados os gráficos com a probabilidade empírica ao longo do tempo de execução. É apresentado também um gráfico de barras com a probabilidade acumulada por algoritmo ao final de sua execução para cada instância.

A Tabela 4.5 apresenta o Gap da melhor solução, da solução média, da pior solução e o desvio padrão dos resultados obtidos nos experimentos realizados para a obtenção dos alvos supracitados. Ainda com base nessa tabela, os três melhores resultados estão em destaque (com a formatação em negrito, sublinhado e itálico), enquanto os piores resultados estão com suas respectivas células com cor de fundo cinza. É possível observar que os piores resultados estão concentrados nos algoritmos que consideram apenas a AGM e que utilizam os procedimentos construtores 1 e 2. Uma grande parte dos melhores resultados foi obtida pelos algoritmos que utilizam o grafo original e o construtor 3.

Em relação ao desvio padrão, é importante ressaltar que sua análise deve ocorrer em conjunto aos gaps apresentados. Assim, os algoritmos em destaque devem possuir Gaps menores e, ao mesmo tempo, desvios padrão baixos.

O algoritmo AE3, por exemplo, embora possua desvio padrão alto (aproximadamente 18), tem seus gaps (média e pior solução) relativamente baixos. As piores soluções obtidas por esse algoritmo possuem gaps inferiores a melhor solução de outros algoritmos. Em contrapartida no algoritmo GRASPAGM3, por exemplo, não houve variação entre as aptidões das soluções obtidas (desvio padrão 0), porém a melhor solução obtida possuem o gap alto, de 50% e 55% para as instâncias 4 e 13, respectivamente.

Embora o algoritmo AE3 também tenha se destacado por obter resultados interessantes nas duas instâncias utilizadas nesse experimento, o algoritmo que obteve melhores resultados para a instância 4 foi o GRASP3, que alcançou a melhor solução conhecida, a pior solução possui gap de 35% e, além disso, possui desvio padrão inferior ao do AE3.

Tabela 4.5: Aptidões dos algoritmos e desvio padrão para as instâncias 4 e 13.

	Instância 4				Instância 13			
	Gap			Desvio	Gap			Desvio
	Melhor	Média	Pior	Padrão	Melhor	Média	Pior	Padrão
AE1	24%	57%	78%	12,44	***21%***	25%	30%	0,77
AE2	58%	58%	58%	0	24%	24%	***28%***	0,53
AE3	***5%***	***31%***	***51%***	17,98	***0%***	***23%***	***28%***	4,22
AEAGM1	50%	84%	108%	14,83	60%	81%	113%	6,16
AEAGM2	58%	94%	121%	18,39	58%	92%	115%	5,92
AEAGM3	***22%***	57%	96%	22,18	55%	60%	67%	1,38
GRASP1	45%	51%	61%	6,32	***21%***	***23%***	***25%***	0,69
GRASP2	58%	58%	58%	0	24%	***24%***	***24%***	0
GRASP3	***0%***	***21%***	***35%***	13,96	24%	26%	***28%***	0,38
GRASPAGM1	74%	81%	96%	6,59	63%	73%	85%	2,29
GRASPAGM2	85%	99%	110%	6,54	55%	85%	90%	3,79
GRASPAGM3	50%	***50%***	***50%***	0	55%	55%	55%	0

4.5.3.1 Análise Probabilística: instância 4

As figuras 4.21, 4.22 e 4.23 apresentam os gráficos de análise probabilística da instância 4 associada aos alvos fácil, médio e difícil, respectivamente. Já a Figura 4.24 apresenta a probabilidade acumulada dos alvos utilizados ao final da execução dos algoritmos.

Com base no gráfico referente ao alvo fácil é possível observar que todos algoritmos alcançam 100% de probabilidade acumulada em aproximadamente 15 segundos de execução. Excetuando-se os algoritmos GRASP2 e GRASPAGM2, que não alcançaram o alvo em nenhuma execução. Os GRASPs que utilizaram o procedimento construtor 2 alcançaram menos soluções válidas em menor quantidade e de pior qualidade, com aptidões superiores, inclusive, ao alvo fácil obtido anteriormente. Isto demonstra que a construção de soluções inicias de boa qualidade cooperam com os demais procedimentos tais como as buscas locais.

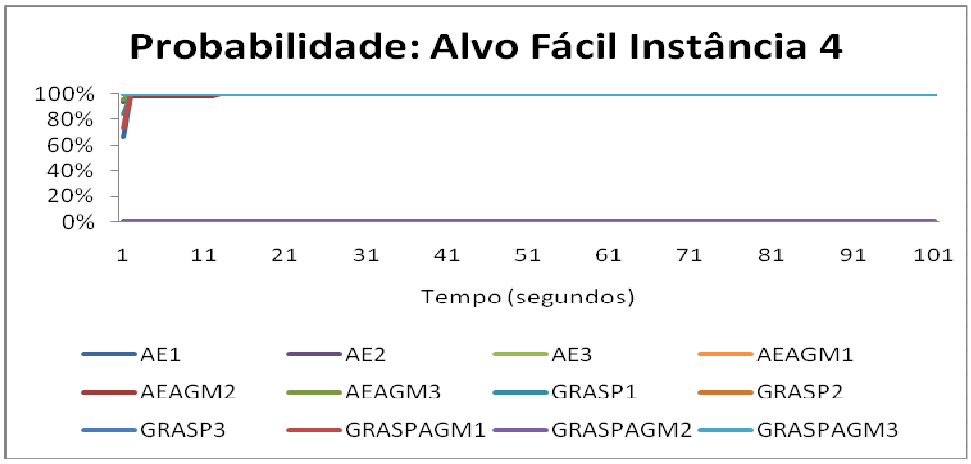


Figura 4.21: Probabilidade acumulada alvo fácil instância 4.

Quando é considerado o alvo médio (Figura 4.22), observa-se que todos os algoritmos que utilizaram o procedimento construtor 3 alcançaram a probabilidade de 100% em aproximadamente 10 segundos de execução. Em relação aos algoritmos que utilizaram o procedimento construtor 1 apenas os AEs alcançaram o alvo, em que as probabilidades acumuladas dos algoritmos AE1 e AEAGM1 foram de 100% e 76%, respectivamente. Já os algoritmos que utilizaram o procedimento construtor 2, apenas o AEAGM2 alcançou o alvo médio, e com probabilidade acumulada de apenas 2%.

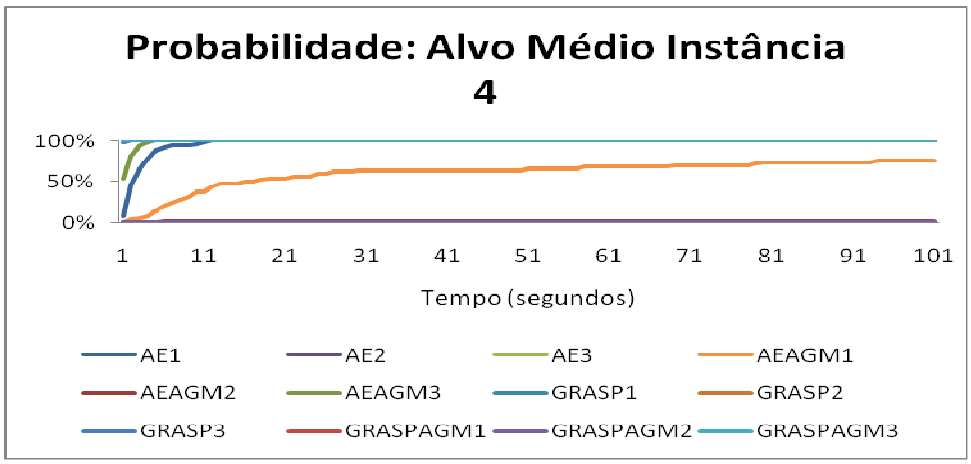


Figura 4.22: Probabilidade acumulada alvo médio instância 4.

Em relação ao alvo difícil é possível observar que nenhum algoritmo que utilizou o procedimento construtor 2 alcançou o objetivo. Além disso, apenas o algoritmo AE1 que utilizou o procedimento construtor 1 alcançou o alvo. E todos os algoritmos que utilizam o

procedimento construtor 3 alcançaram o alvo difícil, destacando-se ainda, o algoritmo AE3 que obteve 100% de probabilidade acumulada ao final do termino de execução.

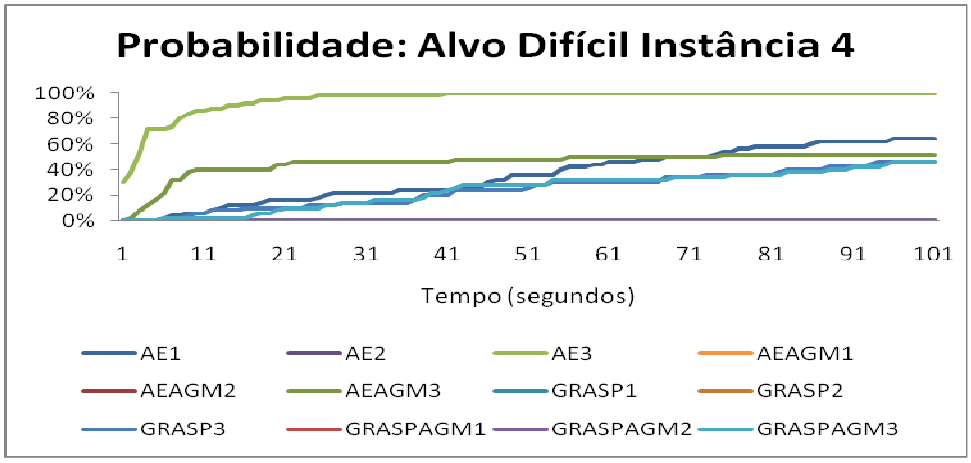


Figura 4.23: Probabilidade acumulada alvo difícil instância 4.

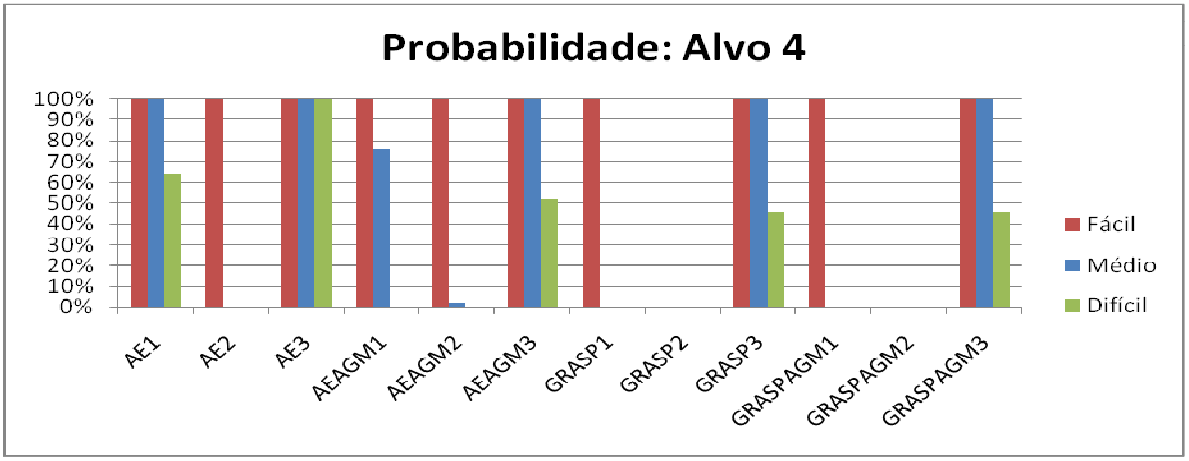


Figura 4.24: Probabilidade acumulada por categoria de soluções – instância 4.

4.5.3.2 Análise Probabilística: instância 13

As figuras 4.25, 4.26 e 4.37 apresentam os gráficos de análise probabilística da instância 13 associada aos alvos fácil, médio e difícil, respectivamente. Já a Figura 4.28 apresenta a probabilidade acumulada dos alvos utilizados ao final do tempo de execução.

Com base no gráfico referente ao alvo fácil (Figura 4.25) é possível observar resultados semelhantes aos dos experimentos realizados com a instância 4. Os algoritmos alcançam

100% de probabilidade acumulada, excetuando-se os algoritmos GRASP2 e GRASPAGM2 que não alcançaram o alvo em nenhuma execução.

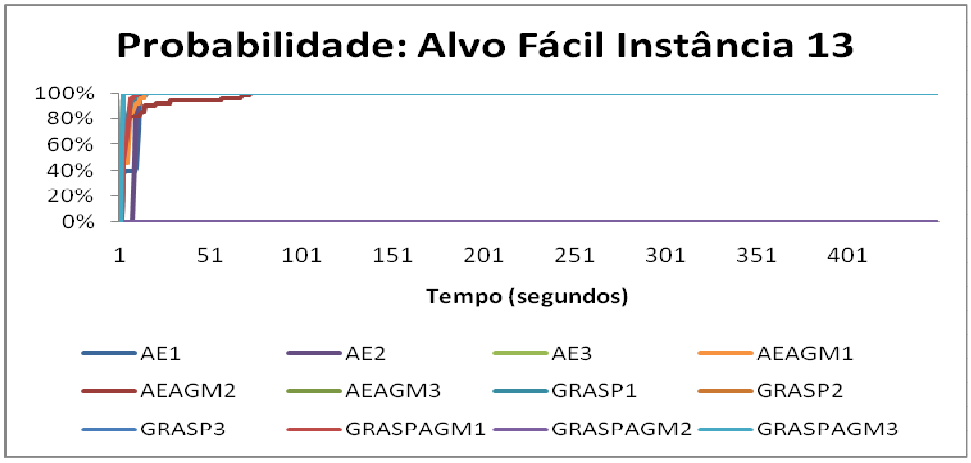


Figura 4.25: Probabilidade acumulada alvo fácil instância 13.

Nos experimentos relativos ao alcance do alvo médio, observa-se um destaque dos AEs em relação aos GRASPs. Os algoritmos que utilizaram o procedimento construtor 3 alcançaram 100% de probabilidade ao final do tempo de execução. Dos algoritmos que utilizaram o procedimento construtor 1 e 2, apenas os AEs alcançaram o alvo. E os algoritmos AE1, AEAGM1, AE2 e AEAGM2 obtiveram probabilidade acumulada de 100%, 100%, 100% e 14%.

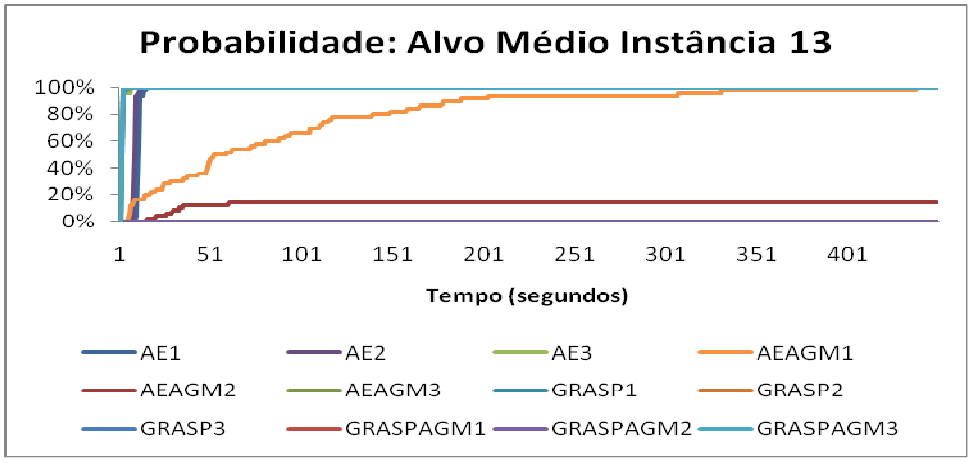


Figura 4.26: Probabilidade acumulada alvo médio instância 13.

Finalmente, no caso do alvo difícil, é possível observar que todos os algoritmos que utilizaram o procedimento construtor 3 alcançaram 100% de probabilidade acumulada ao

termino do tempo de execução. Todos os AEs alcançaram o alvo difícil porém, em relação aos GRASPs, isso ocorreu apenas com as versões que utilizam o procedimento construtivo 3.

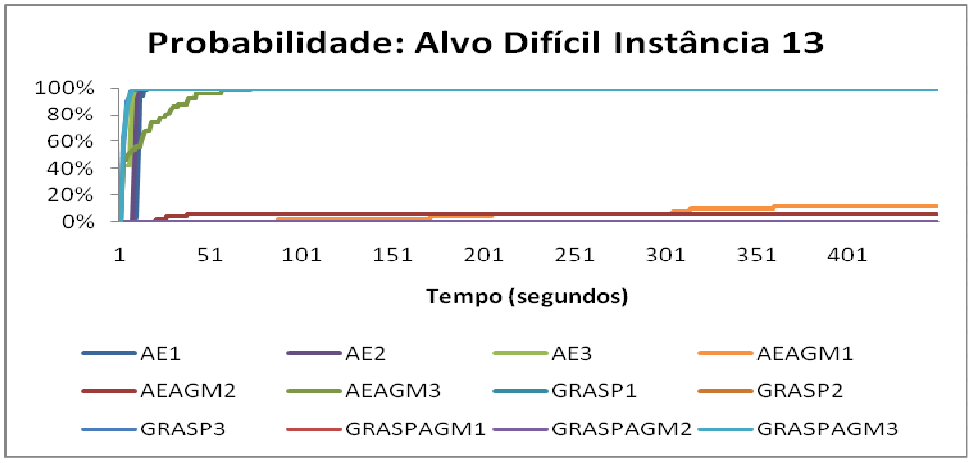


Figura 4.27: Probabilidade acumulada alvo difícil instância 13.

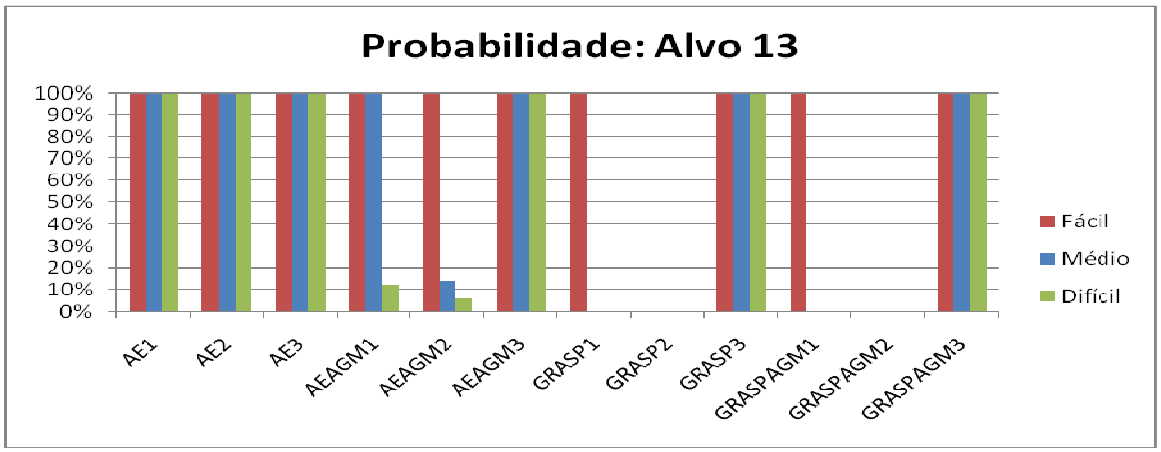


Figura 4.28: Probabilidade acumulada por categoria de soluções – instância 13.

Com base nos resultados apresentados nesse capítulo, observou-se que os AEs que utilizam o grafo original e o procedimento construtor 3 tiveram uma melhor performance em relação aos demais algoritmos.

Capítulo 5
Conclusões e Trabalhos Futuros

Essa dissertação apresentou os algoritmos heurísticos propostos para a resolução do problema de agrupamento com restrições de capacidade e conexidade. O capítulo 1 contém uma breve introdução sobre o problema abordado e as técnicas e algoritmos propostos para sua resolução.

O capítulo 2 abordou uma breve descrição do problema de agrupamento, suas definições, complexidade, requisitos desejáveis para obtenção de bons resultados e medidas para avaliar a qualidade das soluções deste problema. Também foi apresentado nesse trabalho um particular problema de agrupamento, que agrega também as restrições de capacidade e conexidade. Além disso, foi realizada uma revisão da literatura sobre o problema abordado e os estudos de caso, quais sejam: os Problemas de Criação de Áreas de Ponderação Agregadas e de Agregados de Municípios.

O Capítulo 3 apresentou os algoritmos propostos para a resolução do problema de agrupamento abordado nesta dissertação. Tais algoritmos foram desenvolvidos a partir do estudo de Algoritmos Evolutivos e do GRASP**.**

O Capítulo 4 trouxe informações sobre as instâncias utilizadas e sobre os parâmetros que foram considerados para os algoritmos que foram propostos. Os experimentos realizados para avaliar os procedimentos construtores (seção 4.5.1) demonstraram a superioridade da terceira versão, que se destacou na obtenção de soluções válidas por instância, por fator de ajuste e, inclusive, na obtenção de soluções de boa qualidade.

Nos experimentos nos quais cada algoritmo foi executado dez vezes para cada instância utilizada (seção 4.5.2), os algoritmos que utilizam o grafo original se destacaram em relação aos algoritmos que consideram apenas a AGM. Os algoritmos que utilizam o grafo original não produziram nenhuma solução ruim (com gap superior a 70%). Além disso, a quantidade de soluções boas (com gap até 5%) foi consideravelmente superior à quantidade de soluções produzidas pelos algoritmos que consideram apenas a AGM. Tal fato foi decorrente da maior liberdade de migrações de vértices, o que pode facilitar tanto a regeneração de soluções quanto a melhoria de sua qualidade, com a minimização de sua aptidão.

Já os algoritmos que consideram apenas a AGM ficam extremamente condicionados a estrutura da árvore construída, o que dificulta a migração de vértices e que pode, até mesmo, impossibilitar a formação de soluções válidas.

Ainda nos experimentos da seção 4.5.2 pode-se observar que os AEs se destacaram em relação aos GRASPs não apenas no que concerne à qualidade das soluções, mas também pelo tempo de CPU consumido.

O procedimento construtor 3 novamente se destacou entre os demais, alcançando soluções boas (melhor solução obtida ou com gap até 5%) em todas os algoritmos em que foi utilizado. Além disso, nos algoritmos que produziram soluções ruins (gap superior a 70%), esse procedimento obteve os menores quantitativos em relação as demais versões, como pode ser visto na Figura 4.12.

Em relação aos experimentos de análise de probabilidade empírica (seção 4.5.3) novamente os algoritmos que utilizam o procedimento construtor 3 se destacaram, e todos os esses algoritmos alcançaram os alvos difíceis. Na instância 13, em particular, a utilização desse procedimento garantiu o alcance ao alvo difícil, ou seja, em todos os algoritmos que utilizaram esse procedimento, a probabilidade de alcançar o alvo foi de 100%.

Com base nos resultados obtidos nos experimentos supracitados, observa-se a superioridade dos AEs em relação aos GRASPs e, além disso, do procedimento construtor 3 em relação as demais versões.

A utilização do grafo original demonstrou-se mais eficiente que a utilização apenas da AGM, uma vez que ela possibilita uma maior liberdade para migrações de vértices entre clusters, o que reflete tanto na regeneração de soluções penalizadas quanto da obtenção de soluções de melhor qualidade.

Dessa forma, através dos experimentos e análises apresentadas, conclui-se que o algoritmo AE3 é a melhor alternativa para a resolução do problema abordado.

5.1 Trabalhos Futuros

Desde o início desse trabalho, na fase de revisão dos trabalhos da literatura e principalmente durante a fase implementação e realização dos experimentos, várias possibilidades de pesquisa apresentaram-se promissoras. Ainda que não tenha sido possível

realizar tais pesquisas, em função do tempo limitado, é interessante citá-las para a realização de trabalhos futuros e eventuais desdobramentos desse trabalho.

- **Novos experimentos:** realização de novos experimentos utilizando todos os algoritmos em cada instância desse trabalho para a obtenção de soluções com a quantidades K de *clusters* entre 4 e 8.
- **Técnicas de penalização das soluções:** os algoritmos apresentados nesse trabalho atuam na busca por soluções que atendam as duas restrições do problema abordado, ou seja, não penalizadas. Embora a restrição de conexidade seja garantida em todos os procedimentos, a restrição de capacidade mínima por cluster pode não ser satisfeita. Dessa forma, soluções de boa qualidade que possuam algum cluster com capacidade inferior ao pré-estabelecida são armazenadas pelo algoritmo, não sendo, porém, apresentadas nos resultados obtidos. Definir em que grau de penalidade o cluster se encontra ou seja, o quão distante está do valor desejado pode ser uma alternativa para a obtenção de soluções melhores e, inclusive, alvo de novos algoritmos de busca local para sanar penalidades e minimizar a função objetivo. Outra alternativa, potencialmente interessante, é inserir na função de aptidão do algoritmo uma variável referente a eventuais penalidades existentes nos clusters, objetivando a qualificação de soluções mesmo se penalizadas. Para isto, normalmente agrega-se na função objetivo um fator de penalização que possui valor proporcional a inviabilidade da solução. Ou seja, se diferença entre o valor fixado e o valor da restrição for alto, o fator de penalidade é alto e vice-versa.
- **Reconexão de caminhos (*path relinking*):** em linhas gerais, esta técnica corresponde a uma busca local que gera soluções intermediárias entre duas soluções, denominadas solução base e solução alvo, em busca da obtenção de soluções de qualidade superior. No contexto desse trabalho, por exemplo, a seleção de duas soluções de boa qualidade pode gerar soluções intermediárias melhores, ou seja, com menor valor da função objetivo [Bastos *et al.*, 2005].
- **Estudos para calibração dos parâmetros de entrada:** tanto no GRASP, quanto nos algoritmos evolutivos, o ajuste dos diversos parâmetros de entrada é de fundamental importância para a produção de soluções de boa qualidade. Durante a realização dos experimentos houve uma grande dificuldade na definição desses parâmetros e, algoritmos para a calibração desses parâmetros podem ser extremamente úteis e, inclusive, auxiliar na obtenção de resultados melhores.

- **Programação inteira:** realizar experimentos com uma formulação de programação inteira que encapsule as restrições associadas ao problema abordado nesse trabalho, e que ao mesmo tempo, minimize uma outra função de aptidão.
- **Realização de experimentos utilizando instâncias artificiais:** verificar a eficiência dos algoritmos em relação a características especificas das instâncias, tais como: quantidade de vértices e arestas e sua topologia (anel, estrela ou mista). Para isso, torna-se necessário o desenvolvimento de algoritmos para geração de instâncias, em que as características supracitadas são parâmetro de entrada.
- **Utilização de outras metaheurísticas:** desenvolver e realizar experimentos para outras metaheurísticas, tais como: ILS (*Iterated Local Search*), VNS (*Variable Neighborhood Search*), Busca Tabu ou uma combinação destas metaheurísticas.

Referências

[Abramowitz, 1964] Abramowitz, M. *Handbook of Mathematical Functions*. No. 55 in mathematical Applied Series. National Bureau of Standards, 1964.

[Agrawal *et al.*, 1998] Agrawal, R., Gehrke, J., Gunopulos, D., et al., *Automatic Subspace Clustering on High Dimensional Data for Data Mining Applications*, In: *Proceedings of the ACM SIGMOD Conference on Management of Data*, pp. 94-105, Seattle, Washington, USA, June, 1998.

[Aiex et al., 2002] Aiex, R. M., Resende, M. G. C., Ribeiro, C. C.. *Probability distribution of solution time in grasp: An experimental investigation. Journal of Heuristics*, 8 vol. 3, 2002.

[Alvanides *et al.*, 2002] Alvanides, S. Openshaw, S. Rees, P. *Designing Your Own Geographies. The Census Data System*, Jonh Wiley & Sons, 2002.

[Assunção *et al.*, 2002] Assunção , R. M; Lage, J. P.;Reis , A. E.. Análise de Conglomerados Espaciais Via Árvore Geradora Mínima. Revista Brasileira de Estatística, vol. 63, n. 220, 2002, 7-24.

[Assunção *et al.*, 2006] Assunção, R. M.; Neves , M. C.; Câmara, G., Freitas , C. Efficient regionalization techniques for socio-economic geographical units using minimum spanning trees. International Journal of Geographical Information Science 20(7): 797-811, 2006.

[Bastos *et al.*, 2005] Bastos, L. O., Ochi, L. S.; Macambira, E.M. *GRASP with Path Relinking for the SONET Ring Assignment Problem. Proc. of the 5th International Conference on Hybrid Intelligent Systems* (HIS2005), pp. 239-244, 2005.

[Baum, 1986] Baum, E.B. *Iterated descent: A better algorithm for local search in combinatorial optimization problems. Technical report Caltech*, Pasadena, CA. Manuscript, 1986.

[Baxter, 1981] Baxter, J. *Local optima avoidance in depot location*. Journal of the Operational Research Society 32, 1981.

[Beasle *et al.*, 1993] Beasley, D., Bull, D. R. e Martin R. R. *An Overview of Genetic Algorithms: Part I, Fundamentals. University Computing*, vol. 15, no. 2, pp. 58-69, 1993.

[Boley, 1999] Boley, D.; Gini, M.; Gross, R.; Han, E.H.; Hastings, K.; Karypis, G.; Kumar, V.; Mobasher, B. & Moore, J. *Partitioning based clustering for Web document categorization. Decision Support Systems*, 27, 329-341, 1999.

[Brito *et al.*, 2004] Brito, J. A. M.; Montenegro , F. M. T.; Brito L. R.; Passini , M. M.. Uma formulação de programação inteira para o problema de criação de áreas de ponderação agregadas, Anais do SOBRAPO, 2004.

[Censo Demográfico, 2000] Censo Demográfico 2000. Primeiros Resultados da Amostra, Parte I, 2001, IBGE/CDDI.

[Chiou and Lan, 2001] Chiou, Y.C. Lan, L.W. *Genetic Clustering Algorithms*. European Journal of operational Research 135, 2001.

[Cole, 1998] Cole, R. M. *Clustering with Genetic Algorithms. Master's thesis, Department of Computer Science*, University of Western Australia, 1998.

[Cormen, 2002] Cormen, T.; Leiserson , C.; Rivest , R.; Stein , C., *Algoritmos - Teoria e Prática*, 2002.

[Dias and Ochi, 2003] Dias, C.R.; & Ochi, L.S.. *Efficient Evolutionary Algorithms for the Clustering Problems in Directed Graphs. Proc. of the IEEE Congress on Evolutionary Computation* (IEEE-CEC), 983-988. Canberra, Austrália, 2003.

[Dias, 2004] Dias, C.R. *Algoritmos Evolutivos para o Problema de Clusterização de Grafos Orientados - Desenvolvimento e Analise Experimental*. Dissertação de Mestrado, Universidade Federal Fluminense (UFF), 2004.

[Doval *et al.,* 1999] Doval, D., Mancoridis, S. e Mitchell, B. S. *Automatic Clustering of Software Systems using a Genetic Algorithm. In 1999 International Conference on Software Tools and Engineering Practice* (STEP '99), 1999.

[Ester *et al*., 1995] Ester, M., Kriegel, H.-P., and Xu, X., *A Database Interface for Clustering in Large Spatial Databases*, In: *Proceedings of the 1st International Conference on Knowledge Discovery in Databases and Data Mining (KDD-95)*, pp. 94-99, Montreal, Canada, August, 1995.

[Fasulo, 1999] Fasulo, D. *An Analysis of Recent Work on Clustering Algorithms. Technical Report,Dept. of Computer Science and Engineering*, Univ. of Washington, 1999.

[Feo and Resende, 1995] Feo, T.A.; Resende, M.G.C., *Greedy randomized adaptive search procedures*,*Journal. of Global Optimization,* 6, 109—133, 1995.

[Ferreira et al., 2007] Ferreira, C. S., Ochi, L. S., and Macambira, E. M. *Desenvolvimento e Análise Experimental de Heurísticas GRASP para uma Generalização do Problema da Árvore Geradora Mínima*. XXXIX SBPO - Simpósio Brasileiro de Pesquisa Operacional, Fortaleza-CE, 2007.

[Gersting, 2004] Gersting, J. L. *Fundamentos matemáticos para a ciência da computação: um tratamento moderno de matemática discreta*. 5. ed. Rio de Janeiro: LTC, 2004.

[Glover and Kochenberger, 2002] Glover, F. ; Kochenberger, G. A. *Handbook of Metaheuristics*. Kluwer Academic Publishers, 2002.

[Glover, 1986] Glover, F. *Future paths for integer programming and links to artificial*

intelligence. Computers and Operations Research 13, 1986.

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

[Goldschimidt and Passos, 2005] Goldschimidt R.; Passos, E. *Data Mining: um guia prático*. Editora Campus, Rio de Janeiro: Elsevier, 2005.

[Han and Kamber, 2001] Han, J., e Kamber, M., *Cluster Analysis*. In: Morgan Kaufmann. Publishers (eds.), Data Mining: Concepts and Techniques, 1 ed., chapter 8, New York, USA, Academic Press, 2001.

[Hartuv and Shamir, 1999] Hartuv, E.; Shamir, R.. *A Clustering Algorithm based on Graph Connectivity. Technical Report*, Tel Aviv University, Dept. of Computer Science, 1999.

[Holland, 1975] Holland, J. H. *Adaptation in Natural and Artificial Systems. University of Michigan Press*, Ann Arbor, 1975.

[Hruschka and Ebecken, 2001] Hruschka, E. R., Ebecken, N. F. F. *A Genetic algorithm for cluster analysis*. Submitted to: IEEE Transactions on Evolutionary Computation , 2001.

[Karypis and Kumar, 1998] Karypis, G.; Kumar, V. *A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. Journal of Parallel and Distributed Computing*, 48, 71-95, 1998.

[Larose, 2006] Larose, D. T. Discovering Knowledge in Data, *An Introduction to Data Mining*. John Wiley & Sons, 2005.

[Linden, 2006] Linden, R. *Algoritmos Genéticos: Uma importante ferramenta da Inteligência Computacional*, Brasport, 2006.

[Liu, 1968] Liu, G.L.. *Introduction to combinatorial mathematics*. McGraw Hill, 1968.

[Maheshwari and Shen, 1998] Maheshwari, P.; Shen, H. *An efficient clustering algorithm for partitioning parallel programs. Parallel Computing*, 24, 893-909, 1998.

[Michalewicz, 1992] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1992.

[Mladenovic and Hansen, 1997] Mladenovic, N.; Hansen, P. *Variable Neighborhood Search. Computers Operations Research* 24,1997.

[Mladenovic, 1995] Mladenovic, N. *A Variable neighborhood algorithm: a new metaheuristic for combinatorial optimization*. Abstracts of papers presented at Optimization Days, Montréal, 1995.

[Neves, 2003] Neves, M.C. *Procedimentos Eficientes para Regionalização de Unidades Socioeconômicas em Bancos de Dados Geográficos*. Tese de Doutorado, INPE, São

José dos Campos, 2003.

[Ng and Han, 1994] NG, R. T. e Han, J., *Efficient and Effective Clustering Methods for Spatial Data Mining*, *In: Proceedings of the 20th International Conference on Very Large.Data Bases* (VLDB), pp. 144-155, Santiago, Chile, September, 1994.

[Openshaw, 1977] Openshaw, S., A geographical solution to scale and aggregation problems in regionbuilding, partitioning and spatial modelling. Transactions of the Institute of British Geographers (New Series), 2, pp. 459–472, 1977.

[Ramos, 2002] Ramos, F.R. *Análise Espacial de Estruturas Intra-urbanas: o caso de São Paulo. Dissertação de mestrado*, INPE, São José dos Campos, 2002.

[Scheuerer, 2006] Scheuerer, S. W. , R. *A scatter search heuristic for the capacitated clustering problem*. *European Journal of Operational Research* vol. 169, 2006.

[Semaan *et al.*, 2008] Semaan, G. S., Ochi, L. S., Brito, J. A. M.; Montenegro, F. *An Efficient Evolutionary Algorithm for the Aggregated Weighting Areas Problem*. Proc. of the International Conference on Engineering Optimization, 2008.

[Semaan *et al.*, 2009] Semaan, G.S., Brito, J. A. M.; Ochi, L. S. *Um algoritmo evolutivo hibrido aplicado ao problema de clusterização em grafos com restrições de capacidade e contiguidade*. Anais do IX Congresso Brasileiro de Redes Neurais e Inteligência Computacional (IX CBRN),Ouro Preto/MG, 2009.

[Shieh and May, 2001] Shieh, H.M., May, M.D. Solving the Capacitated Clustering Problem with Genetic Algorithms. Journal of the Chinese Institute of Industrial Engineers, Vol. 18, 2001.

[Silva *et al.*, 2004] Silva , A. N.; Cortez, B. F; Matzenbacher , L. A.. Processamento das Áreas de Expansão e Disseminação da Amostra no Censo Demográfico 2000, Textos para Discussão, número 17, 2004, IBGE/DPE/COMEQ.

[Soares, 2004] Soares, S.S.R.F. *Metaheurísticas para o Problema da Clusterização Automática*, Dissertação de Mestrado, Universidade Federal Fluminense (UFF), Niterói-RJ, 2004.

[Trindade and Ochi, 2006] Trindade, A.R.; Ochi, L.S. *Um algoritmo evolutivo híbrido para a formação de células de manufatura em sistemas de produção*. Abstracts in Operational Research / Statistical Theory and Methods Abstracts) vol. 26(2), 2006.